AN INTERACTIVE COMPUTATIONAL APPROACH TO 3D LAYOUT DESIGN OF SINGLE-FAMILY HOUSES BY EVOLUTIONARY ALGORITHMS

A THESIS SUBMITTED TO THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ANIL SAKARYALI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF ARCHITECTURE
IN ARCHITECTURE

DECEMBER 2017

Approval of the thesis:

AN INTERACTIVE COMPUTATIONAL APPROACH TO 3D LAYOUT DESIGN OF SINGLE-FAMILY HOUSES BY EVOLUTIONARY ALGORITHMS

Submitted by **ANIL SAKARYALI** in partial fulfillment of the requirements for the degree of **Master of Architecture in Architecture Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Elvan Altan _____
Head of Department, **Architecture**

Assist. Prof. Dr. İpek Gürsel Dino _____
Supervisor, **Department of Architecture, METU**

Examining Committee Members:

Prof. Dr. Zeynep Mennan _____
Department of Architecture, METU

Assist. Prof. Dr. İpek Gürsel Dino _____
Department of Architecture, METU

Prof. Dr. Arzu Gönenç Sorguç _____
Department of Architecture, METU

Prof. Dr. Mine Özkar Kabakçıoğlu _____
Department of Architecture, ITU

Assist. Prof. Dr. Yasemin Afacan _____
Interior Architecture and Environmental Design, Bilkent University

Date: December 5, 2017

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: ANIL, SAKARYALI

Signature

ABSTRACT

AN INTERACTIVE COMPUTATIONAL APPROACH TO 3D LAYOUT DESIGN OF SINGLE-FAMILY HOUSES BY EVOLUTIONARY ALGORITHMS

Sakaryalı, Anıl
M.Arch, Department of Architecture
Supervisor: Assist. Prof. Dr. İpek Gürsel Dino
December 2017, 184 pages

Customized design is an important feature for single-family houses (SFH). Differently, current stage of the housing industry is generally limited to the standard houses by tract developments. In this way, design tools for non-expert users can provide a strong alternative to the current mode of house production. Certain generative design tools can provide customized house solutions according to the requirements of occupants. A certain problem in this case is the presented level of interaction for the non-expert occupants. A study on the current generative approaches to non-expert design tools showed that generative approaches present a limited interaction for the user due to the limits of their solution space and the required level of expertise for their operation. This research aims to develop a user-friendly design tool for non-expert designers that can work with appropriate solution spaces. In this way, this research presents a new evolutionary computational design tool, Ho-Gen (House Generator), which assists in the design exploration of single-family house layouts through an interactive work process. Ho-Gen is capable to generate multi-floor SFH layouts with geometric and topological criteria. Ho-Gen's interactive interface allows the designer to guide the generation process within the intermediate states to make changes in the problem definition together with the possibility to modify generated solutions. Ho-Gen is tested with two conceptual SFH layout problems with a varying number of layout elements in an increasing level of complexity. The results show that Ho-Gen can generate a variety of valid layouts for the conceptual stage in architecture.

Keywords: computational layout design, single-family house, design exploration, interactive genetic algorithm, evolutionary computation

ÖZ

MÜSTAKİL EV ÖLÇEĞİNDE BİNA YERLEŞİMİ TASARIMINA YARDIMCI ETKİLEŞİMLİ BİR GENETİK ALGORİTMA

Sakaryalı, Anıl
Yüksek Lisans, Mimarlık Bölümü
Tez Yöneticisi: Assist. Prof. Dr. İpek Gürsel Dino
Aralık 2017, 184 sayfa

Müsrakil ev tipolojisinde yaratılan mekanın kullanıcıya özgü olması büyük önem taşımaktadır. Bunun aksine günümüzde müstakil ev üretimi, mimarların kısıtlı bir çevreye verebildikleri hizmetten ötürü genellikle müteahhit tarafıdan üstlenen standart evlere karşılık gelmektedir. Bu durumda bina kullanıcısına kendi evini tasarlayabileceği düzeyde destek sunan hesaplamalı tasarım araçlarının geliştirilmesi mevcut duruma güçlü bir alternatif yaratmaktadır. Bu konuda geliştirilen mevcut yöntemlere bakıldığında kullanıcının ihtiyaç verilerinden özgün ev tasarımları ortaya çıkartabilecek kadar gelişkin modellere rastlanmıştır. Aynı zamanda bahsedilen modeller gerektirdikleri tasarım bilgisi ve içerdikleri sınırlı çözüm alanı doğrultusunda sınırlı bir etkileşim imkanı sunmaktadır. Bu tez, kullanıcının hesaplamalı bina yerleşimi sürecinde kontrolünü arttıracak müstakil ev ölçeğinde çalışan etkileşimli genetik algoritma yöntemini, Ho-Gen'i tanıtmaktadır. Tasarım aracı, kullanıcının geometrik ve topolojik girdilerine göre, ayrık müstakil ev tipolojisine uygun farklı alanlı ve çok katlı kütle modelleri geliştirebilmektedir. Ho-Gen bu kriterlerin yönetimi için etkileşimli bir arayüz sunmakta ve kullanıcının programın duraksadığı ara zamanlarda problem tanımını ve çıkan kütle modellerini değiştirmesine olanak vermektedir. Geliştirilen model, farklı karmaşıklık ve ölçekte iki konsept tasarım probleminde test edilmiştir. Ho-Gen, alınan sonuçlara göre konsept tasarım problemlerine gereken çeşitlikte ve uygunlukta örnekler vermeyi başarmaktadır.

Anahtar Sözcükler: hesaplamalı bina yerleşimi, müstakil ev, tasarım araştırması, etkileşimli genetik algoritma

To my parents.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my thesis supervisor Assist. Prof. Dr. İpek Gürsel Dino, for her guidance, support and patience during the production of this thesis. Her guidance has deeply affected my understanding on computational design from the last year of my undergraduate education and has given me the driving force behind working on this subject.

I would also like to thank to my jury members: Prof. Dr. Zeynep Mennan, Prof. Dr. Arzu Gönenç Sorguç, Prof. Dr. Mine Özkar Kabakçıoğlu, and Assist. Prof. Dr. Yasemin Afacan for their comments and discussions through the research process.

I would like to thank to my mother Melek Sakaryalı, for her limitless belief in my success, emotional support, and patience during the stressful times. I would not be able reach this degree without her support in every step of my life.

Last but not least, I would like to thank to my girlfriend Gizem Akköse for all her love, motivation, support, and understanding; my closest friends Nihan Avcı and Ömer Akyüz for the precious joy they bring into my life; and my colleagues İnanç Eray, Pınar Güvenç, Uğur İmamoğlu, and Egemen Onur Kaya for their help in my hard times.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Motivation

The practice of architectural design is constantly changing with the increase in computational power and the growing extent of research on developing models to integrate computation into the architectural design process. Computational tools for architecture have gone beyond their usual use in representation and documentation which are not directly design related works. Instead, architects take the advantage of computational support just within the design process as an amplifier of their cognitive capabilities. Computation helps architects to work with complex data structures, explore more solutions, and give confident design decisions. Yet, another group of computational design tools aims to shift the focus from the practice of architectural design to the practitioner himself/herself. The target audience of such computational tools is the non-expert designers who are the occupants or future users. The purpose of such a shift is to provide the necessary support for non-expert designers to make them capable in using their own creative and personal ideas for less complex architectural problems. Similarly, this research aims to develop a computational tool that supports non-experts in the architectural design of single-family houses (SFH).

SFH is a suitable architectural typology for the design participation of non-experts in terms of its simplicity and the level of required customization. SFH is a freestanding building that is occupied by a single family. The design process of SFH is usually simpler in terms of the size of the architectural program associated with common domestic needs. Additionally, the design process is very user-centric because of the importance of occupant's lifestyle, aesthetic understanding, and cultural background. This personal information is best known by the occupants themselves. The high level

of customization and the relative simplicity of SFH create a potential for a design participation model. This potential can create a strong alternative to the current state of the housing market.

Nowadays, owning a private architect designed house is a priority for most people. Despite of the importance of SFH in architectural history with iconic SFH examples such as Fallingwater House or Villa Savoye, today architect's participation in the housing industry is very limited. The largest portion of SFH industry is currently held by tract house developers. These developers respond to housing needs usually with cookie-cutter projects. Such projects are usually related to a monotony of box-shaped buildings with little or no alteration. Therefore, current state of the housing industry is away from reflecting the individual taste and needs of their occupants.

Alternatively, a certain number of architectural approaches reveal the advantages of a bottom-up procedure to housing. During the recent years, the popularity of participatory design in SFH has been rising due to a number of projects. A significant example is Quinta Monroy Social Housing[1] project in Chile by Pritzker winning architect Alejandro Aravena. In Quinta Monroy, Aravena provides the occupants only one-half of the house which leaves the occupant the other half to design and expand over time.[2] Another prominent example is WikiHouse[3], an open source project that shares construction documents and assembly manuals of a set of houses. WikiHouse aims to create a new housing industry where occupants or small communities can build for themselves with abundant materials and prevalent manufacturing techniques such as CNC.[4] One other example is the advancing potential of prefabricated houses with the developments in mass customization industry. These developments can provide a powerful alternative to the current

---

[1] "Quinta Monroy / ELEMENTAL," ArchDaily, December 31, 2008, http://www.archdaily.com/10775/quinta-monroy-elemental/.

[2] "Quinta Monroy / ELEMENTAL."

[3] "WikiHouse," WikiHouse, accessed December 19, 2017, https://wikihouse.cc/.

[4] "WikiHouse."

problematic state of SFH industry.

In the abundance of open-source knowledge about construction systems, the increasing prevalence of mass customization and architects' growing intent in participatory design can reverse the top down hierarchy in housing industry. In this context, an important issue is the capability of future occupants in the design of their own houses. The growing extent of house options both in open-source and in prefabricated housing domains are not going reach far away from the current level of standardization without the proper integration of the occupants in the design process. Future occupants should find ways to combine, subtract, and reinvent the available house options in this abundance to reach a level of customization that satisfies their individual needs. In this context the development of computational tools that supports the occupants in the design of their own house come into prominence.

1.2 Background

The idea of user-centered architectural design has a rich historical background both in academic and professional environments. The first collective research on the user participation in architectural design was realized in "Design Participation" conference in 1971 with the worldwide attendance of multi-disciplinary participants.[5] Developing computational tools to support non-experts in architectural design was already an issue in "Design Participation" conference that is presented by architects Yona Friedman and Nicholas Negroponte.[6] Despite of this early historical background, it is not possible to come up with a significant project that takes the advantage of such a model. The recent popularity of such model can be a result of the technological advancements in manufacturing, construction, and computation.

A user-centered design process requires a high-level customization in the

---

[5] Yanki Lee, "Design Participation Tactics: Redefining User Participation in Design," in *Design Research Society International Conference*, 2006, 1.

[6] Theodora Vardouli, "Who Designs?," in *Empowering Users through Design* (Springer, 2015), 23.

manufacturing systems to meet the special needs of the occupants. On the contrary, the efficiency of the past means of production has depended on standardization. Mass production achieves high cost efficiency through the rapid manufacturing of standardized products within specialized factories. This understanding in manufacturing is still shifting with the developments in mass customization. Computer-aided manufacturing systems such as laser cutting and 3D printing are developing in the way to reach the economic efficiency of mass production while creating an increased variety in the products. Nevertheless, certain challenges are still evident for mass customization. According to Zha and Lu,[7] an important challenge for the mass customization model is the decision process for the level of variety.[8] In this way, the process of gathering, analyzing and processing the information about the requirements and preferences of the customers becomes important. This process puts multiple parties in between the manufacturers and the occupants. The information is gathered through market surveys, generalized by data analysts, and turned into actual design by the architects. Alternatively, developing "user-friendly" computational tools that empower users to design their own houses creates a more direct communication between the occupants and the means of production.

Computational tools completely take away the need for surveys or data analysis process as the occupants assess their needs on their own. On the other hand, the exclusion of architects or expert designers does not mean that they have no impact over the design process. The development of computational tools is itself a design problem that requires the integration of architects and expert designers. Architects, as the creator of the toolkits, reflect their subjective opinions over the toolkits by setting limits over user's control over the design process. They also specify the possible actions that the user can interact with tool's interface. In this way, the working principle of design toolkits is closely related with developer's understanding on the

[7] Xuanfang Zha and Wen F. Lu, "Knowledge Support for Customer-Based Design for Mass Customization," in *Artificial Intelligence in Design '02* (Springer, Dordrecht, 2002), 407–29, https://doi.org/10.1007/978-94-017-0795-4_20.

[8] Zha and Lu, 407.

design process and the design problem. The subjectivity of these issues brought out alternative computational tool approaches in terms of the type of user interaction and design support.

The types of current computational tools can be specified as configurators, drafting tools, and generative tools. First, configurators present a list of houses for the occupants that are designed by the prefabricated developers. Users limit these options with very basic assumptions such as the number of bedrooms. Developers offer the users small decorative customizations over the selected house. A notable example of configurators is Postgreen Homes.[9] Postgreen Homes offers a web-based interface for the individual apartments in their multi-family residences. Apartment types vary according to the project and they present users customization in the furniture and appliances. Toll Brothers[10] is another construction business that offers a configurator for customization. This configurator is a checklist interface for small and predefined layout alterations excluding the hard construction work.

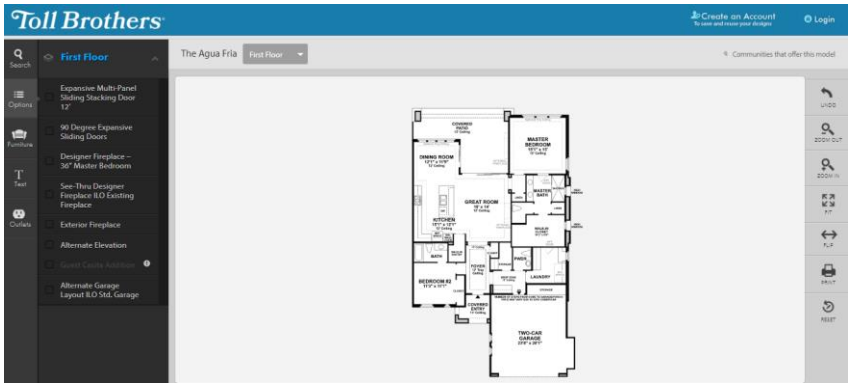


Figure 1. Toll Brother's web-based configurator toolkit offers a list of possible variations (on the left) for the user. Configurator gives simultaneous feedback on the layout to inform the user about the effects of the changes.
(Retrieved from: https://security.tollbrothers.ml-scp.com/FloorPlan/Details/157298, Accessed on 26.12.2017.

[9] "Customize a Green, Modern, Affordable Home by Postgreen Homes - CUSTOMIZE - Passive Houses," accessed December 25, 2017, http://customize.postgreenhomes.com/?s=0.

[10] "New Construction Homes for Sale | Toll Brothers® Luxury Homes," accessed December 25, 2017, https://www.tollbrothers.com/.

Second, drafting tools provide users a simplified and user-friendly CAD or BIM interface. The occupants, with the help of drafting tools, can design their own house completely from the beginning or start from an initial layout template. As an example, Express modular[11] offers Project Homestyler, a home editing software that is supported by Autodesk. The interface works with a library of building, furnishing, and stylistic objects which the user can pick and arrange on the canvas. Certain drafting tools provide additional support through knowledge-based systems. According to Corne and Bentley,[12] knowledge-based systems work with a built-in knowledge base that integrates expert knowledge from the real professionals in the involved domains. As an example, Williams[13] developed a system that trains computational critics for the design of SFHs. This approach differs from the general knowledge-based systems because William's computational critic develops its knowledge base by learning from a collection of real world house examples together with an architect.[14] The architect goes through each example and points out the mistakes in the layout such as the placement of a component.[15] McLeish[16] took the advantage of William's computational critics in his participatory design model for a SFH. McLeish model understands the changes in the layout and updates itself both in 3D and in terms of the computational critics.

[11] "Express Modular," accessed December 10, 2017, http://expressmodular.com/dragonfly_editor.php.

[12] David Corne and Peter Bentley, *Creative Evolutionary Systems*, The Morgan Kaufmann Series in Artificial Intelligence (San Francisco, CA: Morgan Kaufmann, 2002).

[13] Reid E. (Reid Edward) Williams, "Training Architectural Computational Critics by Example" (Massachusetts Institute of Technology, 2003), http://dspace.mit.edu/handle/1721.1/16691.

[14] Williams, 27.

[15] Williams, 27.

[16] Thomas John McLeish, "A Platform for Consumer Driven Participative Design of Open (Source) Buildings" (Massachusetts Institute of Technology, 2003), http://dspace.mit.edu/handle/1721.1/32250.

Figure 2. McLeish's toolkit provides a physical model for the user to arrange appliances and furniture while seeing the results in 3D perspective in real time (left). McLeish also takes the advantage of computational critics that checks the layout for mistakes and present relevant solutions (right). (McLeish, 2003)

Lastly, generative tools present users a direct design solution or a list of choices to select from after gathering knowledge about their lifestyle and domestic needs. A generative system works with an underlying generative logic to create a set of useful and viable solutions according to the needs of the occupant. For example, Huang and Krawczyk[17] developed a generative tool that asks occupants a set of general questions for the required spaces and get to a specific set of questions about the finishes and appliances in the end. [18] At the end of every level, the occupant is given a range of alternative solutions that satisfy their answers.[19] Shape grammars are another generative approach for non-expert design tools. A shape grammar consists of a set of rules to transform an initial geometrical entity in consequent steps.[20] An important example of shape grammar use in non-expert computational tools is

---

[17] Chuen-huei Joseph Huang and Robert Krawczyk, "A Choice Model of Consumer Participatory Design for Modular Houses," 2007.

[18] Huang and Krawczyk, 682,684.

[19] Huang and Krawczyk, 681.

[20] George Stiny, *Shape : Talking about Seeing and Doing* (Cambridge, Massachusetts : The MIT Press, [2006], 2006).

Duarte's[21] approach that generates houses in the style of Alvaro Siza's Malagueira Housing project. The grammar is capable of creating the same houses that Siza designed and a wide range of similar alternatives according to user preferences.

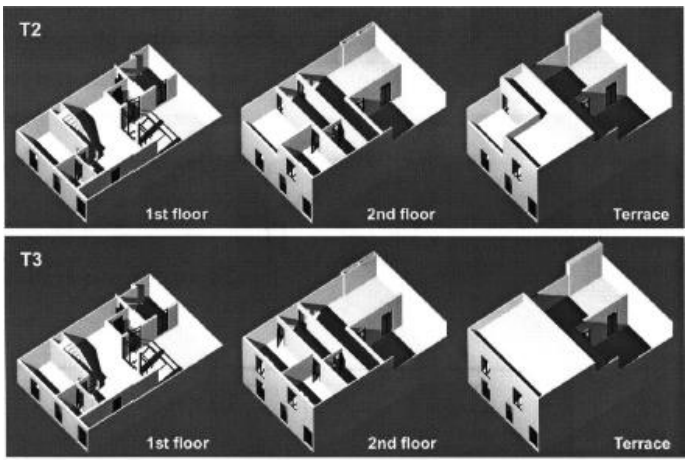


Figure 3. Duarte's shape grammar despite the hard coded geometric rules can create a variety of house forms similar in the style of Alvaro Siza's Malagueira Housing project. (Duarte, 2001)

1.3 Problem Statement

Computational tools for non-experts create an opportunity to abide the limitations of the current housing market by presenting a co-design environment for the occupants to develop their own living environments. Despite of the growing extent of computational tools for non-expert designers, these approaches show certain limitations in the essential user interaction process. It is logical that these tools present a certain level of limitation as an expert level of freedom can be overwhelming for a non-expert user. Nevertheless, such limitations should not prevent non–expert designers to present their design abilities and creative ideas.

[21] José Pinto Duarte, "Customizing Mass Housing : A Discursive Grammar for Siza's Malagueira Houses" (Massachusetts Institute of Technology, 2001), http://dspace.mit.edu/handle/1721.1/8189.

According to Von Hippel and Katz,[22] computational toolkits for non-experts should present five important qualities: 1. Trial-and-error learning environment for the user, 2. Large enough solution space, 3.User-friendly interface that requires little or no training, 4. Wide library of modules, and 5. Direct manufacturing without any alterations.[23] This research mainly focuses on the first three articles Von Hippel's list because of their problematic state compared to the advances in the remaining through computational manufacturing and BIM interfaces. In terms of the large library of modules, BIM interfaces creates enough support while open-source architecture continues to create a collaborative library of easily manufactured building systems and appliances. On the other hand, computational manufacturing gives the possibility to create customized products with a less or no penalty compared to the mass production facilities. On the contrary, non-expert design tools show certain limitations in terms of trial-and-error learning, appropriate solution space, and user-friendly interface.

Computational tools do not present satisfactory user interface models that encourage the user to learn through a trial-and-error process. Configurators provide a small amount of choice to non-experts. The extent of user freedom does not get far from small aesthetic decisions about the harmony of covering materials and economic decisions about the total coverage of the selected appliances. Conversely, drafting tools overwhelm the user with the level of freedom. Non-experts cannot be expected to generate a wide range of alternatives for an effective trial-and-error process. Generative tools seems to solve this problem by automating the generative process however, they go through this process in a closed fashion. User does not have much control or idea during the generation process, thus has a limited knowledge about the reasons behind the generated solution.

Non-expert design tools have redundant limitations over the solution space. A large

---

[22] Eric Von Hippel and Ralph Katz, "Shifting Innovation to Users via Toolkits," *Management Science* 48, no. 7 (2002): 9–13.

[23] Von Hippel and Katz, 9–13.

solution space is required for computational design tools in order to create the required level of customization for the user. Configurators provide small extents of solutions space. A calculation of the total possible combinations by configurators can bring a large number of alternatives, however the overall impact of such combinations are inferior. As an example, configurator approaches do not present the opportunity to configure the dimensions of the layout. On the other hand, the determinism available in the current generative tools puts serious limits on the solution space. The low interactivity in current non-expert generative tools requires the predetermination of many serious design decisions during the development process. A level of variation is still possible within this determinacy; however, it is not possible to expect novel solutions. As an example, Duarte's shape grammar can create a planned variety of houses according to the user's needs. However, the user knows that he/she will end up with an Alvaro Siza house from the start.

The mentioned tools also fall short in terms of providing a user-friendly interface to the user. As an example, the user should not need to go through an intense amount of training before using the tool. From another point, the tool should give constructive criticism upon the actions and decisions of the user to extend his/her vision. One other useful quality is the flexibility of the tool. The tool's interface should not be limiting and easy to configure without the need of a deep computational or architectural expertise. All type of computational tools gave certain problems in terms of the mentioned qualities for user-friendly interface.

Configurators work by a simplistic procedure that hardly needs any training. Nevertheless, this is mostly due to the limited choice available for the user. The type of support is closer to an online portfolio for the developer rather than a user-friendly design tool.

Drafting toolkits, on the other hand, present a user-friendly interface for drafting rather designing. The user can start drafting immediately with the help of modules in the library. However, developing a full alternative is going to take a large amount of time for a non-expert. Certain drafting toolkits utilize knowledge-based support for the users but the type of support presented by knowledge-based systems requires an expertise to understand and implement. The recommendations provided by such

systems can be multiple at times because of the large amount of conditional rules in their domain. This kind of decision making on such conflicting situations is a design expertise in itself. Another problem about knowledge-based systems is the difficulty in the development process. Adding new rules to the system requires a high level of architectural and computational expertise.

As for generative tools, the user interaction can be claimed as user-friendly. Explaining the requirements in a guided way and selecting from a list of generated alternatives is a simple and easy process. However, there is a certain point that brings generative tools and configurators closer. Generative tools offer user a wider range of control over the solutions and in a way, they generate novel solutions specific to user's requirements. Yet, the user is distant from the actual generation process because of the level of automation. Adding to that, the options presented to the user is already limited by the developer of the tool. The procedure to learn occupant's needs and the ways to provide for those needs is already decided by the developer. The process is similar to an interview session with a foreign architect that speaks so little in occupant's language. The occupant has no other choice than communicating the requirements in this limited and predetermined sense. The user also has little option to customize the working principle of generative tool. As an example, defining a new shape grammar requires a computational and architectural expertise beyond the level of non-expert.

In a creative process such as designing a house for the self, current computational tools present certain limitations. There is a need for new computational approaches that enhance user's control within the design process. In this way, non-expert users require user-friendly interfaces in order to start the design process immediately rather than going through an intense learning process. The provided design process should not limit the user with a small number of choices. Instead, the user needs to create the choices by exploring a large solution space. As a non-expert designer, the user needs a level of guidance to find better paths. However, this assistance should not be forced. Alternatively, the user needs the freedom to take other paths that can lead to dead ends. In such instances, the tool should provide ways to modify the initial problem structure, play with the available solutions, or interact with the guidance mechanism so that the user can act upon the mistakes immediately and

learn from them.

1.4 "House Generator", a New Computational Tool for Non-experts

This research targets the development of a computational approach to enhance non-expert designers' capability to generate alternative SFH layouts in a more time and effort efficient way. It presents House Generator (Ho-Gen), a novel genetic algorithm (GA) with an interactive interface that can generate 3D conceptual layout alternatives according to the user-defined geometric and topological criteria. Ho-Gen acknowledges non-expert designers' requirement for trial-and-error learning, appropriate solution space, and user-friendly interface thus, presents an interactive generative approach that is enhanced by genetic algorithms (GA).

Genetic algorithms (GA) are efficient and effective search methods that can work with large solution spaces. David Goldberg[24] highlights GA approaches as some of the most flexible, efficient, and robust algorithms in computational science. GA do not require deterministic hard coded rules to generate satisfactory solutions, instead they use the creative capabilities of evolution. According to a definition by Douglas Futuyma,[25] evolution is a blind process without predefined aims and objectives. As Futuyma, evolutionary mechanism depends on the mindless process of "natural selection".[26] Natural selection is a simple process which occurs by the replacement of less suitable organisms by organisms possessing certain genetic variations that enhance their reproduction and survival capabilities.[27] Evolution, despite of the simplicity of its mechanism, is the main driving force behind the vast variation in the natural environment. GA takes the advantage of evolution as a creative mechanism for the problems of various professions from engineering to architecture. GA

---

[24] David Edward Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning* (Boston [u.a.: Addison-Wesley, 2012), 2.

[25] Douglas Futuyma, *Evolution* (Sinauer, 2013), 282.

[26] Futuyma, 282.

[27] Futuyma, 282.

eliminates the need for providing hard coded and deterministic rules for generative systems, thus provides users an extended solution space to explore and achieve higher customization.

Ho-Gen acknowledges the initial complexity of design problems and the limitations in the design capabilities of future users. In this way, GA provide certain advantages. In his book, Davis argues that GA are very forgiving algorithms that can generate acceptable solutions despite the mistakes in their implementation and application.[28] Ho-Gen gives possibility to start with vague problem definitions and creates a chance for the user to intervene in the generative process to modify the problem definition. Therefore, Ho-Gen aims to support user's learning process through trial-and-error. The user can acquire emergent feedback to the changes in the problem definition and iterate towards better solutions.

1.5 Research Questions

The research is developed around the following main question:

What is the interactive computational model that can support occupants in the design process of their single-family house?

The main research question is divided and answered by the following sub-questions:

- What are the processes and potentials of automated generation and user interaction during the design exploration of single-family house?
- How can non-expert designers interact with computational tools in layout design exploration?
- What are the specific design requirements of single-family house layouts?

---

[28] Lawrence Davis, "Handbook of Genetic Algorithms," 1991; quoted in Peter J. Bentley and David W. Corne, "Introduction to Creative Evolutionary Systems," in *Creative Evolutionary Systems*, ed. Peter J. Bentley and David W. Corne (Morgan Kaufmann Publishers Inc., 2002), 8.

- What are the existing computational approaches to architectural layout design?

1.6 Research Outline

This research is organized in five chapters. Chapter 2 is dedicated to a Literature Review. The review starts by presenting single-family house and the changing dynamics in its production. Additionally, the review discusses the advantages of computational design tools for non-experts within user-centered mass customization models. The chapter also examines the models of computational support for non-expert designers in order to decide upon the degree of designer control and computational automation. It focuses on the general and architectural layout design process, and presents past computational approaches to the layout design process. Finally, a brief description on the history of genetic algorithms (GA) and an extended study on its computational implementation are given.

Chapter 3 is dedicated to Tool Development. It presents the overall procedure of Ho-Gen in both user interaction and evolutionary form generation processes. Representation part explains Ho-Gen's interpretation of a SFH layout with both the elements of SFH layout and their interrelations. User interaction part explains the communication process between the user and Ho-Gen during the initial problem definition through present inputs. Generation & Guidance part describes the genotype and phenotype representations for the layouts that are necessary for the genetic algorithm procedure. Various ways that the user can interact with the search process to provide direct user guidance are presented. Lastly, Evaluation part describes the mathematical model that evaluates layout solutions according to the user inputs. This part explains the hard and soft constraints and the way they integrate to the fitness function.

Chapter 4 presents Case Studies. This part aims to test Ho-Gen's performance in generating optimal layouts. Ho-Gen is tested in five case studies with two levels of complexity in terms of the number of layout elements and floors together with varying user requirements. This part presents the definitions of case study problems and the results of these studies in terms of the solutions generated by Ho-Gen and

fitness graphs.

Chapter 5 is the Conclusion part. This part discusses the contribution of this research in computational design in architecture. The thesis concludes with a brief discussion on the limitation of Ho-Gen in the conceptual design exploration and future work to develop Ho-Gen.

CHAPTER 2

1. LITERATURE REVIEW

2.1 Computational Tools for the Design of SFH

2.1.1 Single-Family House

A house is a residential building for people to live and meet their domestic needs. The domestic needs of humans are an essential feature in the definition of a house, which require a complex and multi-dimensional examination process. Marcus[29] defines these needs in the following hierarchy: Shelter, security, comfort, socializing & self-reflection, and aesthetic requirements. According to this hierarchy, a house needs to serve people both for their low-level requirements and high-level requirements. Low-level requirements start from the most general requirements such as shelter and security. On the other hand, higher-level requirements are personal and subjective. How well a house fulfills its purpose is related to its success in supplying low and high-level requirements.

A house, on the lowest level, is an essential necessity for survival, which serves as a shelter from environmental threats. The structure of this shelter should be stable enough to resist against physical loads. The outer skin should be well insulated to keep the indoor temperature within habitable levels. A house is also a secure place that borders between the occupants and outsiders. These lower-level needs, despite their importance, hardly define the purpose of house by themselves. Thus, they can also be supplied by simpler structures such as emergency shelters.

---

[29] Clare Cooper Marcus, *Easter Hill Village: Some Social Implications of Design* (Free Press, 1975).

House, apart from sheltering needs, is also a cultural issue. The connection between spaces of inhabitation and life-style is a result of owner's cultural and traditional expression.[30] These needs start to become evident after the third level of the hierarchy, comfort. Comfort is a vague term that can refer to environmental comfort as a technical aspect, or it can also relate to privacy from a socio-cultural perspective. The later steps of the hierarchy are higher-level needs. Occupants want to reflect their values and taste onto the spaces they inhabit. As such, a house becomes a medium of self-expression. Occupants use this medium to show their status and identity in the society. This reflection is not just for the social environment. Identity of a place is an essential factor to establish a sense of belonging with that space.

Single-family house (SFH) is a typology that is defined as a free-standing building that is occupied by a single family. SFH is an individual property, an empty space that is open to any collaborative interpretation of the occupants and the architect. A private house, as the unique representation of its user needs, a "home" that only serves for the occupants' individual will, lifestyle, desire, and taste. SFH, together with its value as a medium that represents its occupant, has been also used by architects as a way to manifest their architectural ideas and concepts.

2.1.2 From Mass Production to Mass Customization

20th century is a prominent time for manifestations by leading architects in the body of SFHs. Villa Savoye is a representative of Le Corbusier's "5 points of architecture" with its pilotis, roof garden, free plan, free facade, and horizontal windows.[31] Farnsworth House is another example where Mies van der Rohe creates a house in its simplest form only with two horizontal planes for ceiling and floor, eight slender

---

[30] Renee Y. Chow, "House Form and Choice," *Traditional Dwellings and Settlements Review*, no. 2 (1998): 51.

[31] Le Corbusier, *Towards a New Architecture* (London, Architectural Press [1946], 1946).

vertical supports, a whole glass plane on the boundary, and nearly no partition walls except for the core that bounds the wet spaces.[32] One other example is Robert Venturi's play on "signs" in Venturi House. The exterior of Venturi House calls the general image of a house with its gable roof, dramatically large chimney, and an arch over the main door.[33] Venturi emphasize these "signs" further by putting a large slit in the middle of the front façade, revealing that these elements do not serve for any structural function. In addition to the more personal ideals of individual architects, SFH is also used as a general manifestation for the use of mass production in the construction industry.

Mass production has played an important role in both the increase in number of SFH and its standardization. SFH is a preferred house type for families, but it is not feasible on the economic side because of the cost of land and construction expenses. Henry Ford's assembly line, which was initially used for automobile manufacture, introduced mass production techniques to the housing industry in the 20th century.[34] Mass production achieves high cost efficiency through the rapid manufacturing of standardized products within specialized factories. Tract house developments used mass production to construct mass housing sites that is made of the same two or three types of houses. It was also used by housing kit developments that provided the occupants a *do-it-yourself kit* to assemble the house themselves. This period resulted with a great increase in the quantity of SFHs, while also causing a downgrade on the quality of new houses by the high standardization it brought.[35]

---

[32] Werner Blaser, *Mies van Der Rohe: Farnsworth House-Weekend House*, 1 edition (Basel ; Boston: Birkhauser, 1999).

[33] Robert Venturi, *Complexity and Contradiction in Architecture*, The Museum of Modern Art Papers on Architecture (New York : Museum of Modern Art ; Boston : distributed by New York Graphic Society, 1977., 1977).

[34] Duarte, "Customizing Mass Housing."

[35] David Gartman, *From Autos to Architecture: Fordism and Architectural Aesthetics in the Twentieth Century* (New York: Princeton Architectural Press, 2010).

Figure 2-1. Levittown is an early mass produced SFH development in USA. (Retrieved from: http://www.newsday.com/long-island/nassau/levittown-history-in-photos-1.13458781#16, Accessed on 11.08.2017.)

The problem of standardization in the manufacturing industry has been slowly resolving with the rise of mass customization. Mass customization is a manufacturing model that aims to supply modified or completely original products for specific user needs with efficiency near mass production model. Mass customization is not a new production model, on the contrary, the term was originated in 1987 from the book Future Perfect by Stan Davis.[36] The manufacturing capabilities of 1980s, however, was not developed enough to answer such efficiency in resources and customization. Today with the advances in digital manufacturing techniques from CNC to 3D printing, it is possible to start the manufacturing process immediately without the need for a specific setup for every product. Nevertheless, certain challenges are still evident for mass customization. According to Zha and Lu,[37] an important challenge for the mass customization model is the decision process for the level of variety.[38] Zha and Lue argue that mass customization companies should supply minimum variety that satisfies an enough range of

[36] Stanley M. Davis, *Future perfect.* (Reading, Mass. [u.a.]: Addison-Wesley Publ. Co., 1987).

[37] Zha and Lu, "Knowledge Support for Customer-Based Design for Mass Customization."

[38] Zha and Lu, 407.

customer requirements in order to balance manufacturing expenses.[39]

2.1.3 Computational Tools for Mass Customization

An important challenge for mass customization is the process to gather, analyze, and implement the information about the requirements and preferences of the customers. Von Hippel and Katz[40] give a list of four methods for this process in product development such as market research techniques, lead user idea generation, configurators, and non-expert tools. First, companies can use market research techniques to gather the requirements and choices of many customers and use this information to design standard products that satisfy a public.[41] Second, companies can define lead users or the leading customer profiles in the marketing trends and acquire their design solutions to integrate them into standard products.[42] Third, companies can invite customers to configure their own products from a menu of predesigned options.[43] Last, companies can develop "user-friendly" computational tools that empower future users' non-expert design capabilities to let them create their own custom products.[44] Within this list of methods, computational tools that empower users provide certain advantages compared to other three methods.

First, non-expert tools provide the most user centric customization model as they put the users both in the head of analysis and synthesis processes. This is a significant advantage for the level of customization required for SFH. SFH is a typology that is occupied by a single family. SFH is an individual property, an empty space that is open to any interpretation by the changing lifestyles, values, and tastes. House, apart from sheltering and functional needs, is also a cultural issue. The connection

---

[39] Zha and Lu, 407.

[40] Von Hippel and Katz, "Shifting Innovation to Users via Toolkits," 16,17.

[41] Von Hippel and Katz, 17.

[42] Von Hippel and Katz, 17.

[43] Von Hippel and Katz, 16.

[44] Von Hippel and Katz, 16.

between the spaces of inhabitation and lifestyle is a result of owner's cultural and traditional expression.[45] Second, occupants' experience in the design process of their own house creates an additional value for the generated outcome. According to Belk, people embrace things s as a part of self when they invest their own efforts, time, and attention in their production.[46] The connection between the occupants and SFH is an important issue as SFH is usually a lifetime investment and the amount of time owners spend in their house. Third, customized production of houses prevents overproduction. The design of standard products from a general public opinion reduces the overall overproduction compared to mass production. Nevertheless, a risk of excessive production is still possible considering the custom domestic needs of the occupants. This research will proceed with the model that corresponds to the development of computational tools for non-expert designers because of the mentioned advantages.

2.2 Computational design tools and automation

On one hand, design is a natural ability that everyone possesses on a certain extent.[47] People develop necessary design skills to help in their daily tasks such as choosing outfits or organizing their personal space. Design is also a hobby activity for certain people. Occupants follow design magazines or websites that provide small tips or guides that anyone can follow to design a better living space. Naturally, such articles give occupants certain small points to consider or start with in the complexity of this design process. However, according to Hubert L. Dreyfus[48] the dependence on strict rules while performing any skill is a general indication of a novice's attitude. Single-

---

[45] R.Y CHOW, "House Form and Choice," *Traditional Dwellings and Settlements Review* 9, no. 2 (1998): 51.

[46] Russell W. Belk, "Possessions and the Extended Self," *Journal of Consumer Research* 15, no. 2 (1988): 144.

[47] Nigel Cross, *Designerly Ways of Knowing* (London : Springer-Verlag London Limited, 2006., 2006), 20.

[48] Hub+ert L. Dreyfus, "Intelligence Without Representation–Merleau-Ponty's Critique of Mental Representation the Relevance of Phenomenology to Scientific Explanation," *Phenomenology and the Cognitive Sciences* 1, no. 4 (2002): 367,368.

family house has a simpler functional program compared to institutional or office buildings. However, it is certainly a harder problem than daily design activities. In this way, non-expert designers require certain tools that amplify their design abilities. In this context, it is essential to decide on the ways and the extent of collaboration with computational tools.

The definition of the extent of support that can be provided by computational tools is an important decision. This decision is a determinant factor in non-expert's level of collaboration within the design process. In the most labor-intensive way for the occupant, computational design tools can be expected to provide a fully automated process. This kind of scenario can be thought as working with an automated architect that generates a single-family house form after learning occupant's special requirements and requests. Today, this level of automation is actually a very popular concept for many occupations.

2.2.1 Automation and Occupations

The replacement of human workforce by computers is a popular subject in many online articles. A report by Gallup in 2017, as cited in McGrady's article,[49] asserts that 37% of Millennials face the threat of being replaced by automation in their workplace. According to another study by McKinsey Global Institute, as cited by 2017 article by Whitehouse, Rojanasakul and Sam,[50] today's technological capabilities can fully automate only the five percent of whole occupations, however, it is possible to automate a third of the total workload within the sixty percent of the occupations. Such high numbers in these statistics can bring questions regarding the

[49] Andrew Dugan and Bailey Nelson, "3 Trends That Will Disrupt Your Workplace Forever," Gallup.com, June 8, 2017, http://news.gallup.com/businessjournal/211799/trends-disrupt-workplace-forever.aspx; quoted, in Vanessa McGrady, "New Study: Artificial Intelligence Is Coming For Your Job, Millennials," Forbes, June 9, 2017, https://www.forbes.com/sites/vanessamcgrady/2017/06/09/millennial-jobs/.

[50] James Manyika et al., "Harnessing Automation for a Future That Works," *New York: McKinsey Global Institute*, 2017; quoted in Mark Whitehouse, Mira Rojanasakul, and Cedric Sam, "Is Your Job About To Disappear?: Quicktake," *Bloomberg.Com*, June 22, 2017, https://www.bloomberg.com/graphics/2017-jobs-automation-risk/.

possibility of computers outperforming humans.

Indeed, computers already proved their success against humans in certain instances such as games. Such an event occurred in 1997, when World chess champion Garry Kasparov lost a chess match against Deep Blue II.[51] According to Campbell, Hoane, and Hsu,[52] Deep Blue II is the successor hardware and software model of IBM's research on developing machines that can play chess. As Campbell et al., Deep Blue II played chess by ranking possible moves according to a mathematical function that measures the advantage of positions, but did it in a very fast way that reached up to 126 million moves in the game against Kasparov.[53] Naturally, Deep Blue II's chess strategy is very different from a human player. In the worst-case scenario, Deep Blue II has to measure every possible move for the position which exceeds the capabilities of humans. Making a nearly complex calculation for nearly every possible move while keeping every result in mind to rank and compare is not possible for human cognition. In this way, Deep Blue II took the advantage of higher data storing and data processing capabilities of computation.

However, this was also not the peak level in the capabilities of computers against humans in games. More recently, a computational model by Google, AlphaGO, won a match against world's number one GO player Ke Jie.[54] GO is a very different game than chess in certain ways which requires different computational strategies compared to Deep Blue II's working principle. One study by Burmeister and Wiles,[55] examines the differences between GO and chess. According to this study, GO is a

---

[51] Murray Campbell, A. Joseph Hoane, and Feng-hsiung Hsu, "Deep Blue," *Artificial Intelligence* 134, no. 1 (January 1, 2002): 57, https://doi.org/10.1016/S0004-3702(01)00129-1.

[52] Campbell, Hoane, and Hsu, 58.

[53] Campbell, Hoane, and Hsu, 60.

[54] Agence France-Presse, "World's Best Go Player Flummoxed by Google's 'Godlike' AlphaGo AI," *The Guardian*, May 23, 2017, sec. Technology, http://www.theguardian.com/technology/2017/may/23/alphago-google-ai-beats-ke-jie-china-go.

[55] Jay Burmeister and Janet Wiles, "The Challenge of Go as a Domain for AI Research: A Comparison between Go and Chess," in *Intelligent Information Systems, 1995. ANZIIS-95. Proceedings of the Third Australian and New Zealand Conference On* (IEEE, 1995), 181–186.

more open game in terms of the number of possible moves that can be played and positions in a GO game resist to mathematical ways of evaluation.[56] In this way, Deep Blue's strategy of evaluating a high number of possible moves does not work for a GO game. However, Google overcome this problem by taking another direction. An article by two members of AlphaGO's developer team, Silver and Hassabis,[57] explains that AlphaGO designate its own rules through learning from real matches against real players instead of depending on predefined rules. Such advancements in the development of intelligent machines can form a basis for using computers for automating occupations.

Yet, game playing no matter its complexity is just a single task compared to the multifarious variety of duties within occupations. A machine developed for chess can win games against chess masters; however, it lacks any other capability beyond its programming. The difficulty of developing machines that can replace human workforce also varies within different occupations. The previously mentioned news article by Whitehouse, Rojanasakul and Sam[58] shares an interactive graph based on the statistics provided of U.S. Bureau of Labor Statistics[59] that compares the expected automation rate of occupations to their annual earnings. According to the graph, various occupations face a high risk of automation such as accountants with a rate of %94 or taxi drivers with %89, while certain occupations are on a safer end such as architects with 1.8% or graphic designers with %8.2.[60] To understand the reasons behind this degree of difference in the automation expectancies, it is necessary to explore a field of study with a long-standing background, namely artificial intelligence (AI).

---

[56] Burmeister and Wiles, 182–84.

[57] David Silver and Demis Hassabis, "AlphaGo: Mastering the Ancient Game of Go with Machine Learning," *Research Blog* (blog), January 27, 2016, https://research.googleblog.com/2016/01/alphago-mastering-ancient-game-of-go.html.

[58] Manyika et al., "Harnessing Automation for a Future That Works"; quoted in Whitehouse, Rojanasakul, and Sam, "Is Your Job About To Disappear?: Quicktake."

[59] Carl Benedikt Frey and Michael A. Osborne, "The Future of Employment: How Susceptible Are Jobs to Computerisation?," *Technological Forecasting and Social Change* 114 (2017): 254–280.

[60] Whitehouse, Rojanasakul, and Sam, "Is Your Job About To Disappear?: Quicktake."

Stuart Russell and Peter Norvig,[61] in their book that presents AI in the very introductory and undergraduate level, defines AI as the study of building intelligent entities that can act or think in humanly or rational ways. While the term "humanly" is more relatable and understandable despite of the complex neurological mechanisms behind the human acts and thoughts, the term "rational" brings questions beyond this definition. According to Russell and Norvig,[62] rationality is an ideal measure that is constructed by the limited knowledge of the entity for assessing the performance of doing the "right" thing in a certain context. A calculator is a very basic example for a rational artificial intelligence despite its outmoded level of intelligence compared to the novel AI capabilities. The most basic calculator is capable of making four arithmetical operations, but it always returns the right result no matter the complexity of given operations. In a way, calculators have no other option than giving the correct result as their actions are strongly limited by an exact and strict language of mathematics.

Similarly, occupations that are governed by strict rules and procedures show higher expectancies on automation. The state of accountants can be a good example in this sense. Various tasks involved in the practice of accounting are defined by Merriam-Webster[63] dictionary as "…recording and summarizing business and financial transactions and analyzing, verifying, and reporting the results". Despite the multiplicity of tasks, accounting practice is governed by strict standardizations and principles imposed by such organizations that vary between different countries. For example, accounting practice in U.S. is governed by Generally Accepted Accounting Principles (GAAP) which is a collection of concepts, objectives, standards, and conventions that guides the presentation and preparation process of financial

---

[61] Stuart J. Russell and Peter Norvig, *Artificial Intelligence : A Modern Approach*, Prentice Hall Series in Artificial Intelligence (Englewood Cliffs, N.J. : Prentice Hall, 2010., 1995), 1,2.

[62] Russell and Norvig, 1.

[63] "Accounting," *Merriam-Webster*, accessed January 14, 2018, https://www.merriam-webster.com/dictionary/accounting.

statements.[64] Another task of such organizations, however, is to find ways to reduce the complexity of these standards.[65] A study by Donelson, McInnis, and Mergenthaler,[66] mentions a common criticism towards GAAP because of its dependence on rules in excessive detail. Apparently, human accountants are having difficulties in following GAAP rule sets for standard procedures on common tasks. On the other hand, a rational AI model should not face with much trouble while following a large database of rules mentioning the "right" thing to do under certain conditions. Similar to Deep Blue, an accounting AI can surpass humans in the speed and accuracy for checking rule sets because of their matchless data processing and storing power.

The popularity of artificial intelligence approaches are not limited to procedurally governed occupations. The studies on the development of self-driving cars notably by Waymo[67] or Tesla[68] have come into such an attention level that caused certain discussions[69] about the replacement of public transportation by driverless technologies. In a general look, automation can provide better drivers as they take out the potential of human error in the traffic. A machine can prevent distraction-based incidents because they do not exhaust unlike humans do. Alternatively, machines can observe farther or see in better detail with the help of digital cameras or censors. Adding to that, machines are already better in locating addresses and

---

[64] "About GAAP," accessed January 14, 2018, http://www.accountingfoundation.org/cs/ContentServer?c=Page&cid=1176164538898&d=&pagename=Foundation%2FPage%2FFAFBridgePage.

[65] "Simplifying and Improving GAAP," accessed January 14, 2018, http://www.accountingfoundation.org/jsp/Foundation/Page/FAFBridgePage&cid=1176164540272.

[66] Dain C. Donelson, John McInnis, and Richard D. Mergenthaler, "Explaining Rules-Based Characteristics in US GAAP: Theories and Evidence," *Journal of Accounting Research* 54, no. 3 (2016): 827.

[67] "A New Way Forward for Mobility," Waymo, accessed January 12, 2018, https://waymo.com/redirect/.

[68] "Autopilot," accessed January 12, 2018, https://www.tesla.com/autopilot.

[69] "Forget Self-Driving Cars. Automated Public Transportation Is Coming," Roadshow, accessed January 12, 2018, https://www.cnet.com/roadshow/news/self-driving-cars-automated-public-transport-bus/; "Forget Cars, Self-Driving Shuttles Are the Future of Transportation," WIRED, accessed January 12, 2018, https://www.wired.com/story/las-vegas-shuttle-crash-self-driving-autonomous/.

finding routes because of the novel capabilities of GPS technology. On the other hand, driving is a highly intuitive task that is not governed by strict rules unlike accounting. As Russell and Norvig,[70] developing an automated taxi driver is a complex open-ended problem with countless events to consider because of the combinational possibilities.

As an example, a self-driving car should be able get through a complex process of making multiple observations and decisions while taking a left on a crowded crossroad. It has to recognize its destination, direction of road lines, traffic signs, and vehicles within the traffic. AI also has make certain calculations regarding the travelling speed of multiple vehicles, and understand their upcoming moves from certain signs. In many ways, traffic is a complex and unpredictable environment to develop a definitive list for every possible scenario. A vehicle can violate the laws by skipping the signal or a driver can accidentally make a move than changes mind. Also modeling a decision system moves is not convenient considering the combinations of vehicles and their possible moves. Instead, self-driving cars follow the procedure of AlphaGO to develop their own rules by machine learning algorithms.

According to a definition provided by Ethem Alpaydın,[71] "Machine learning is programming computers to optimize a performance criterion using example data or past experience". As Alpaydın, machine learning can be used to develop intelligent machines for real world tasks that humans cannot define in systematic instructions because of the tasks' realization in an "unconscious" manner.[72] Waymo, in order to provide real world experience for their self-driving cars, has fabricated a city in US to conducts tests.[73] Although developing a database of rules for self-driving cars is

---

[70] Russell and Norvig, *Artificial Intelligence*, 41.

[71] Ethem Alpaydin, *Introduction to Machine Learning*, 2nd ed. (The MIT Press, 2010), 3.

[72] Alpaydin, 3.

[73] Alexis C. Madrigal, "Inside Waymo's Secret World for Training Self-Driving Cars," *The Atlantic*, August 23, 2017, https://www.theatlantic.com/technology/archive/2017/08/inside-waymos-secret-testing-and-simulation-facilities/537648/.

not possible because of the involved level of intuition, the tasks' performance criterion is rather simple. A rational act for a self-driving car, in the most general sense, is to drive safely between two destinations without causing collusions. This task can be simulated by real world or virtual applications for the self-driving car to gain experience.

On the other hand, design related occupations such as architecture have a seriously low automation expectancy compared to accountants and drivers. Computers, however, are becoming increasingly essential for architectural practice. Yet, they serve more as tools for architects rather than being fully autonomous design machines. Despite of artificial intelligence's success on working with complex rule sets or learning intuitive tasks, full automation of architectural design is still out of question. Certain problems of architectural design are shared among accounting and driving as well. Architects also consider a large amount of governmental and technical engineering oriented standards. Additionally, architecture is also considered as an intuitive task. However, certain differences within the design practice keeps AI models away from full automation. Next section examines these relevant problems in architectural design.

2.2.2 Automation and Design

Nigel Cross[74] traces the origin of computation in design back to the "design methods movement" in 1960s that aimed to develop a rational and objective design processes against the resultant problems of Second World War. As Bayazit,[75] researchers involved in design methods were searching for "rational methods" to integrate "scientific techniques and knowledge" to develop a rational base for design decisions. For example, Christopher Alexander[76] developed a rational and systematic

---

[74] Cross, *Designerly Ways of Knowing*, 95.

[75] Nigan Bayazit, "Investigating Design: A Review of Forty Years of Design Research," *Design Issues* 20, no. 1 (January 1, 2004): 19, https://doi.org/10.1162/074793604772933739.

[76] Christopher Alexander, *Notes on the Synthesis of Form*, vol. 5 (Harvard University Press, 1964), 84.

approach to design based on an analysis-synthesis process. For Alexander, design tasks are complex problems that require a mathematical and logical analysis process to disintegrate into smaller parts[77]. After such an analysis process that brings the problem in a manageable simplicity, a designer requires to pass on the synthesis process where he/she solves disintegrated parts.[78]

Yet, design methods movement faced a deep criticism in 1970s in terms of their overly systematic approach.[79] A new base of research provided alternative definitions for design problems in order to differentiate such problems fully scientific and systematic problems. In this way, Rittel and Webber[80] defined design problems by the term "wicked". In the same article, Rittel and Weber give a list of ten properties that differentiates wicked problems from other "tame" ones.[81]According to three items within this list, wicked problems lack an exact formulation, they have no stopping rule, and they cannot be tested with an immediate or ultimate method.[82]

The unavailability of exact definitions of design problems have been stated by multiple researchers. According to Bryan Lawson,[83] design problems unlike puzzles or mathematical operations are in absence of clear objectives as well as apparent difficulties inherent in the process of realizing these objectives. Cross[84] views design problems as a variant of "ill-defined problems". An ill-defined problem, for Cross,[85]

---

[77] Alexander, 5:84.

[78] Alexander, 5:84.

[79] Cross, *Designerly Ways of Knowing*, 96.

[80] Horst WJ Rittel and Melvin M. Webber, "Dilemmas in a General Theory of Planning," *Policy Sciences* 4, no. 2 (1973): 155–69.

[81] Rittel and Webber, 160.

[82] Rittel and Webber, 161–64.

[83] Bryan Lawson, *How Designers Think : The Design Process Demystified* (Oxford ; Burlington, MA : Elsevier/Architectural, 2006., 2006), 56.

[84] Nigel Cross, "Design Cognition: Results from Protocol and Other Empirical Studies of Design Activity," in *Design Knowing and Learning: Cognition in Design Education.*, ed. Charles M. Eastman et al. (Oxford, England: Elsevier Science Ltd, 2001), 3, https://doi.org/10.1016/B978-008043868-9/50005-X.

[85] Cross, 3.

is a problem with only an approximate definition that corresponds to a vague information about the objectives and limitations. This obscurity within the architectural design problems can be examined within the openness, subjectivity, and multiplicity of its objectives.

Architectural design, up to a certain extent, is bounded by strict rules and standardizations. A certain example is building and zoning legislations provided by higher authorities. Zoning regulations limit the layout of the building with setback boundaries, or they constraint the total area of the building by the floor area ratio (FAR). Another example is the functional requirements related to the building program. The designed space should be suitable activities that are planned to take place. This suitability is directly related to such points as the number of people that space serve or the appliances and equipment required for such activities. These kinds of rules are generally nonnegotiable and required to be met at all costs.

On the other hand, the designed space does not emerge solely from such hard rules and standards. Otherwise, there will not be any reason to call design activity as a routine problem solving process with well-defined problems and a series of clear directions on solution. Design problems, however, require another kind of process rather than mere problem solving because of their initial vague definition different from other kinds of problems. Design is a creative process with open definitions that presents a level of freedom for the designer. Kees Dorst[86] defines this "openness" with the levels of "underdetermination" that is available in design problems.

According to Dorst,[87] a large part of design problems is "underdetermined". The clarification of such problems and the selection of suitable design solutions emerge together after a multiplicity of proposals by the designer.[88] Designing a house with

---

[86] Kees Dorst, "The Problem of Design Problems," in *Expertise in Design* (Design Thinking Research Symposium 6, Sydney, Australia, 2003), 136.

[87] Dorst, 137.

[88] Dorst, 137.

low-energy requirements, supposedly, is a determinant design objective which can be evaluated through technical calculations. However, this objective does not directly correspond to a fixed number that limits the energy considerations. This objective value varies in terms of different design problems. The experts can make initial predictions by calculations that consider certain standard values. Architectural design, however, also includes certain issues that resist objective estimations in advance. The dimensional requirements for certain spaces in the house do not necessarily correspond to exact numeric definitions. The occupants can use such descriptions as “cozy”, “functional”, or “comfortable” to define their expectations from a certain space.

As Dorst, certain parts of the design problems go beyond this underdetermination. Design problems are also partially “undetermined” with an amount of space to the purely subjective intentions of the designer.[89] Higher-level domestic needs such as aesthetics cannot be modeled in a pre-descriptive way. There is no evident scientific fact that supports the beauty of a house over another one. In these cases, architects are free within the limits of their communication skills to persuade the occupant in the aesthetics of their house.

In addition to the various levels of openness inherent in design objectives, these objectives are also endless in numbers. Architectural design is related to a vast amount of interrelated and multidisciplinary objectives. These objectives may range from spatial solutions for the required functional program, socio-cultural context, and economical boundaries to technical aspects like natural and mechanical lighting, thermal conditions, fluid dynamics, structure, and acoustics. An architect is not necessarily an expert in all these fields. However, they are still required to have a general knowledge in these areas to establish the necessary communication between the involved disciplines.

---

[89] Dorst, 137.

The objectives of architectural design are rarely separate from each other. These requirements are interrelated in the sense that working on one objective can give way to problems on other objectives. During the design process, the occupants can ask for design revisions. This change in the programmatic objective causes changes in other objectives. For instance, a larger room brings larger loads on the structure, which in return can cause an increase in the dimension of structural elements. This increase in the area also changes the cooling or heating requirements of the house which again lead to changes in the mechanical installments to provide enough capacity.

Alexander's rational and systematic approach has strong relations with the multiplicity and the interdependence between the objectives. Alexander's analysis method on design problems requires enlisting all the requirements related to the problem, then establishing an organization in this list through exploring interactions between these relations.[90] Alexander also developed a mathematical method that structures the group of objectives into clusters that work together by utilizing statistical and mathematical functions.[91]In a way, Alexander's approach is a divide-and-conquer model that creates a set of meaningful and operable objectives from a whole and complex one.

Bryan Lawson[92] is critical towards the design analysis method of Alexander. Lawson asserts that Alexander's method treats every problem equally in the structuring process which contrasts with general actions of a designer.[93] According to Lawson, "Alexander fails to appreciate that some requirements and interactions have much more profound implications for the form of the solution than do others".[94] Such conflicting situations may require the designer to decide upon the relative importance of conflicting goals one upon the other. Domestic needs also have a

---

[90] Alexander, *Notes on the Synthesis of Form*, 5:93.

[91] Alexander, 5:174–91.

[92] Lawson, *How Designers Think*, 76,77.

[93] Lawson, 77.

[94] Lawson, 77.

hierarchy. It is usually simpler to decide about goals in different levels of hierarchy such as the structural safety over aesthetic intentions. In other times, these decisions also require the subjective interpretation of the architect to establish the requirements in a priority. These priorities, similar to the objectives, can also change during the design process.

Occupants can initially ask for a single-story house because of various reasons such as age and disabilities. The overall program requirements, however, can exceed the total area of setback boundaries. Pushing this objective would either require a decrease in the total area or another site for the house. The architect, instead, can convince the occupants in the advantage of a second story and the possibility of using a small elevator. Similarly, the economic constraints on the design process can be seen as a nonnegotiable at first. Related calculations, however, are usually made with very simplistic terms within the design process so they lack the precision. An architect can propose solutions that exceed the economic means, but they can convince the occupant about the increase in quality of their home by these exceeding amounts. In these terms, economic boundaries become also a negotiable constraint that can be evaluated in terms of the other advantages of the solution. Such a level of flexibility inherent in the design process presents designers a high multiplicity of solution ways to take which results with a large number of solutions to consider.

2.3 Computational design tools and user interaction

Development of computational tools for the complete automation of architectural design process is problematic. Beyond the rationally incomputable objectives of architectural design such as designing aesthetic buildings or designing for psychological comfort, a rational definition for an architectural design problem is not existent as well. Indeed, such a process is a part of the creativity inherent in architectural design. Architects mostly define design problems in their own subjective way rather than applying a deep analysis on the problematic design context. Yet, computation's incapability to provide a fully automated design process does not necessarily keep them away from architecture. On the contrary, the collaboration between computation and architects are more apparent than ever. Today, computers are effective partners for architectural design. Computer-aided

drafting (CAD) tools provide a precise and quick interface for data-intensive architectural drawings. Rendering engines enhances the presentation capabilities by photorealistic design representations. Simulation software provides complex calculations on the environmental or structural performance of buildings. BIM tools augment the collaboration of the vast multiplicity of professions involved in the design process. Naturally, non-expert designers require a broader support than practicing architects. A simulation on the thermal performance of a building may not be the most vital aid for a non-expert as they miss the very fundamentals of architectural design.

Von Hippel and Katz, in a general guideline for the development of tools that support non-expert users, provided a list of five objectives: 1. Trial-and-error learning environment for the user, 2. appropriate solution space, 3.User-friendly interface, 4. Wide library of modules, and 5. direct manufacturing without any alterations.[95] This research is limited with the first three objectives in Von Hippel and Katz's list. The development of open source architecture portals and the prevalence of BIM methods provide the necessary library for the designers. On the other hand, digital manufacturing technologies are continuing to develop rapidly and becoming personalized with their easy access. This easy access, in a way, gives everyone the support to be a manufacturer. The remaining elements correspond to broad definitions without the actual product to be designed. In this way trial-and-error learning, appropriate solution space, and user-friendly interface will be examined together with architectural design and single-family house.

2.3.1 User-friendly interface and trial-and-error learning

This section will examine studies on designer behavior in order to define the importance of trial-and-error learning and reach the requirements for a user-friendly interface. A main beneficial outcome of design methods movement's aim in

[95] Von Hippel and Katz, "Shifting Innovation to Users via Toolkits," 9–13.

integrating rational and objective methods to design process is the enhancing body of knowledge on designers' attitude and action within complex design situations. Nigel Cross[96] unifies this related body of research under the name "science of design". Cross defines "science of design" as "that body of work which attempts to improve our understanding of design through 'scientific' (i.e., systematic, reliable) methods of investigation".[97] According to Cross,[98] the studies on designing were realized in many ways from academic reflections on designer's self-reports to experiment based methods such as protocol studies. Such studies on designer behavior are not expected to give assistance in creating a complete rational procedure to tackle design problems because of the high variety in design problems and the innate subjectivity involved in the design process. However, this body of knowledge can give an insight about the general character of design actions which can help in the development of similar purpose tools to aid non-expert designers.

As a contrast to the deep analysis methods inherent in highly systematic and objective models for design, designers take another way in the initial stages of this process. Bryan Lawson[99], after a protocol study that targeted to reveal the behavioral differences in problem solving between architecture and science students, observed that architectural students generally showed a solution-focused strategy compared to the problem-focused strategy taken by science students. As Lawson,[100] this solution-focused strategy is an indicator or the synthesis based analysis methods of designers. Lawson notes that in the obscure context of design where problems are away from being obvious, designers find their problems through making certain moves such as using primary generators.[101] A primary generator, as Lawson, is a concept developed by Jean Darke that corresponds to a general solution concept or a limited definition

---

[96] Cross, *Designerly Ways of Knowing*, 99.

[97] Cross, 99.

[98] Cross, 17.

[99] Lawson, *How Designers Think*, 42.

[100] Lawson, 44.

[101] Lawson, 56,295.

of the design problem.[102]

Donald Schön, in another protocol which studies an architectural design review between a studio master and a student, exemplifies his theory of "reflection-in-action".[103] In his study, Schön differentiates the design actions of the novice and expert designer involved in the study by novice designer's halt in the failure of her design idea versus expert's constant struggle with the problem context through design moves.[104] As Schön, these design moves are the means of communication for the expert designer to put a conversation with the problem context which in some cases, stimulate the design problem to talk back.[105] In these certain instances, the expert discovers new things about the problem and shifts the position to consider new moves on the context.[106] Schön asserts this set of procedure is a general process for experts in ill-defined and vague problem contexts, no matter the difference in their design moves or shifting positions.[107]

According to these protocol studies, trial-and-error learning is an essential part of the design process. The ill-defined nature of design problems resists to deep analysis methods. In this way, trial-and-error learning starts within the very early and vague stages of the design process. The subjective decisions on the definition of open-ended parts and the relative importance of design objectives are taken through a trial-and-error learning process. This early process includes the development of a large amount of concepts and alternative solutions. Non-experts, on the other hand, lacks the required design experience and education to utilize this process as well as an expert architect.

---

[102] Jane Darke, "The Primary Generator and the Design Process," *Design Studies* 1, no. 1 (1979): 36–44; cited in Lawson, *How Designers Think*, 46,47.

[103] Donald A. Schön, *The Reflective Practitioner : How Professionals Think in Action /* (New York : Basic Books, c1983.), 102–4.

[104] Schön, 102.

[105] Schön, 94.

[106] Schön, 94,95.

[107] Schön, 103.

The ill-defined nature of design problems is just one difficulty that designers come across during the process. According to Lawson, the inability to define a design problem in a complete and exact manner causes the designers to work with an "inexhaustible" list of solutions.[108] Even an initial interpretation of the problem during the design process can bring large amounts of solution candidates. As it can be seen from the previous examples, the constraints of the design problems are not overly strict. In this way, the designer can also ignore these limitations based on his/her persuasion skills on the clients or with the presentation of better design solutions beyond the limitations. Such level of flexibility causes the designer to work with a high number of solutions.

For designing, even simpler problems have a large solutions space. As an example, the very basic houses that only served for the sheltering needs of the society created a vast amount of residential forms in the past. Sheltering needs, as mentioned previously in Marcus'[109] hierarchy, stays within the most general domestic requirements of people. Sheltering needs can be associated with less subjectivity in terms of the decisions of the occupants because the general threat is more or less the same. On the other hand, material possibilities within the environment pushed people to build in different ways and forms which resulted with the diverse range of stylistic variety in houses.

During the design process, architects work with very simple representational means that are away from the realism of the building's final form.[110] These representations include simple bubble diagrams and conceptual mass models to save time for creating more alternative solutions. As Liu, Chakrabarti, and Blight[111] exploring the

---

[108] Lawson, *How Designers Think*, 121.

[109] Marcus, *Easter Hill Village*.

[110] Gabriela Goldschmidt, "The Dialectics of Sketching," *Creativity Research Journal* 4, no. 2 (January 1, 1991): 123–43, https://doi.org/10.1080/10400419109534381.

[111] Y.-C. Liu, A. Chakrabarti, and T. Bligh, "Towards an 'Ideal'Approach for Concept Generation," *Design Studies* 24, no. 4 (2003): 341–355.

widest range of design options is a crucial part of conceptual design process to grow better design concepts. To assess the quality of a large range of concepts, designers utilize divergent thinking together with convergent thinking. Divergent thinking expands the range of design exploration to locate alternative solutions to the problem; on the contrary, convergent thinking shrinks the range of possibilities by focusing on better solutions.[112]

Designers are required to find a balance between the divergent and convergent thinking in the design process. Cross,[113] associates convergence dominated design processes with novice behavior. The design process can be limited to an initial good performing concept to keep things very simple. However, designers work with complex problems, in this sense one-shot operations hardly generate satisfactory solutions. A divergence dominated design process, on the other hand, can fail to meet the detailed examination required for meeting long list of design related objectives in time.

Despite the unavailability of a fully automated computational method to support non-experts in the design process, computational tools can still provide support in these underperforming abilities of non-expert designers. In this way, a user-friendly interface should enable non-experts to develop and explore a wide range of design solutions with their inherent design abilities. Such a tool should be forgiving in terms of the generation of high quality solutions opposed to the occupant's vague or false definitions on the design problem. Accordingly, the computational tool should be able to look over the layout design problem from a wider perspective, including the multiplicity of objectives as a whole. Additionally, this wider perspective should also be flexible and open to user's interpretation during the generation process because of the openness and subjectivity involved in design problems and objectives.

---

[112] Liu, Chakrabarti, and Bligh.

[113] Cross, *Designerly Ways of Knowing*.

2.3.2 Appropriate solution space

The solution space of a computational design tool corresponds to the extent of alternative solutions that the design tool can generate. The size of this solution space is an essential factor for controlling the level of customization allowed for the user. A very small solution space can present a scenario worse than the standard solutions inherent in housing industry; on the other hand, a very large solution space can lower the effect of computational exploration and bring inadequate solutions. In this way, the solution space for a computational design tool should be just right. According to Cagan, Campbell, Finger, and Tomiyama[114] the range of solution candidates is directly related to the "representation" of computational model. As Cagan et al.[115], two important considerations for the development of a representation is the definition of building blocks and their interrelations. Building blocks are smaller customizable parts of the generated product. For example, a house can both be represented as a complex collection of bricks or a simpler combination of wall and roof systems. The determination of the building blocks for a SFH and therefore an "appropriate" solution space requires a deeper look in the functions and parts associated with a SFH.

Functional analysis of SFH

A house should function as a space that supplies occupants' low and high-level domestic needs, as discussed in the previous section. It is possible to define these needs with their corresponding domestic activities. Several domestic activities are generally available in every house setting as they are closely associated with the above-mentioned low-level needs. These domestic activities are numerous, such as recreation, sleeping, eating, cooking, socializing etc. Furthermore, the activities within a house can vary according to occupants' lifestyle. As an example, home

[114] J Cagan et al., "A Framework for Computational Design Synthesis: Model and Applications," *JOURNAL OF COMPUTING AND INFORMATION SCIENCE IN ENGINEERING* 5, no. 3 (September 2005): 172.

[115] Cagan et al., 172.

offices can partially act as professional working environments, while also serving for domestic activities.

In a way, a house should function as a comfortable space that can provide for the related activities. As Habraken[116] states, when a space is given a certain function, it should be designed with the capacity that it can allocate a configuration of objects associated with the related function. In this way, even the general activities can require different spatial solutions depending on the occupant behavior. The eating activity of a family of four will certainly require a larger table and more seats than a family of two. Cultural interpretations of the activities can also have a role in the required capacity such as the on-the-floor dining arrangement of the Japanese culture.

Apart from spaces with well-defined uses, a space within the house does not necessarily correspond to just one activity. For instance, bedrooms are associated with sleeping. On the other hand, bedroom can also stand for a place that serves for the need of resting and intimacy. The occupant can require an additional living quarter of their own within their bedroom. Children's rooms are very multi-functional in this sense. Children use their rooms as a place for play and study in addition to sleeping.

There may also be spaces within a house that does not correspond to an activity in a general way. In this sense, the mere purpose of circulation spaces is to connect different areas within a home. Halls and corridors provide horizontal circulation, while stairs and elevators provide vertical circulation for multi-floor houses. The lack of an overall activity can bring the issues about circulation space as a dull and soulless spaces, however it is still possible to make circulation spaces contribute more to the living in house with a few interpretations.

---

[116] N. John Habraken, "The Control of Complexity," *Places* 4, no. 2 (1987).

The function of a house directly depends on the occupant. The scale of certainty of the spaces within a house can change from project to project. It is possible that the architect does not communicate directly with the occupants. The client can also be a contractor company that wants to build a housing site rather than a customized house. In that sense, the activities of the house cannot be well-defined as in working with the real occupant. Alternatively, the occupant can require an adaptable house that can change according to changes in the family's lifestyle overtime. There are certain ways to answer these circumstances such as open-plan houses that do not divide spaces with hard installations rather the user can bring transitory divisions over time. Tadao Ando's Walless House is such an example where the architect proposed movable furniture to make the space adaptable. This study addresses a case when an architect works on a customized SFH with static spatial requirements.

The functional analysis of SFH, as a customized design product, requires a meaningful communication with the occupants. Information needs to be gathered regarding the number of rooms and their sizes such as the size of the family, the type of fixed installations on the wet spaces, the conditions for overnight guests, or the type of outdoor spaces etc. The designer and the occupant define a general building program through the briefing sessions which stands for the number of required spaces, their sizes, and the relationship in between them.

A house cannot be reduced directly to the sum of the separate spaces for different domestic activities. Hillier[117] states that human space is more than the properties of the individual spaces; instead, the configurational aspect of space with the relations between many spaces makes the space a whole. The relationship between the public and private spaces, as an example, is an important concern of the configurational aspect of a house. The availability of sleeping quarters alone does not necessarily provide privacy alone in itself without the careful placement of this quarter within the house. Bedrooms are generally separated from the more public living quarters

[117] Bill Hillier, "The Art of Place and the Science of Space," *World Architecture* 185 (2005): 96–102.

both physically and visually either by a circulation space or a floor difference. This configurational aspect includes the neighborhood relations between the spaces, zoning of certain activities, and the orientation of the spaces within the overall layout.

Layout design in general

Configuration design is a generic activity that is shared generally by all the design related professions. Configuration design refers to the assembly of a predefined set of components into a meaningful and purposeful whole that satisfies certain predefined conditions.[118] This definition provides an overall unifying explanation for very distinct design problems such as a production facility, a computer chip, a website, or a house. These configuration problems, despite of the difference in their scale, require allocating various predefined elements in a limited exterior boundary.

Configuration design can be tackled with a bottom-up approach, as it gathers the whole from the part. The designer starts from the very basic or the smallest available elements and produces the layout in different levels of hierarchy. This is similar to starting the design of a house from considering the arrangement of furnishing, circulation, and related activities within a room. The form of rooms is the result of the all furnishing arrangements and the space of activities around them. This process proceeds into the configuration of rooms within the house after the definition of room forms. Rooms are arranged with one another according to the flow of movement between them and the client preferences on their proximity. This process can go on further to the arrangement of resultant houses within a site boundary if the designer is building a neighborhood. This configurational design process refers to building layout design in architecture. Architects conduct layout design in the early conceptual design phase, usually after the definition of the building program that stands for area requirement estimations according to the requests of the occupants.

---

[118] Sanjay Mittal and Felix Frayman, "Towards a Generic Model of Configuraton Tasks.," in *IJCAI*, vol. 89, 1989, 1395–1401.

2.4 Computational Approaches to Layout Design Problem

Computational approaches to layout design, started with facility layout problems in the 1950s.[119] Since the late 1950s, engineers and architects developed computational models to tackle many layout problems whose scale range from computer chips[120] to urban design[121]. Layout design approaches mainly diverge in how they define the layout problem (the solution representation, constraint formulation etc.) and the search methods they use to compute solutions.

Computational approaches to layout design problems can be classified as construction and improvement methods. A constructive method starts from scratch and builds the layout in sequent actions. This action can be the placement of an individual space in each consecutive step. Improvement approaches, on the other hand, start with complete solutions and improve this solution in sequent actions. This action can be the pairwise exchange of layout elements in every phase.

2.4.1 Construction methods

Construction methods work close to state-space search, which is assumed as a classical search method. It can be defined as "the process of looking for a sequence of actions that reaches the goal is called search".[122] Construction methods represent the search space as a tree of states in between the initial state and goal state. Exact algorithms and some heuristic algorithms are in this category.

---

[119] Tjalling C. Koopmans and Martin Beckmann, "Assignment Problems and the Location of Economic Activities," *Econometrica: Journal of the Econometric Society*, 1957, 53–76.

[120] Kazuhiro Ueda, Hitoshi Kitazawa, and Ikuo Harada, "CHAMP: Chip Floor Plan for Hierarchical VLSI Layout Design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 4, no. 1 (1985): 12–22.

[121] Pirouz Nourian, "Configraphics: Graph Theoretical Methods for Design and Analysis of Spatial Configurations," *A+ BE| Architecture and the Built Environment* 6, no. 14 (2016): 1–348.

[122] Russell and Norvig, *Artificial Intelligence*, 66.

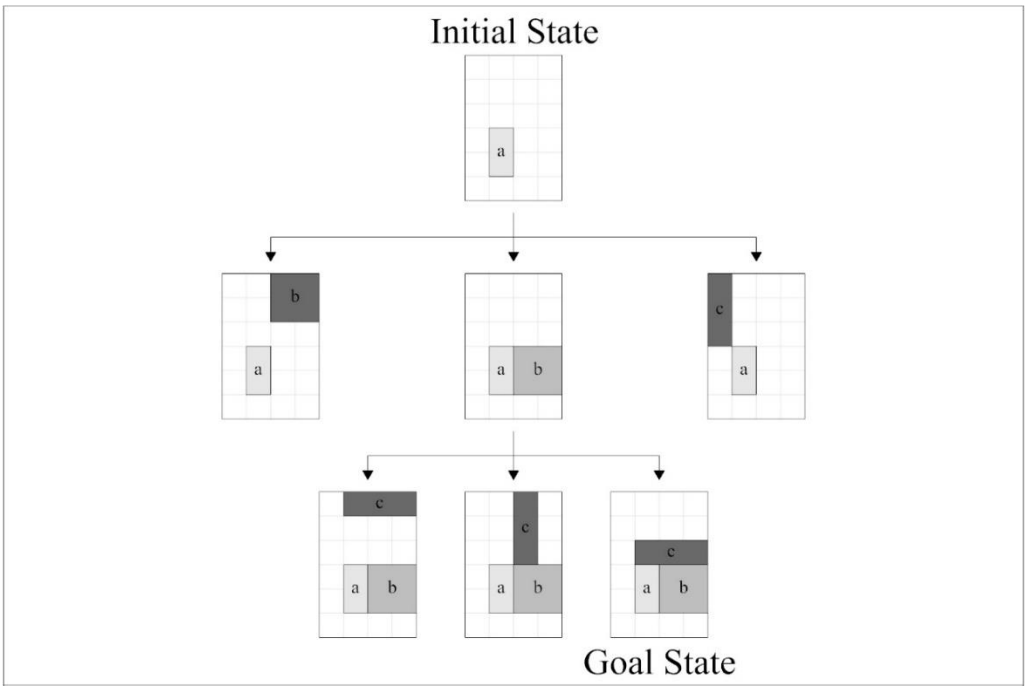


Figure 4. Schematic description of a simple layout construction algorithm (Drawn by the author).

Exact Algorithms

Exact approaches are precise algorithms that target optimal solutions. A very basic exact approach is the brute-force search algorithm that can enumerate all layout options and select the best one. Layout problems are, however, "NP-complete" (non-deterministic polynomial), which means optimal solutions require extended amounts of time even for small scale problems, as the solution space grows exponentially even with a low number of layout elements.[123] Therefore, exact approaches to layout design problems need either to narrow their search space and simplify the conditions for the optimal solution or require more sophisticated algorithms than brute-force search.

[123] Jun H. Jo and John S. Gero, "Space Layout Planning Using an Evolutionary Approach," *Artificial Intelligence in Engineering* 12, no. 3 (1998): 3.

The automatic furniture layout tool of Abdelmohsen et al.[124] is an exact approach with relaxed conditions and narrow search space. The tool places the furniture according to specific object-object and object-space relations rather than exhausting all placement options.[125] Several other exact layout design approaches use branch and bound algorithms.[126] Branch and bound algorithms make selective enumeration by pruning certain useless branches within the search tree if the algorithm assess that it would not produce an optimal solution. This way, branch and bound algorithms can solve layout problems with a maximum 18 equal area layout elements, however, they fail to support real life layout problems with unequal area elements.[127]

Real world layout problems include a high level of complexity. Generating an optimal solution within this complexity is a demanding operation that exceeds the time-related and computational resources. Designers, instead of optimality, look for good enough solutions that are "satisficing" within the complex problem contexts.[128] Similarly, computational approaches to layout design should work with approximations to use the resources in more intelligent search procedures.

Construction Heuristics

Heuristic approaches are approximate methods which do not guarantee to find an optimal solution, but usually find good enough results that are close to the optimal

---

[124] Sherif Abdelmohsen et al., "A Heuristic Approach for the Automated Generation of Furniture Layout Schemes in Residential Spaces," in *Design Computing and Cognition '16* (Springer, Cham, 2017), 459–75, https://doi.org/10.1007/978-3-319-44989-0_25.

[125] Abdelmohsen et al., 499.

[126] M.s. Bazaraa, "Computerized Layout Design: A Branch and Bound Approach," *AIIE Transactions* 7, no. 4 (01 1975): 432–38, https://doi.org/10.1080/05695557508975028; Ulrich Flemming et al., "Hierarchical Generate-and-Test vs Constraint-Directed Search," in *Artificial Intelligence in Design'92* (Springer, 1992), 817–838, https://link.springer.com/chapter/10.1007/978-94-011-2787-5_41.

[127] Russell D. Meller and Kai-Yin Gau, "The Facility Layout Problem: Recent and Emerging Trends and Perspectives," *Journal of Manufacturing Systems* 15, no. 5 (1996): 351–366.

[128] Herbert A. Simon, "Rational Choice and the Structure of the Environment.," *Psychological Review* 63, no. 2 (1956): 129.

solution.[129] In the absence of design support by exact approaches, heuristic search can still provide a layout solution that can provide insight into the design problem and solution space. Several examples have been developed that makes use of heuristics.

SHAPE is a construction heuristic that produces unequal area layouts by assigning smaller grid-like modules for every department.[130] SHAPE assigns departments according to the given major and minor production flows. Most common department within the production flows is chosen as the center and the others grow from this initial point. Although SHAPE can operate with a large number of layout elements, it also produces departments with irregular form. NLT is another constructive approach that assigns unequal area rectangular shapes for departments.[131] NLT uses a multiple stage framework that handles area requirements and adjacency relations in different stages.

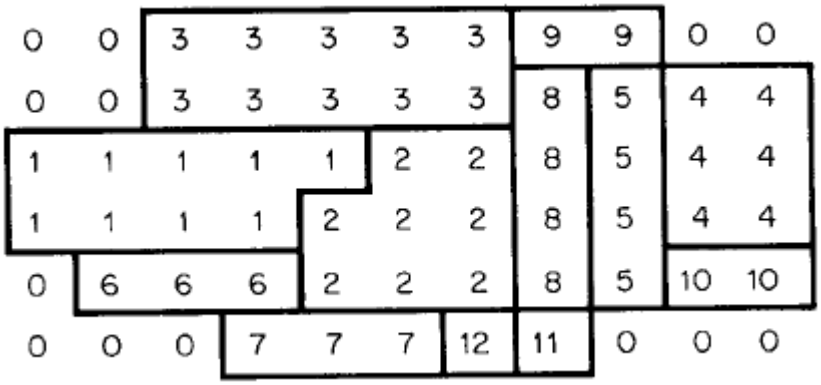


Figure 5. A final layout by SHAPE. (Hassan, Hogg, and Smith, 1986)

[129] Yehuda E. Kalay, *Architecture's New Media: Principles, Theories, and Methods of Computer-Aided Design* (MIT Press, 2004).

[130] Mohsen MD Hassan, Gary L. Hogg, and Donald R. Smith, "SHAPE: A Construction Algorithm for Area Placement Evaluation," *International Journal of Production Research* 24, no. 5 (1986): 1283–1295.

[131] Drew J. Van Camp, Michael W. Carter, and Anthony Vannelli, "A Nonlinear Optimization Approach for Solving Facility Layout Problems," *European Journal of Operational Research* 57, no. 2 (1992): 174–189.

These approaches, although successful in generating solutions, can grow and evaluate solutions in a state-by-state procedure. However, design problems are not solved one by one, but processed in an integrated way. As Lawson[132] points out, "Design solutions are often holistic responses". In the case of layout problems, the placement order of spaces does not matter if they take the designer to the same solution. Thus, the insignificance of the solution path makes construction algorithms less successful in terms of layout problems. Thus, designers usually work with completed solutions rather than evaluating them on the way. Improvement methods offer an effective alternative to these design issues.

2.4.2 Improvement Methods

Improvement methods carry out a local search operation. Local search "evaluates" and "modifies" one or more solutions rather than analyzing the paths of transfer options; thus, these methods are advantageous for problems in which only accounts for the solution not the sequence of actions to reach it.[133] There are heuristic and metaheuristic improvement methods.

---

[132] Lawson, *How Designers Think*, 122.

[133] Russell and Norvig, *Artificial Intelligence*, 120.

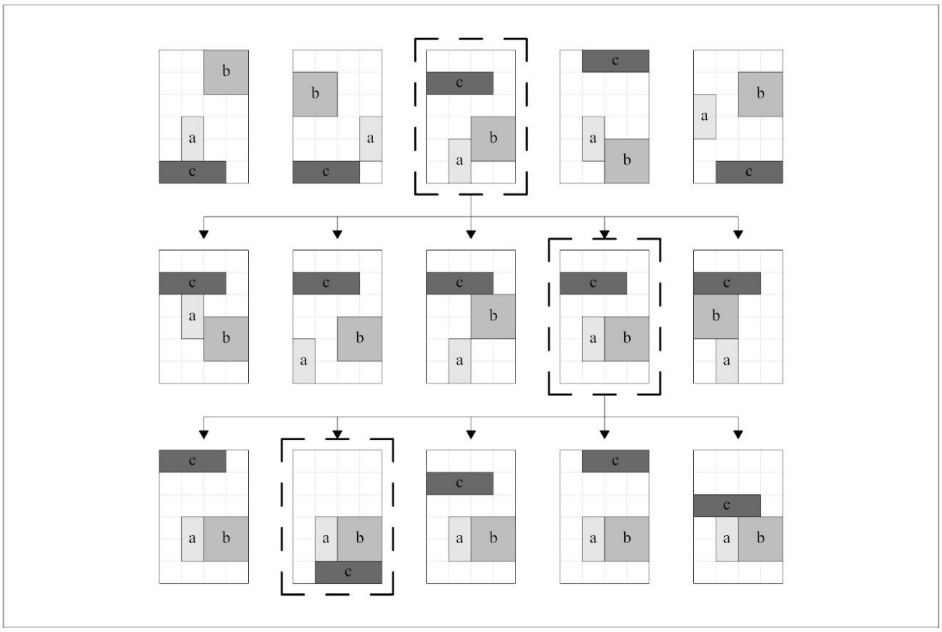


Figure 6. Schematic description of a simple layout improvement algorithm, where layout elements a, b and c are to be arranged into a compact building form (Drawn by the author).

Improvement Heuristics

CRAFT is an early improvement heuristic approach that produces unequal area solutions by pairwise exchange operations between two or three layout elements.[134]After the exchange operations, CRAFT estimates a cost function to choose the best exchange operation that caused the largest reduction in cost. This operation goes on until there is no possible way to cause a reduction by pairwise exchanges. MULTIPLE develops CRAFT's algorithm to solve multiple-floor production facility layouts.[135] MULTIPLE improved the number of exchange operations by introducing space-filling curves and provided an additional cost function to limit the irregularity of department geometries. A more recent

[134] Gordon C. Armour and Elwood S. Buffa, "A Heuristic Algorithm and Simulation Approach to Relative Location of Facilities," *Management Science*, no. 2 (1963): 294.

[135] Yavuz A. Bozer, Russell D. Meller, and Steven J. Erlebacher, "An Improvement-Type Layout Algorithm for Single and Multiple-Floor Facilities," *Management Science*, no. 7 (1994): 918.

improvement heuristic by Guo and Li[136] generates sophisticated multi-floor layouts with horizontal and vertical circulation elements. They used a two-step procedure where a multi-agent system generates an initial with correct topological relations, and then another process randomly pushes or pulls the faces of layout elements until the layout satisfies user-defined geometric criteria.[137]

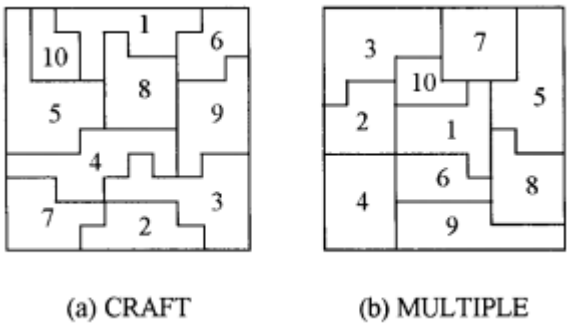


Figure 7. Example layouts from CRAFT (left) and MULTIPLE (right). (Lee and Kim, 2000)

According to Kalay, heuristic approaches can solve layout problems that exact approaches fail, however, they do not generate novel solutions.[138] Heuristic approaches are problem specific; they process on problem-specific knowledge to make intermediate decisions. The limitation of the search space with rule of thumbs creates faster but also routine solutions. It is also possible for a designer to lack an initial description for the required problem.

Metaheuristics

Metaheuristics are generic algorithms that can be applied to solve any search

---

[136] Zifeng Guo and Biao Li, "Evolutionary Approach for Spatial Architecture Layout Design Enhanced by an Agent-Based Topology Finding System," *Frontiers of Architectural Research* 6, no. 1 (March 1, 2017): 53–62, https://doi.org/10.1016/j.foar.2016.11.003.

[137] Guo and Li, 54,57.

[138] Kalay, *Architecture's New Media*.

problem, if solutions can be easily generated and evaluated. According to a definition provided by Talbi,[139] metaheuristics are "upper level general methodologies (templates) that can be used as guiding strategies in designing underlying heuristics to solve specific optimization problems". Unlike other heuristic approaches, metaheuristics do not require the definition of any problem specific solution method that initially limits the search space. The success of metaheuristics, instead, comes from the balanced exploration and exploitation they put into the search process.[140] There are single solution and population-based metaheuristics.

Single solution metaheuristics start with a single solution and achieve the result by making alterations on the initial solution. Chao and Liang,[141] developed a tabu search algorithm to solve unequal area multiple-floor facility layout problems. Their tabu search algorithm is based on swapping certain departments which puts bad swapping moves in a dynamic tabu list to limit their use for a period. Simulated annealing (SA) is another single solution metaheuristic that starts with a high exploration rate then reduces it gradually to escape local optima in the initial phases.[142] Yi and Yi,[143] developed a simulated annealing algorithm to assign a collection of three-dimensional apartment block types in a truncated box boundary.

---

[139] El-Ghazali Talbi, *Metaheuristics : From Design to Implementation* (Hoboken, N.J. : John Wiley & Sons, c2009., 2009), 1.

[140] Mauro Birattari et al., "Classification of Metaheuristics and Design of Experiments for the Analysis of Components," 2001, http://hdl.handle.net/2013/ULB-DIPOT:oai:dipot.ulb.ac.be:2013/77018.

[141] Lou Y. Liang and Wen C. Chao, "The Strategies of Tabu Search Technique for Facility Layout Optimization," *Automation in Construction* 17, no. 6 (2008): 657–669.

[142] Stuart Russell and Peter Norvig, "A Modern Approach," *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs* 25 (1995): 27.

[143] Hwang Yi and Yun Kyu Yi, "Performance Based Architectural Design Optimization: Automated 3D Space Layout Using Simulated Annealing" (2014 ASHRAE/IBPSA-USA Building Simulation Conference, American Society of Heating, Refrigeration, and Air-Conditioning Engineers (ASHRAE), 2014), https://experts.illinois.edu/en/publications/performance-based-architectural-design-optimization-automated-3d-.

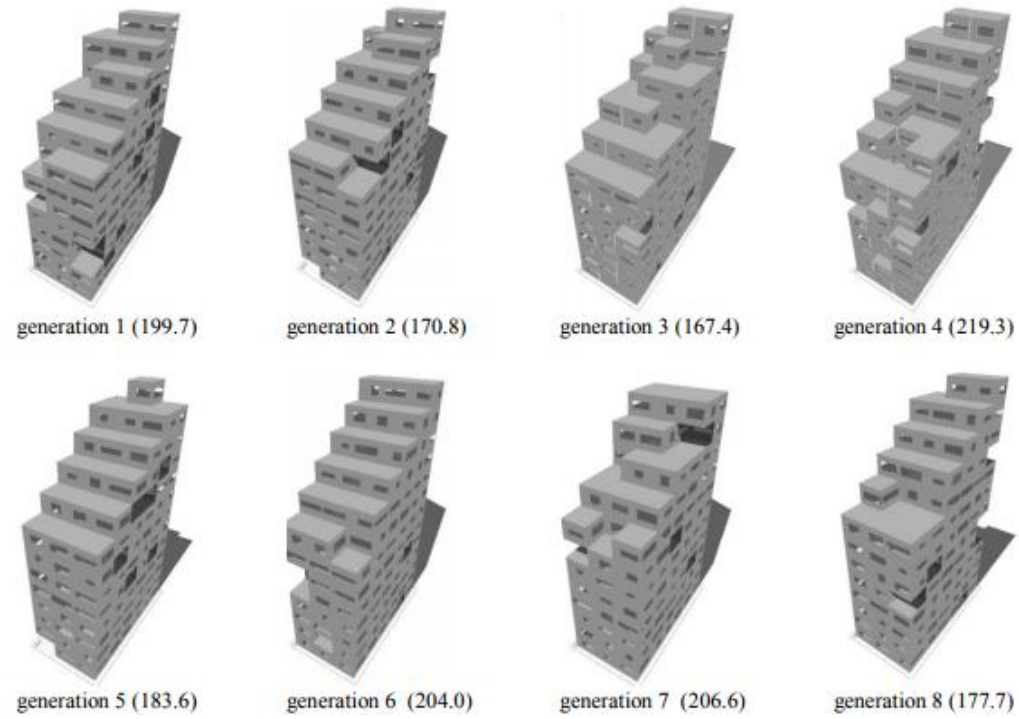


Figure 8. Apartment configuration by Yi and Yi's SA algorithm (Yi and Yi, 2014).

Single solution metaheuristics approaches have an inherent limitation, which is their inability to operate in search space with multiple local maximum points. This is because a single solution, unaware of the global context, might not be able to escape the local maxima and prematurely converge to sub-optimal solutions. Population-based search, on the other hand, can explore the search space in a more efficient way by simultaneously exploring many different points in the search space.

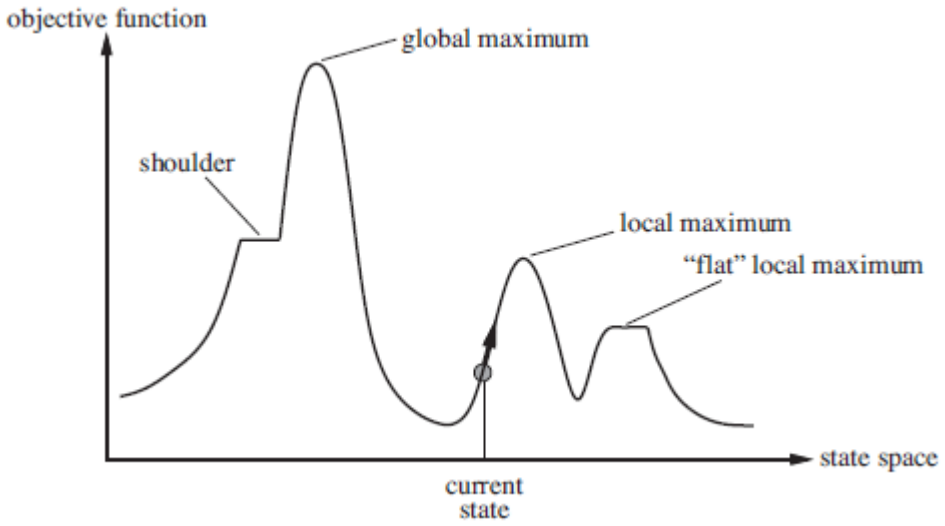


Figure 9. The fitness landscape of design problems is usually multimodal with multiple peaks. (Russell and Norvig, 1995)

Population-based metaheuristics expand the exploration power further by incorporating more solutions at the start. Ant colony optimization imitates the swarm behavior of ants in finding the shortest path to their solution.[144] Shea, Sedgwick, and Antonuntto[145] implemented ant colony optimization for the design of building envelopes according to lighting and cost. Evolutionary approaches are several population-based metaheuristics that imitates Darwin's theory of evolution. Genetic programming (GP) and genetic algorithms (GA) are also popular evolutionary methods in layout design. GA is developed by Holland.[146] GP is developed by Koza[147] as an extension of GA that searches for effective computer programs instead of direct solutions.

---

[144] Talbi, *Metaheuristics*.

[145] Kristina Shea, Andrew Sedgwick, and Giulio Antonuntto, "Multicriteria Optimization of Paneled Building Envelopes Using Ant Colony Optimization," *Intelligent Computing in Engineering and Architecture*, 2006, 627–636.

[146] John H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence* (MIT press, 1992).

[147] John R. Koza et al., "Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming," *IEEE Transactions on Evolutionary Computation* 1, no. 2 (1997): 109–128.

EDGE is a GP approach by Jagielski and Gero[148] that improved a previous QA heuristic by Liggett in multiple-floor facility layout problems. Quadratic Assignment (QA) is a combinational assignment problem that arranges a set of equal sized facilities to fixed locations. Verma and Thakur[149] developed a GA that generates multiple floor apartment layouts according to adjacency requirements and a traditional Indian system of layout rules. Dino[150] developed Evolutionary Architectural Space Layout Explorer (EASE). EASE generates 3D architectural layouts for a given building mass according to various user-defined constraints.[151] EASE uses a specific genotype definition to deal with overlapping and empty areas; and takes the advantage of additional repair operators to aid convergence.[152]

---

[148] Romuald Jagielski and John S. Gero, "A Genetic Programming Approach to the Space Layout Planning Problem," in *CAADFutures 1997: Proceedings of the 7th International Conference on Computer Aided Architectural Design Futures*, CAAD Futures (Kluwer Academic Publishers, 1997), 875–84.

[149] Manisha Verma and Manish K. Thakur, "Architectural Space Planning Using Genetic Algorithms," in *Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference On*, vol. 2 (IEEE, 2010), 268–275, http://ieeexplore.ieee.org/abstract/document/5451497/.

[150] Ipek Gürsel Dino, "An Evolutionary Approach for 3D Architectural Space Layout Design Exploration," *Automation in Construction* 69 (2016): 131–150.

[151] Dino, 131,132.

[152] Dino, 138,140.

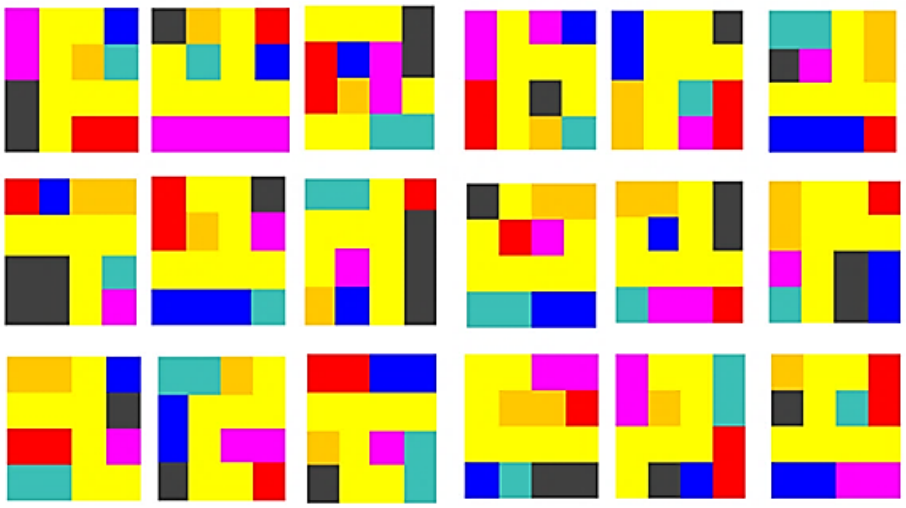


Figure 10. A set of apartment layout solutions generated by Verma and Thakur's algorithm. (Verma and. Thakur, 2010)

Doulgerakis[153] used GP to divide multiple layers of rectangular geometries into smaller units and then used an agent-based algorithm to assign program elements to created subdivisions in multiple floors according to the area requirements, which is called Area Dissection (AD). AD takes an initial floor shape and divides it into unequal segments. The units are placed within these smaller areas according to their topological relations and geometric criteria. Knecht and König[154] developed an approach that utilizes kd algorithm to divide a predefined area and then uses GA to fit these divisions into topological, rational, and dimensional constraints.

[153] A. Doulgerakis, "Genetic Programming + Unfolding Embryology in Automated Layout Planning" (UCL (University College London), 2007), http://discovery.ucl.ac.uk/4981/.

[154] Katja Knecht and Reinhard König, "Generating Floor Plan Layouts with Kd Trees and Evolutionary Algorithms," in *Generative Art Conf*, 2010, 238–253.

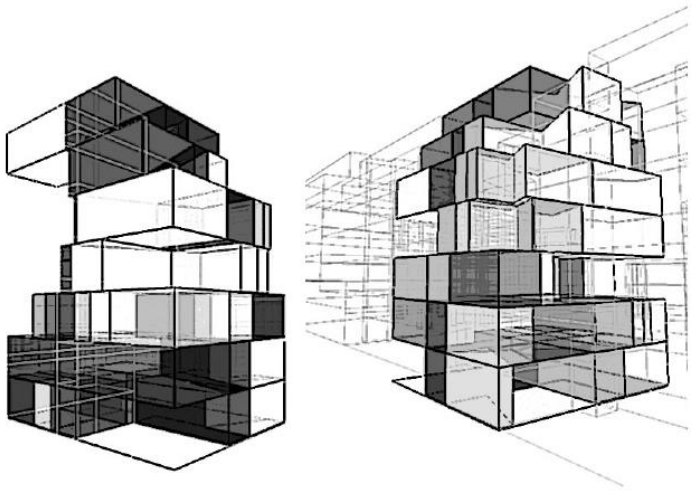


Figure 11. Perspective views of layout solutions by Doulgerakis' algorithm. (Doulgerakis, 2007)

Flack[155] developed a GA that creates layouts according to topology, area requirements, and given ratios for individual rooms with Area Positioning (AP). AP positions fixed or variable geometries in a predefined boundary. This method increases the control over the geometry of layout elements, so it makes possible to create specialized layout elements. This specialization makes AP more advantageous in small-scale layouts as minor differences in the unit geometries contribute more to the layout exploration. The algorithm can also work with a non-rectilinear geometry as a boundary. EPSAP is a hybrid approach that uses Evolutionary Search (ES) and Stochastic Hill Climbing (SHC) to generate multiple floor generic layouts.[156] EPSAP uses a set of repair rules predefined for certain problems that are randomly implemented. The time requirement of the correction rules is balanced with the use of a different global search mechanism, ES, which is a simpler version of GA that produces solutions solely by mutation. EPSAP creates detailed layouts with vertical and horizontal circulations, windows, and doors. Interactive Layout Recommender

---

[155] Robert W. J. Flack, "Evolution of Architectural Floor Plans" (Brock University, 2011), http://dr.library.brocku.ca/handle/10464/3409.

[156] Eugénio Rodrigues, Adélio Rodrigues Gaspar, and Álvaro Gomes, "An Approach to the Multi-Level Space Allocation Problem in Architecture Using a Hybrid Evolutionary Technique," *Automation in Construction* 35 (2013): 482–498.

System (ILRS)[157] is another approach by Bahrehmand et al. that generates layout options from a predefined set of regular and irregular shaped layout units. ILRS provides a real time rating interface where the user can give any layout a rating from one to five to increase or decrease its chance of survival in the evolutionary process.[158]

Figure 12. Three floors of layout configurations by EPSAP. (Rodrigues, Gaspar, and Gomes, 2013)

Rosenman[159] developed a GA to create house layouts according to area and adjacency requirements. The approach uses Hierarchical generation (HG) to produce layouts gradually from basic elements to complex configurations. Algorithm first creates rooms from rectangular modules, then these rooms are allocated into zones, and finally zones are arranged into house layouts. GENETICA is a GP approach that designs multiple floor hotel layouts with fully furnished hotel rooms.[160] GENETICA defines layout items, for example doors, beds, sitting arrangements etc. as standard

[157] Arash Bahrehmand et al., "Optimizing Layout Using Spatial Quality Metrics and User Preferences," *Graphical Models* 93, no. Supplement C (September 1, 2017): 25–38, https://doi.org/10.1016/j.gmod.2017.08.003.

[158] Bahrehmand et al., 31.

[159] M. A. Rosenman, "The Generation of Form Using an Evolutionary Approach," in *Evolutionary Algorithms in Engineering Applications* (Springer, 1997), 69–85, http://link.springer.com/chapter/10.1007/978-3-662-03423-1_4.

[160] Lefteris Virirakis, "GENETICA: A Computer Language That Supports General Formal Expression with Evolving Data Structures," *IEEE Transactions on Evolutionary Computation* 7, no. 5 (2003): 456–481.

units with fixed dimensions. These items are arranged according to “physical space” which is the real space used by the layout item that cannot be overlapped, and “functional space” which stands for the space required to use such item.[161] Layouts are generated through the arrangement of layout items, and then the physical boundaries are procedurally generated by a CAD application.

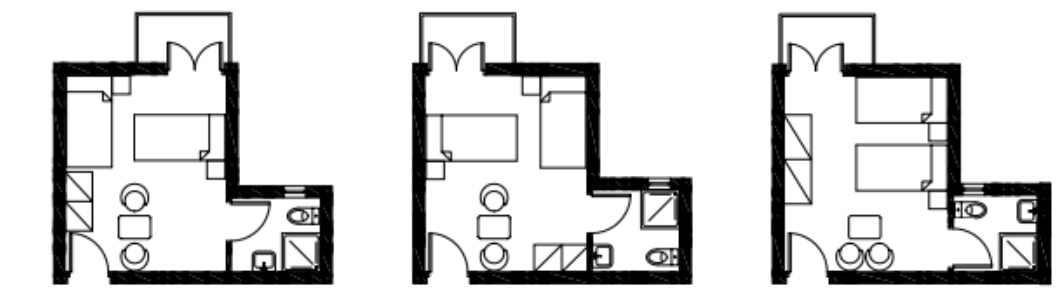


Figure 13. Hotel room arrangements by GENETICA. (Virirakis, 2003)

Metaheuristic approaches require an additional generative heuristic for layout design problem. This research will utilize an AP heuristic because of its advantages in SFH layouts. AP gives more control on the geometrical representation of layout elements in the initial state. This control creates a possibility to define layout elements in various characters such as open spaces or vertical and horizontal circulation spaces. AP, consequently, requires more computational resources to cope with this variety of layout elements, however this is not going to be a critical issue in the SFH context which involves less number of layout elements. The possibility to neglect irregular forms for layout element is another advantage of AP heuristic. The user can represent the layout elements within the limits of a geometrical shape such as a rectangle.

Among the evolutionary metaheuristic approaches, GA gathered more attention on their success on design problems. Goldberg[162] sees an evident similarity between the

---

[161] Virirakis.

[162] David E. Goldberg, “Genetic Algorithms as a Computational Theory of Conceptual Design,” in *Applications of Artificial Intelligence in Engineering VI* (Springer, 1991), 3–16, https://link.springer.com/chapter/10.1007/978-94-011-3648-8_1.

design phases and the mechanism of GAs. Rodrigues[163] provided the popularity of GAs in layout problems with a literature survey. Flack[164] provided the efficiency of GAs over GPs by the experiments on layout design. Accordingly, this research develops a GA approach because of its success in coping with vague problem definitions and locating valid solutions in multimodal search spaces.

2.5 Genetic Algorithms (GA)

GA is a search algorithm that is introduced by Holland[165] to demonstrate the capabilities of adaptation in natural systems and to emulate this process in creating new artificial systems. GA takes the advantage of two concepts from evolutionary biology, survival of the fittest, and natural selection, to locate solutions in multimodal and complex search spaces. GA, similar to evolution, is a blind process that lacks a reason for the realized actions. On the other hand, GA uses the information in the past generations to predict new exploration directions with improved performance.[166]

GA is beneficial in problems that are too complicated to tackle with fast and abstract solution methods. In this problem context, GA can be used to generate novel approaches by an efficient trial-and-error process. They have been successfully employed in various interesting fields, such as game AI,[167] musical composition,[168]

---

[163] Eugénio Rodrigues, "Automated Floor Plan Design: Generation, Simulation, and Optimization; Desenho Automático de Plantas: Geração, Simulação e Optimização" (Universidade de Coimbra, 2014), http://oatd.org/oatd/record?record=handle%5C%3A10316%5C%2F25438.

[164] Flack, "Evolution of Architectural Floor Plans."

[165] Holland, *Adaptation in Natural and Artificial Systems*.

[166] Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*.

[167] N. Cole, S. J. Louis, and C. Miles, "Using a Genetic Algorithm to Tune First-Person Shooter Bots," in *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, vol. 1, 2004, 139–145 Vol.1, https://doi.org/10.1109/CEC.2004.1330849.

[168] P. M. Gibson and J. A. Byrne, "NEUROGEN, Musical Composition Using Genetic Algorithms and Cooperating Neural Networks," in *1991 Second International Conference on Artificial Neural Networks*, 1991, 309–13.

or abstract painting.[169]

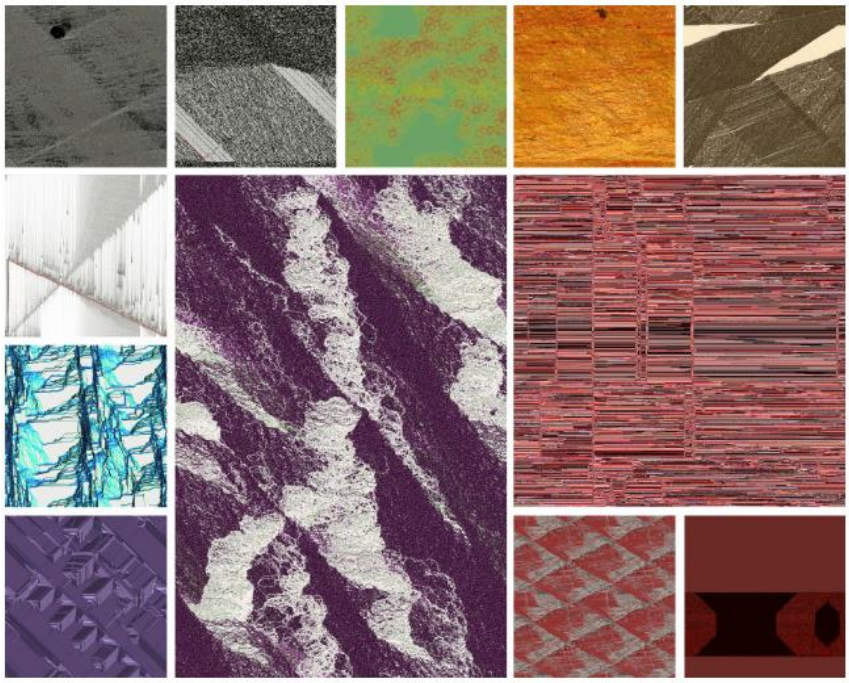


Figure 14. Abstract paintings by EVOECO. (Feng and Ting, 2014)

A population of solutions are generated and evaluated in every step to assess their fitness in the environment. In every generation, a group of less fit solutions dies, while another group of survived solutions gains a chance to reproduce to fill the gaps in the population with better solutions. The success of GA depends highly on the initial user definition that represents the problem.

Evolutionary processes require the definition of solution candidates in both genotype and phenotype. Genotype definition refers to the underneath genetic structure of the solution that is put under genetic transformations, whereas phenotype definition refers to the outer appearance of the solution that is considered for the fitness

[169] Sheng-Yu Feng and Chuan-Kang Ting, "Painting Using Genetic Algorithm with Aesthetic Evaluation of Visual Quality," in *Technologies and Applications of Artificial Intelligence* (Springer, 2014), 124–135, http://link.springer.com/chapter/10.1007/978-3-319-13987-6_12.

evaluations.

2.5.1 Mechanism of genetic algorithm

The evolutionary process requires an *initial* population of solutions which is usually generated randomly. A user-defined *fitness function* evaluates these solutions in terms of their success in satisfying the user-defined conditions. Parent solutions are *selected* from the better performing candidates. Parent solutions are *reproduced* to generate a new population of self-similar design solutions. The new entities are generated by *crossover* that mixes the genotype of both parents randomly into offspring solutions. These offspring are *mutated* to make random changes within their genetic structure, which is required for securing the genetic variety.

The set of individuals generated in each repetition of this mechanism is named as a *generation.* The runtime of a GA is limited with the *termination* of user-defined criteria. These criteria can refer to achieving a level of success within the fitness function, or reaching a maximum number of total generations.

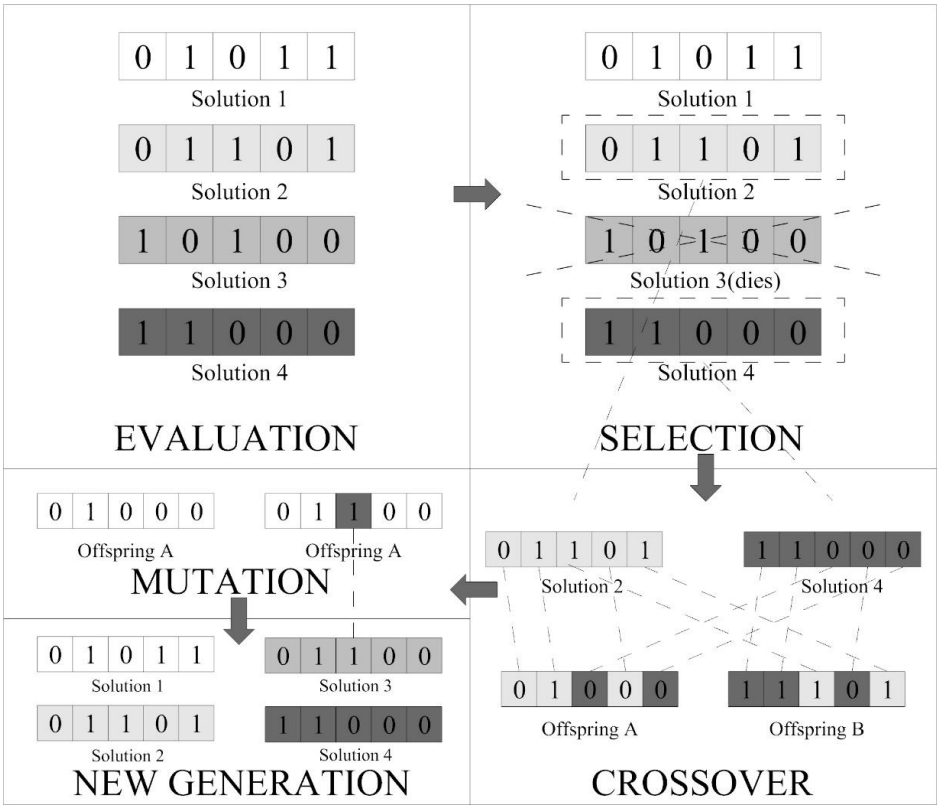


Figure 15. GA mechanism (Drawn by the author)

The review on past evolutionary approaches to layout problem generally considered examples with a limited user interaction until now. GA, as a metaheuristic method, is a black-box algorithm. Black-box algorithms' solution process is not visible to the user. User can only interact with the interface in the initial state through the definition of inputs. Design problems are characteristically "underdetermined"[170] where the problem definition constantly changes through the observations of the designer. Although, GA's generic structure allows the user to modify problem definition through the observation of results in consequent runs, this can be an exhaustive process for the user. GA's capability to explore large solution spaces comes with a disadvantage on its use of resources. Skiena[171] asserts that GA uses

[170] Dorst, "The Problem of Design Problems."

[171] Steven S. Skiena, *The Algorithm Design Manual*, 2nd ed. (Springer Publishing Company, Incorporated, 2008).

long periods of time on “nontrivial problems”. GA’s requirement of long periods of time is problematic because of the latency between the problem definition and feedback mechanism. Alternatively, a number of interactive approaches were developed to eliminate distance between the designer and computational models by increasing user’s effect on the computational process.

2.5.2 Interactive genetic algorithm

The interactivity does not need to cover just the architect but take another way to directly give the process to the user. This way the designer can design the possible procedure between the user and the generative mechanism. Such as that approach. But also this tool should not fall in the pit of a large knowledge domain Interactive genetic algorithm (IGA) refers to a genetic algorithm with a degree of user aid on certain parts of the evolutionary mechanism. In the most extreme case, a user can completely replace the fitness function by means of human evaluation. In this way, IGAs can be used in problems where an exact mathematical function is not available to evaluate solutions. A notable example is GADES,[172] an IGA that can generate 3D objects based on the aesthetic preferences of the user. GADES requires human evaluation for each individual in the population and slowly converges into a population of “interesting solutions”.[173]

[172] Bentley and Corne, “Introduction to Creative Evolutionary Systems.”

[173] Bentley and Corne, 42.

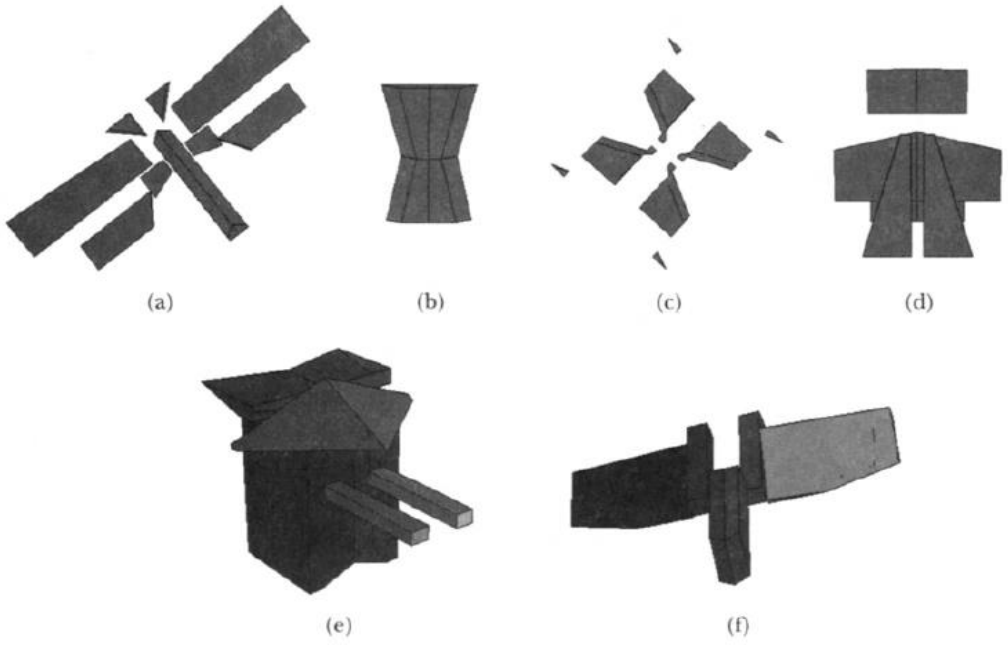


Figure 16. Generated 3D solutions by GADES. (Bentley and Corne, 2002)

Numerous layout design approaches take the advantage of IGAs. Rosenman's hierarchical generation approach allows users to choose satisfactory layout and layout elements during the search process.[174] The hierarchical structure of the generation process gives the user the opportunity to evaluate the solutions in intermediate stages. Room pairs are generated from a population of chosen room geometries which in the end generate the whole layout. Michalek, Choudary and Papalambros[175] proposed an IGA approach that allows changing the problem definition together with the geometric modifications on the solutions. User can add, modify, or delete both constraints and objectives to modify the problem definition. The user can modify the generated solutions via the user interface and then iterate over the modified layout. The algorithm also gives permission to guide the search process by initial layouts. Quiroz, Louis, Banerjee, and Dascalu[176] developed a

[174] Rosenman, "The Generation of Form Using an Evolutionary Approach."

[175] Jeremy Michalek, Ruchi Choudhary, and Panos Papalambros, "Architectural Layout Design Optimization," *Engineering Optimization* 34, no. 5 (2002): 461–484.

[176] J. C. Quiroz et al., "Towards Creative Design Using Collaborative Interactive Genetic

collaborative IGA for layout design. Users, in this example, guide the evolution of the population by the selection of generated alternatives in intermediate stages. Users can only see a limited portion of their own population, instead they are provided with the population examples from the other user.

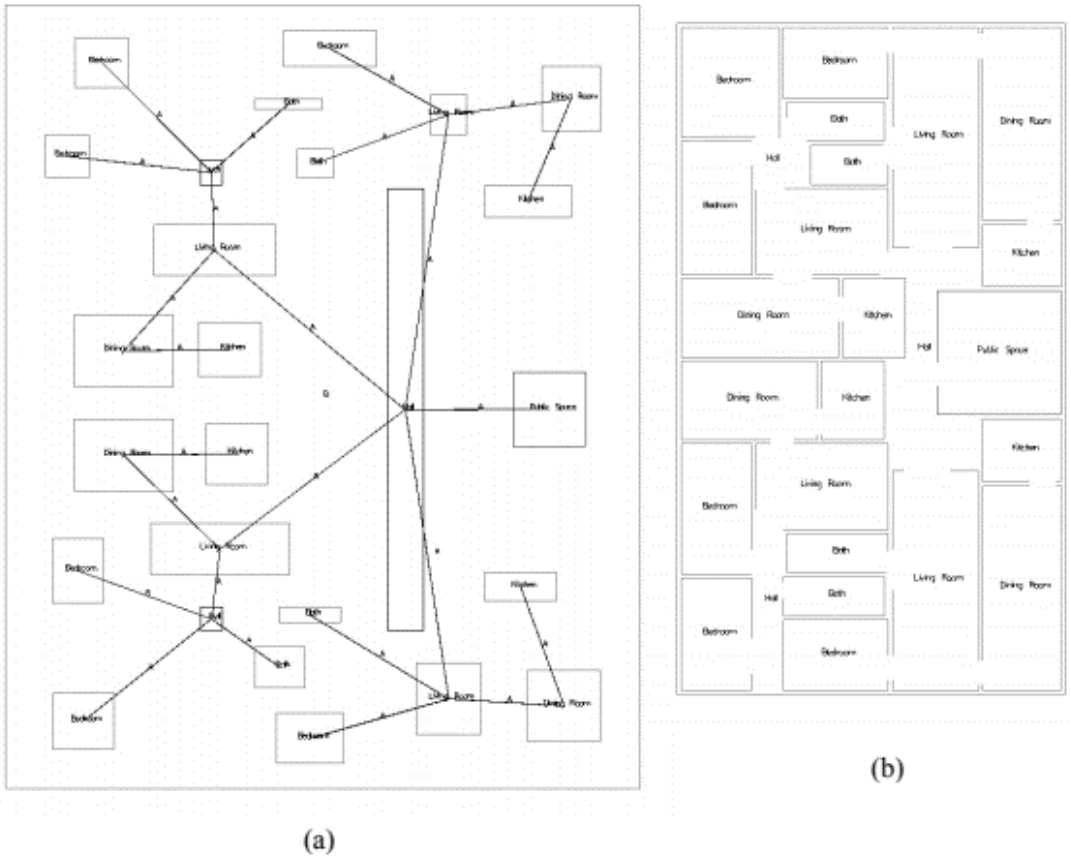


Figure 17. Layout generated from the initial user sketch by the IGA of Michalek. (Michalek, 2002)

An interactive fitness function, despite the possibility to integrate subjective and vague criteria, creates an enormous burden on the designer to evaluate a large set of solutions through generations. Considering that GA's work with large population

---

Algorithms," in *2009 IEEE Congress on Evolutionary Computation*, 2009, 1849–56, https://doi.org/10.1109/CEC.2009.4983166.

spaces, the manual evaluation of the user both takes a large amount of time and causes the designer to lose the ability to compare within the large multiplicity of solutions. However, the approach of Michalek et al. is found beneficial, because of its resemblance to the iterative character of the design process. The approach places the user in an active position by allowing changes on the problem definition as well direct changes within the search process such as sketch revisions or initial layouts.

To conclude, SFH, despite of its small scale, is an important architectural design problem that requires the continuous collaboration of the architect and the user. Alternative spatial solutions that are developed by the architects serve as a medium for this collaboration in terms of revealing the intentions of both parties. Architectural layout design is an important part of this process as it uses essential information about the user's lifestyle such as area requirements and furniture organizations for activities and their interrelations. Time restrictions and the limits of human cognition on the alternative generation process can reduce the affectivity of layout design generations. Computational layout design methods can enhance architect's abilities by the computational data processing and storing capabilities. A review on the current computational methods shows the affectivity of GA because of their capability to work with vague problem definitions and finding satisficing solutions from a vast solution space. However, GA also limits the control of the designer over the generation process by its black-box working principle. IGA approaches are found beneficial in terms of the level of designer control during the search process. This research argues that the reviewed IGA approaches are deficient in terms of supplying the specific requirements of SFH thus, presents a new computational approach for the generation of SFH layouts.

CHAPTER 3

TOOL DEVELOPMENT

In the previous chapter, the potentials and limits of computational layout design approaches on conceptual SFH problems are discussed. Metaheuristic approaches and especially GAs are found more advantageous due to their capabilities in handling the computational complexity of such conceptual problems. GA is a divergent search method that is advantageous in vague design problems with multiple solutions and design objectives. On the other hand, the advantages of GA also make the designer more distant from the design process because of the black-box formulation of GA. An interactive GA approach is found beneficial in improving designers' control over the computational search process. This can allow the designer to be more active in the design process beyond the definition of inputs and the mere observation of final solutions. As mentioned in the previous section, an interactive GA method for SFH layout problems is not encountered in the current literature. This, therefore, is one of the main contributions of this research to the layout design research.

Ho-Gen (House Generator) is an interactive computational model and a tool that is developed to support designers in the layout design of SFH. Ho-Gen can generate multi-floor and unequal area SFH layouts. Ho-Gen is not expected to generate complete and detailed layouts, but multiple layout alternatives to facilitate divergent exploration during conceptual design. Ho-Gen follows the generic representation, generation, evaluation, and guidance synthesis cycle described by Cagan et. al.[177] According to Cagan et al.,[178] the "representation" phase corresponds to the decisions about the search process with the level of detail in solution representations

[177] Cagan et al., "A Framework for Computational Design Synthesis."

[178] Cagan et al., 172.

and process or the comprehensiveness of the solutions to be involved in the search space; “generation” phase creates solutions according to the defined representation, “evaluation” phase rates the suitability of generated solutions to the user-defined criteria, and “guidance” phase uses this data to direct the search direction towards better solutions.

Ho-Gen implements an interactive GA for this synthesis process. The generation and guidance phases are realized by initialization, selection, crossover, and mutation algorithms while the user can also interfere with the process and guide the search by manually generated solutions. The evaluation is realized by a single criterion fitness function that is made of the weighted sum of geometrical and topological sub-criteria that are specific to SFH.

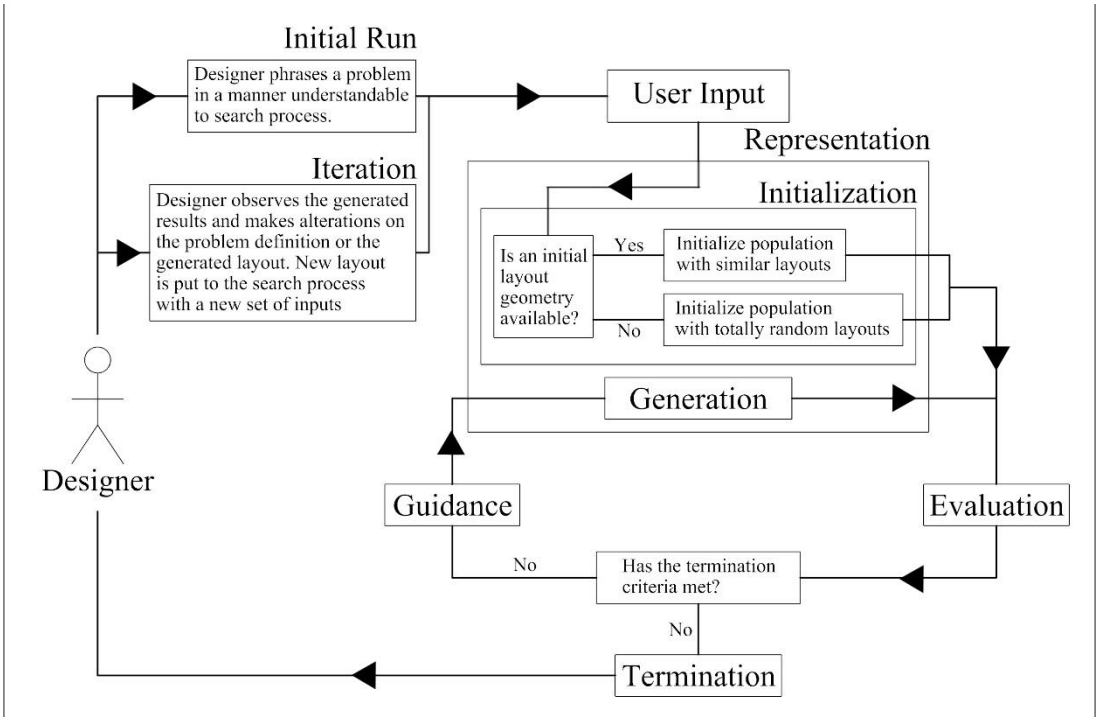


Figure 18. Framework representing the working principle of Ho-Gen (Drawn by author).

3.1 Representation

3.1.1 Building blocks

In Ho-Gen, a building has a hierarchically structured representation. A building is decomposed into its floors (FLO) and layout elements (LE). LEs can be optionally clustered into groups by the designer, if necessary. These groups (GRO) represent LEs that are related to each other and therefore need to be placed in close proximity with each other in the layout.

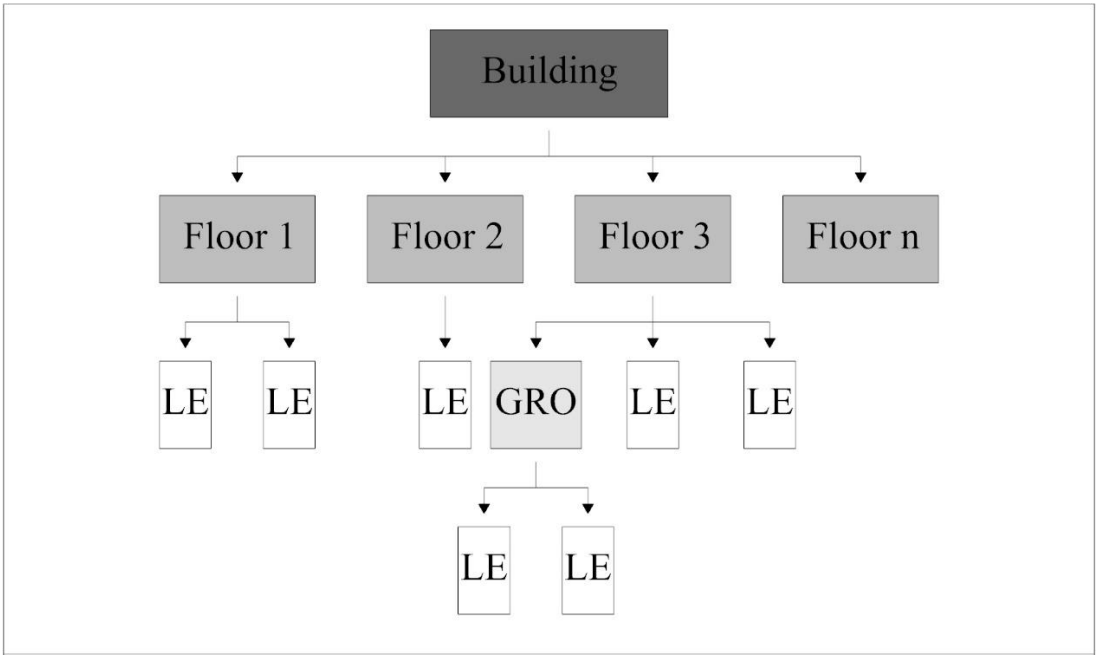


Figure 19. An example layout hierarchy in Ho-Gen (Drawn by the author).

Table 1. The layout elements (LEs) in Ho-Gen

Layout element (LE) categories	**Sub-categories**	**Description**
Spaces (SP)	Main Spaces (MS)	Spaces with SFH functions, i.e. bedroom, kitchen etc.
	Patio (Pa)	Outdoor spaces for open area activities
	Porch (Po)	Outdoor spaces usually near the entrance.
	Garage (G)	Indoor car parking.
Stairs (STA)	-	Spaces that facilitate the flow of occupants between floors.
Chimney (CHI)	-	Object description for the fireplace and its chimney.

Ho-Gen considers several basic assumptions that guide the formation of LEs as well as their physical characteristics. These assumptions are as follows:

- Spaces in Ho-Gen can be both indoors and outdoors. For instance, an indoor space is enclosed by surfaces (walls and slabs), or alternatively it can be an open or semi-open space outside the physical boundaries of the building (i.e. a porch).
- A LE can span multiple floors (i.e. stairs, double-space living room).
- LEs are theoretically bounded by rectangular prisms (but not necessarily physically bounded for outdoors spaces).
- For the search space reduction, LEs are allocated onto a grid with a size of 0.5 x 0.5 meters.

3.1.2 Interrelations

SFH involves a smaller amount of layout elements when compared to larger facilities. The computational complexity involved in a SFH layout task, then, can be stated as lesser than general because of the smaller amount of possible layout

combinations. This idea changes dramatically considering the importance of architectural form for SFH. Certainly, this general privilege on the form for SFH layout compositions reflects upon every separate relation between layout elements. This way, the computational complexity involved in SFH layouts should be evaluated on the quality of relations rather than the quantity. Such complexity requires the exploration of larger solution spaces because of the formal variation evident in the topologically identical SFH layouts.

Relationships play a critical role in Ho-Gen. LEs in Ho-Gen relate to each other in three types (Table 2). In the first type, layout elements form topological relations with one another. This relationship type is crucial to Ho-Gen, and will be discussed in Evaluation. In the second type, LEs form groups. Ho-Gen presents the user three options to define the grouping mechanism. First, the grouping process can be totally passed by letting every element form a group. Second, the user can initially specify a group for every LE manually. Third, LEs can be grouped according to their FLO. Lastly, every LE can be gathered into a total GRO. These floors are topologically connected by stairs (STA). In addition to that, the user can specify geometrical criteria that control the interrelation of separate floors with each other. This way, the final geometry of SFH can vary in the third-dimension as well.

Table 2. Relationship types in Ho-Gen

Relationship types	**From-To**	**Cardinality**	**Description**
Space-topological relations	LE to LE	1-To-1	0 – No adjacency 1 – Adjacent
Group relations a. None b. Manual c. By Floor d. Total	LE to GRO	1-To-Many	a- No groups. b- User defines group relations one-by-one. c- LE's in FLO make individual GROs. d- Every LE is gathered into one GRO.
Floor - LE relations	SP to FLO	1-To-1 or 1-To-Many	User defines the floor of LE

3.1.3 Initial User Interaction

The designer interacts with Ho-Gen through a group of inputs that define the layout problem together with the control mechanism over the generation process. These inputs can be divided into five groups: general, specific, evaluation, evolutionary, and termination. General inputs involve global variables that control the layout as a whole. Specific inputs are specialized inputs to control different types of LEs. Evaluation inputs include the coefficients that control the weight of different constraints during the search process. Evolutionary inputs make up the variables for genetic algorithm. Termination criteria represent the condition that halts the search.

General Inputs

Table 3. General inputs.

Name	Type	Description
Gi_m: Main Entrance	Direction	Specifies the direction for SFH's main entrance.
Gi_b: Boundary limits	{dimension, dimension}	X and Y dimensions for the rectilinear plot boundary.
Gi_m: Maximum cantilever	Dimension	Maximum cantilever distance between the vertical sequences of floors.
Gi_t: Topological relations	Matrix	the connection and adjacency relations between every separate layout element.
Gi_{fh}: Floor Height	Dimension	Vertical dimension between floors
Gi_{ch}: Chimney Height	Dimension	Vertical dimension of the chimney from ground
Gi_t: Porch Height	Dimension	Vertical dimension of porch's ceiling slab from ground

Specific Inputs

Table 4. Main Space Inputs

Name	Type	Description
MS^{1}_{n}: Room Name	Text	Name of the room for the user to check generated layout solutions.
MS^{1}_{a}: Room Area	Dimension	The area amount of the room. Circulation elements are automatically fixed to 0 to minimize their area.
MS^{1}_{min}: Minimum edge dimension	Dimension	The minimum dimension value for a separate edge of room geometry. Building regulations usually involve standard minimum dimensions for residential buildings.
MS^{1}_{max}: Maximum edge dimension	Dimension	The maximum dimension value for a separate edge of room geometry. User can also specify the upper bounds of edge dimensions for every layout element to control geometric ratios. Fixed automatically to Area / MinE when not specified.
MS^{1}_{v}: Room View	Direction	The direction for the layout element to have an unobstructed view.
MS^{1}_{g}: Room Gallery	Ratio	The maximum percentage of the gallery space that can be occupied by the upper layout elements.
$MS^{1}_{f:}$ Room Floor	Numeric	Specifies the floor to place the room
$MS^{1}_{gr:}$ Room Group	Numeric	Specifies the group that the room belongs

Table 5. Stair Inputs

Name	Type	Description
STA^{i}_{n}: Stair Name	Text	Name of the stair for the user to check generated layout solutions.
STA^{1}_{fmax}: Maximum Flight Width	Dimension	Maximum dimension for the flight width.
STA^{1}_{fmin}: Minimum Flight Width	Dimension	Minimum dimension for the flight width.
STA^{1}_{ld}: Landing depth	Dimension	The horizontal dimension for stair landing.
STA^{1}_{amax}: Maximum Flight Angle	Degree	The maximum angle between the ground and stair flight.
STA^{1}_{amin}: Minimum Flight Angle	Degree	The minimum angle between the ground and stair flight.
STA^{i}_{fs}: Starting floor	Number	The lowest floor that stair contacts.
STA^{1}_{fe}: Ending floor	Number	The highest floor that stair contacts.

Table 6. Other Specific Inputs

Name	Type	Description
CHI^{i}_{min}: Minimum edge	Dimension	Minimum horizontal dimension for chimney.
CHI^{i}_{max}: Maximum edge	Dimension	Maximum horizontal dimension for chimney.
Ga_{c}: Number of cars	Number	Number of cars to be parked within the garage.
G_{e}: Length of entrance	Dimension	Width of the garage door.
Po^{i}_{a}: Porch area	Dimension	Area requirement for porch.
Po^{i}_{min}: Porch minimum edge	Dimension	The minimum dimension value for a separate edge of porch geometry.
Po^{i}_{max}: Porch maximum edge	Dimension	The maximum dimension value for a separate edge of porch geometry.
Pa^{i}_{a}: Patio area	Dimension	Area requirement for patio.
Pa^{i}_{min}: Patio minimum edge	Dimension	The minimum dimension value for a separate edge of patio geometry.
Pa^{i}_{max}: Patio maximum edge	Dimension	The maximum dimension value for a separate edge of patio geometry.

Evaluation Inputs

Ho-Gen allows the user to prioritize the importance of certain soft criteria over the other. In this way, user defines the dominance of selected evaluation algorithms by allocating different coefficient values to the related criteria. It is also possible to disqualify a criterion totally from the search process by defining its coefficient as zero. Detailed information on the constraints can be found in **Section 3.3**.

Evolutionary Inputs

Table 7. Evolutionary Inputs

Name	Type	Description
E_p: Population Size	Number	Specifies the number of layout solutions in a generation.
E_m: Mutation Rate	Coefficient [0,1]	Chance of mutation for a gene after every crossover.
E_{mi}: Mutation Rate Increase	Value	Increase in the mutation rate in the case of stagnation in successive generations.
E_s: Crossover Rate	Coefficient [0,1]	Rate of population to be generated with mating.

Termination Inputs

Table 8. Termination Inputs

Name	Type	Description
T_t: Time limit	Minutes	Runtime limit.
T_{ml}: Maximum generation	Number	Number of generations for each run.
T_{mr}: Stagnation generation	Number	Number of generations to go after stagnation.

3.2 Generation & Guidance

3.2.1 Genotype and phenotype representation

Main Spaces

Genotype

MS^i_{cx}: Coordinate X - MS^i_{cy}: Coordinate Y - MS^i_{dx}: Dimension X – MS^i_{dy}: DimensionY

Phenotype

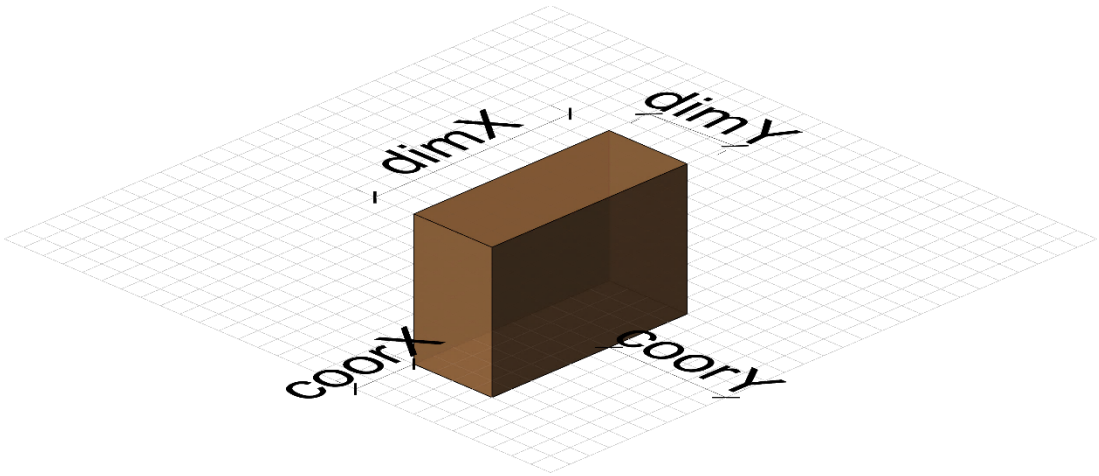


Figure 20. Main Space Phenotype. (Drawn by the author)

Stair

Genotype

STA_r: Rotation – STA_{cx}: Coordinate X - STA_{cy}: Coordinate Y – STA_{fw}: Flight Width - STA_a: Flight Angle

Phenotype

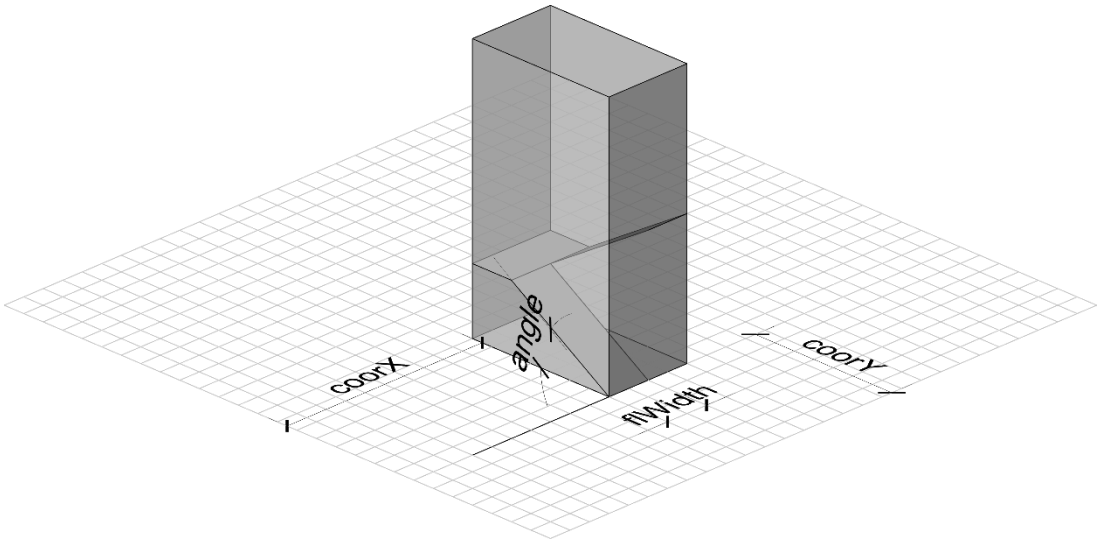


Figure 21. Stair Phenotype. (Drawn by the author)

Chimney

Genotype

CHI^{i}_{cx}: Coordinate X - CHI^{i}_{cy}: Coordinate Y

Phenotype

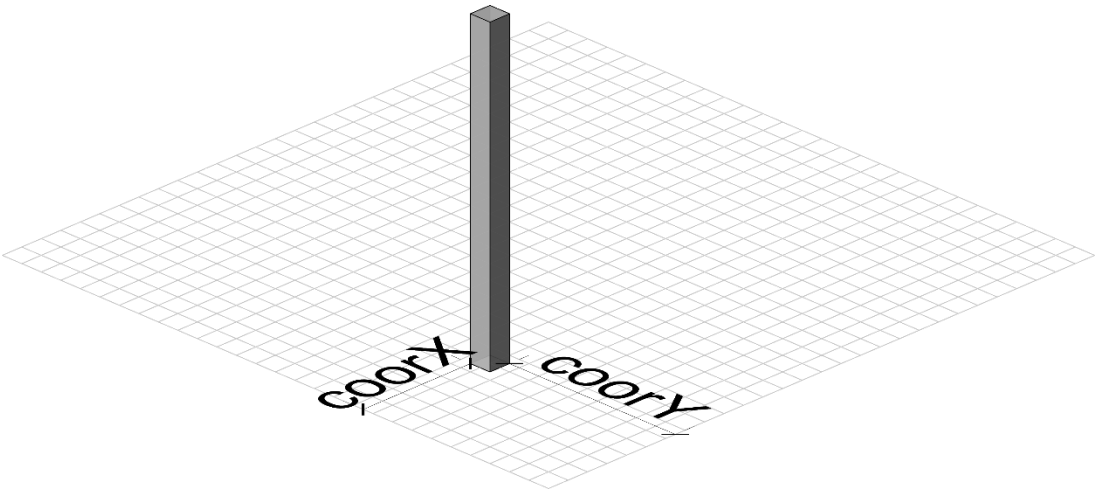


Figure 22. Chimney Phenotype. (Drawn by the author)

Garage

Genotype

G^i_{cx}: Coordinate X - G^i_{cy}: Coordinate Y

Phenotype

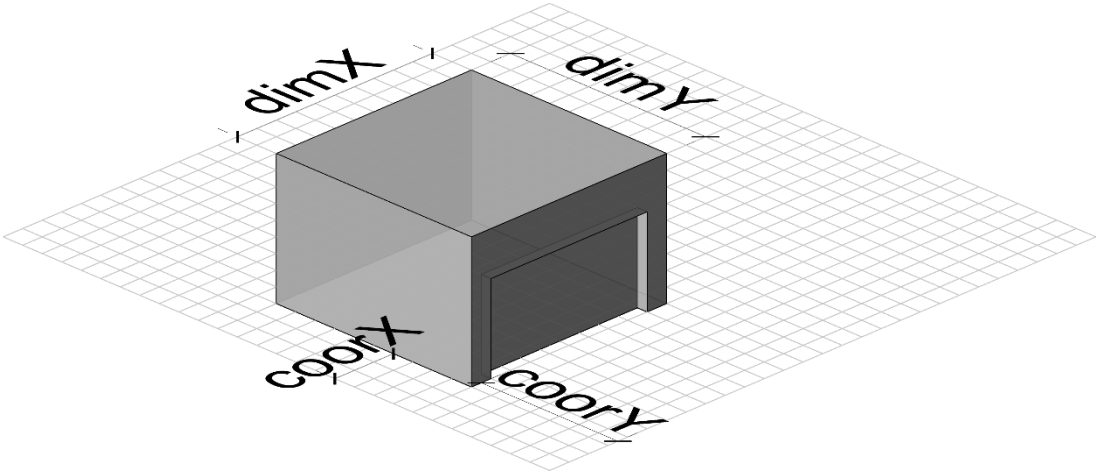


Figure 23. Garage Phenotype. (Drawn by the author)

Porch

Genotype

Po^{i}_{cx}: Coordinate X - Po^{i}_{cy}: Coordinate Y - Po^{i}_{dx}: Dimension X - Po^{i}_{dy}: Dimension Y

Phenotype

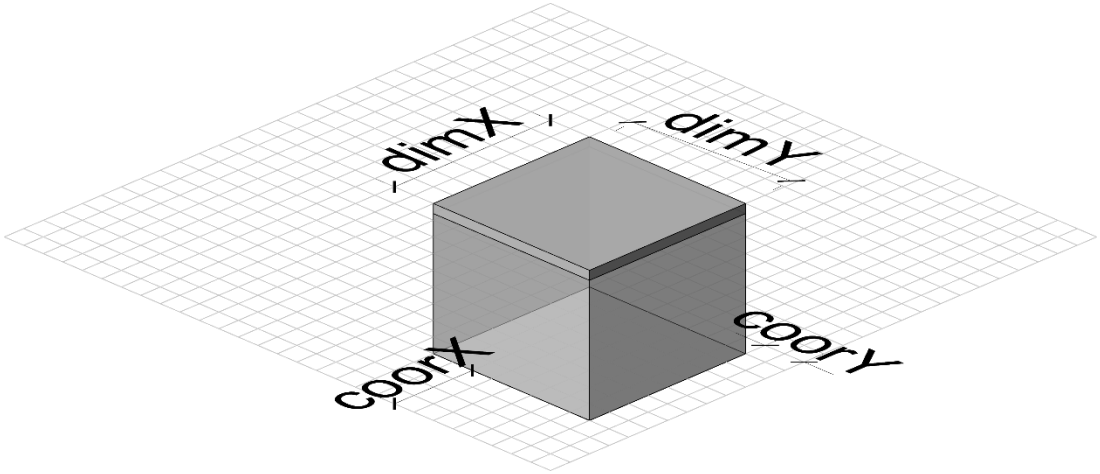


Figure 24. Porch Phenotype. (Drawn by the author)

Patio

Genotype

Pa^{i}_{cx}: Coordinate X - Pa^{i}_{cy}: Coordinate Y - Pa^{i}_{dx}: Dimension X - Pa^{i}_{dy}: Dimension Y

Phenotype

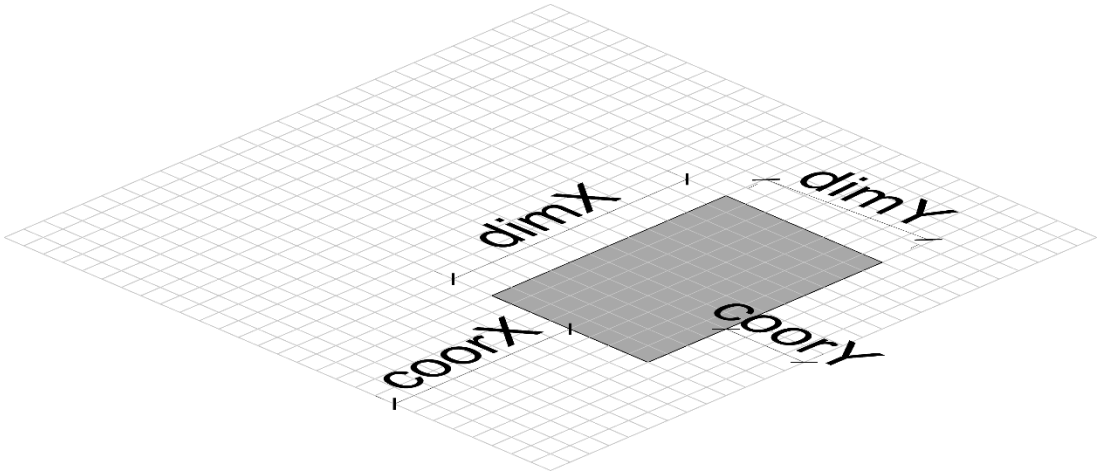


Figure 25. Patio Phenotype. (Drawn by the author)

3.2.2 Genetic Algorithms

Initiation

Initiation generates several individual solutions according to the user-defined population size. In a non-interactive run, an initial population is randomly generated. Random operations are realized within the bounds of initial problem representation. In an interactive run where the user supplies an initial layout solution, many individuals that are relative to a user-defined ratio are generated from the mutations on the initial layout. The rest of the population is again generated randomly.

Selection

In every generation, a portion of population is selected for reproduction to generate the new generation of solutions. Layouts for reproduction are selected according to their fitness score. Ho-Gen's reproduction algorithm requires two solutions in every generation. This selection process does not directly take the best solutions in the population as this process can cause an early reduction in the variety of population. Ho-Gen, alternatively, uses tournament selection which randomly picks a group of four solutions from the population and selects the best individual among this small group as the first parent solution. The selection of the second parent repeats the same process, but this time random selection is realized within a population without first

parent solution. First parent solution is temporarily removed from the population to eliminate the risk of self-mating that produces duplicate layouts within the total generation. Ho-Gen does not create a new population in every generation as this can cause fit individuals to get lost in the reproduction process. Instead, reproduction process uses Elitism, which saves 10% of the fittest individuals directly to the new population.

Crossover

Crossover algorithm is the first part of the reproduction process. Crossover generates a child from the random genotype combination of two parent solutions. This new individual carries the properties of its parents however, it still carries a certain level of difference which increases the exploration space of the algorithm. Ho-Gen uses uniform crossover which makes a random decision for every gene to decide about its source. This way, the genotype order of a parent is not purposely carried to the children. Ho-Gen also favors the fitter parent in gene distribution. In this way, child solution takes 70% percent of its genes from the fitter parent. The crossover operation is repeated until reaching the population limit.

Mutation

Mutation algorithm takes the newly generated solution and performs random changes in their genotype. A random operation between [0,1] is realized for every gene and the gene is randomized if it is below E_m. This simple operation is essential for keeping a level of genetic variety within the population as it avoids early convergence. As an initial condition, Ho-Gen starts with an E_m of %1 but this rate changes during the run. Ho-Gen checks the best fitness score within every generation and compares this value to the previous one. If two scores are the same, then E_m is increased by %0.01.

Termination

Ho-Gen continues to produce new generations until meeting one of the termination criteria. These criteria are defined in Table 8. In the termination process, Ho-Gen

generates the best individual layout of the last generation in Rhinoceros together with the related fitness graphs. In this context, the user can pass into the user guidance process to modify the generated solution and the problem definition or he/she also can start a new Ho-Gen run from the scratch.

3.2.3 User Guidance

Ho-Gen's interactive engine allows designer to iterate over the results through modifying the problem definition and reshaping the solutions. The generated solutions are baked in Rhinoceros by the Grasshopper definition. These baked geometries are also defined in the Grasshopper definition to keep the algorithm informed of the user modifications in Rhinoceros' interface. Grasshopper simultaneously translates the geometric definition in Rhinoceros to its genotype definition in GA. The user can also make changes on the problem definition by using Grasshopper similar to the initial state. This data is also simultaneously translated for GA. The user can make the following modifications in the intermediate states:

- Changing the components of the design problem. For example, changing the area requirement of one LE, removing or adding an LE, changing topological requirements, changing general inputs, or modifying the evaluator weights.

- Adding or removing a LE

- Modifying general inputs

- Modifying topological criteria

- Modifying evaluator weights

- Manual adjustments on the solutions via Rhinoceros to change the scale and location of LEs. From another perspective, the user can also start with an initial layout to guide the search process from the start.

In the end of user guidance phase, GA starts from the initialization phase again. This time it derives the population from the given layout input. The mechanism is explained in detail within the initiation phase of genetic algorithm section

3.3 Evaluation

Ho-Gen evaluates the solutions by a single fitness function that corresponds to the weighted sum of eight evaluator penalties (**Table 9)**. If a solution cannot meet certain requirements of evaluators, a penalty score that is relative to its degree of violation is assigned to the solution. Every evaluator penalty is then multiplied by its own user-defined weight and added to the general fitness function. In this way, fitness score of the solution corresponds to its degree of incompatibility with the initial requirements. All the evaluator penalties except for the C_{dim} and C_{view} correspond to area values. These penalty values are square rooted in order to equalize their effect with the C_{dim} and C_{view} evaluators. The fitness function can be described with the following formula:

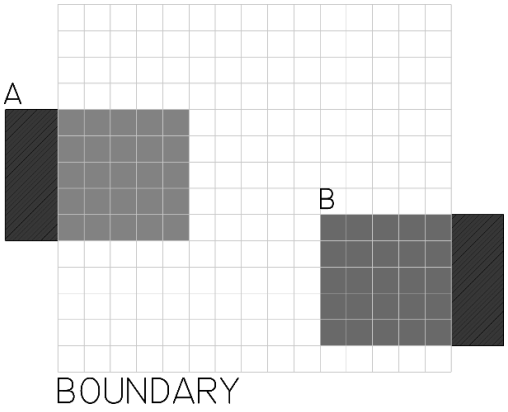
$$C_{total} = (W_{ovf} * \sqrt{C_{ovf}}) + (W_{int} * \sqrt{C_{int}}) + (W_{dim} * \sqrt{C_{dim}}) + (W_{comp} * \sqrt{C_{comp}}) + (W_{cant} * \sqrt{C_{cant}}) + (W_{circ} * \sqrt{C_{circ}}) + (W_{rel} * C_{rel}) + (W_{view} * C_{view})$$

Figure 26. Fitness function equation. (Drawn by the author)

Table 9. Ho-Gen constraints

Constraint Name	**Description**
C_{ovf}: Overflow constraint	Evaluates the solution by the percentage of total area that is out of the user-defined bounding geometry.
C_{int}: Intersection constraint	Evaluates the solution by the area of intersection between separate layout elements.
C_{dim}: Dimension constraint	Evaluates the solution by the difference between AREA input and areas of the generated layout elements.
C_{rel}: Relation constraint	Evaluates the solution by the distance between generated layout elements that are specified as related in the topological inputs.
C_{comp}: Compactness constraint	Evaluates the solution by the difference of arranged group geometries from a bounding rectangle.
C_{cant}: Cantilever constraint	Evaluates the solution by the difference between the given maximum cantilever and the actual cantilever distance on the upper floors.
C_{circ}: Circulation constraint	Evaluates the solution by the area of circulation units, circulation units are tried to be minimized by area with this method
C_{view}: View constraint	Evaluates the solution by the length of interruption by the input side for every layout element with view criteria.

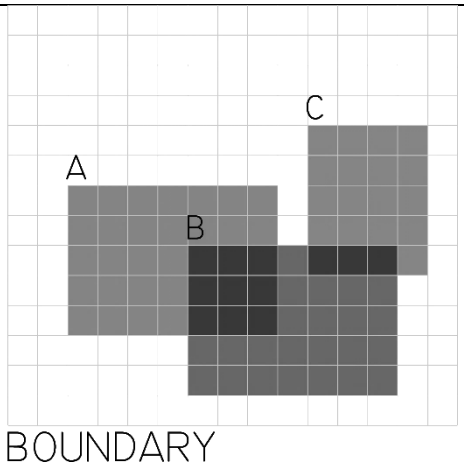
Overflow evaluator checks every floor for LEs that got out of the predefined boundary geometry. It assigns the total area outside the boundary geometry as a penalty.



for each LE:
    interL.**APPEND(**LE)
interL.**APPEND**(Boundary)
Covf = **inter**(interL) - $Boundary_{area}$

Figure 27. Overflow Evaluator (Drawn by the author).

Intersection evaluator checks every floor for overlaps between LEs. The total area of intersection in every floor is assigned as a penalty value



for each FLO:
    interL = [ ]
    totalArea = 0
    for each LE:
        interL.**APPEND(**LE)
        totalArea = totalArea + LE_{area}
interFloor = totalArea - inter(interL)
$C_{int} = C_{int} + interFloor$

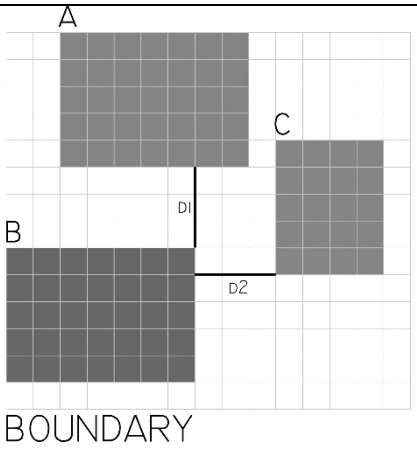
Figure 28. Intersection evaluator (Drawn by the author).

Dimension evaluator checks how close the SP areas are to the initial user inputs. It assigns the absolute difference between the two values as a penalty.

for each SP:
IF $SP_{area} > 0$:
$C_{dim} = C_{dim} + \mathbf{abs}(SP_{area} - (SP_{dimX} * SP_{dimY}))$

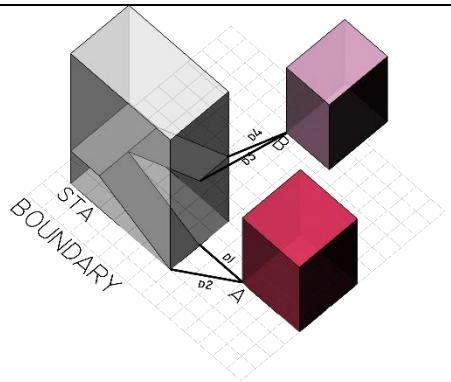
Figure 29. Dimension evaluator (By the author).

Relation evaluator checks how close the distances between LEs are to the initial topological inputs. The distance calculation process varies with the type of LE under consideration. SP distances are taken as the shortest distance between their borders. STA distances are the closest distances between the two corners of both stair flight edges and the border of the relevant SP. CHI distance is the closest distance between the center of CHI geometry and the relevant SP border. Relation distances cannot be negative, so the negative values are replaced by zero. Relation evaluator works relevant to two types of adjacency. 0 corresponds to NO RELATION between LEs so this pair is not evaluated. 1 corresponds to CONNECTION between LEs which requires a certain overlapping between LE borders to place a door. Ho-Gen automatically considers 1 meter as a standard door dimension and gives the penalty of 1 for pair of LEs intersect with point intersection. 2 corresponds to ADJACENCY between LEs which does not require a physical connection but still they require being in close proximity.




```
for x in unitTOPO:
    for y in x:
        distX=abs(SPx.centerX-SPy.centerX)- (SPx.dimX-SPy.dimX)/2
        distY=abs(SPx.centerY-SPy.centerY)- (SPx.dimY-SPy.dimY)/2
        IF y = CONNECTION:
            IF distX=0 AND distY=0:
                interRel = interRel +1
            IF distX<0: distX = 0 IF distY<0: distY
            interRel = interRel + √(distX^2 + distY^2)
        IF y = ADJACENCY:
            IF distX<0: distX = 0 IF distY<0: distY
            interRel = interRel + √(distX^2 + distY^2)
        C_rel = C_rel + interRel
```

Figure 30. Relation evaluator - SP (Drawn by the author).



for x in staTOPO:

for y in x:

$distX_1$=**abs**(STA.p1X-SPy.centerX) - (SPy.dimX)/2

$distY_1$=**abs**(STA.p1Y-SPy.centerY) - (SPy.dimY)/2

$distX_2$=**abs**(STA.p2X-SPy.centerX) - (SPy.dimX)/2

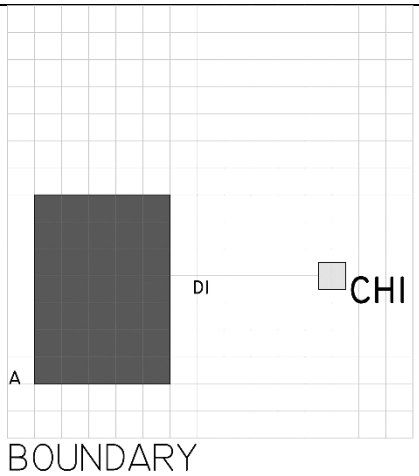
$distY_2$=**abs**(STA.p2Y-SPy.centerY) - (SPy.dimY)/2

IF distX<0: distX = 0 **IF** distY<0: distY

interRel = interRel + $\sqrt{(distX^2 + distY^{2)}}$

$C_{rel} = C_{rel}$ + interRel

Figure 31. Relation evaluator - STA (Drawn by the author).



FOR x in chiTOPO:

$distX_1$=**abs**(CHI.coorX-SPy.centerX) - (SPy.dimX)/2

$distY_1$=**abs**(CHI.coorY-SPy.centerY) - (SPy.dimY)/2

IF distX<0: distX = 0 **IF** distY<0: distY

interRel = interRel + $\sqrt{(distX^2 + distY^{2)}}$

$C_{rel} = C_{rel}$ + interRel

Figure 32. Relation evaluator - CHI (Drawn by the author).

Compactness evaluator checks the irregularity of GRO geometries together with the unoccupied regions within them. The forming process of GROs can be found in

Table 2. For every layout, a minimum rectangular region is generated that contains all SPs in a GRO. Evaluator assigns the difference between the area of bounding rectangular region and the total area of SPs in relative GRO as a penalty. The penalty cannot be below zero so a negative value is replaced by zero.

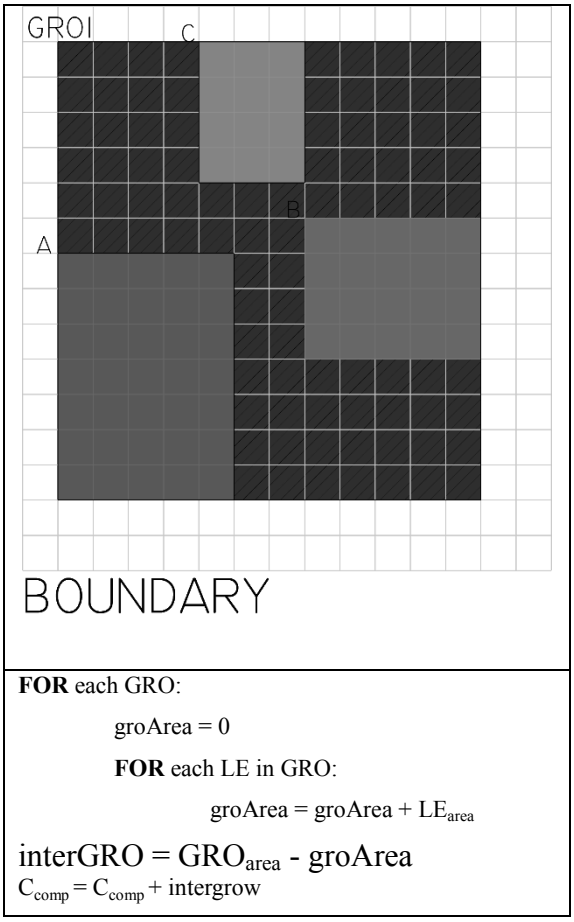
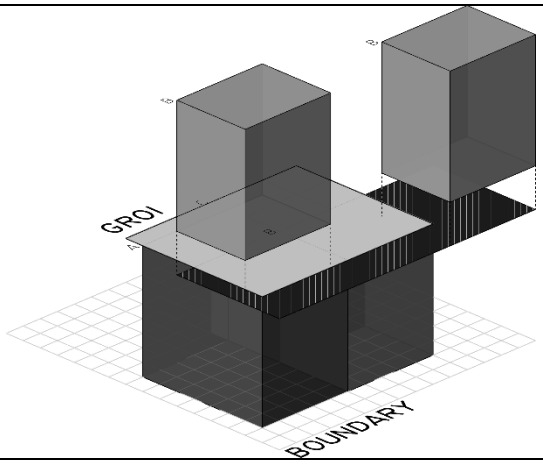


Figure 33. Compactness evaluator (Drawn by the author).

Cantilever evaluator checks the relevance of maximum cantilever distances between succeeding floors to user inputs. Evaluation process repeats for every two sequent floors. Cantilever evaluator considers the GRO geometries rather than LEs separately to reduce the time requirement for calculations. First, cantilever evaluator considers the SPs in separate GROs in the lower floor. These SPs are offset by the maximum cantilever input and a minimum bounding rectangle is generated. Evaluator calculates the union area of bounding rectangles of separate GROs. Second, cantilever evaluator takes into account the SPs in the upper floor and repeats

the process without the offsetting. Area calculation is repeated with all bounding rectangles in the lower and upper floors. The difference between the second and first result is assigned as the penalty. The penalty cannot be negative, so a negative value is replaced by zero.



FOR each FLO:
    FOR each GRO:
        groFloorLow = [ ]
        groFloorHigh = [ ]
        FOR each LE in GRO:
IF LE_{floor} = FLO: groFloorLow.append(LE(LE.coorX- Gi_m, LE.coorY- Gi_m, LE.dimX+ 2*Gi_m, LE.dimY+ 2*Gi_m)
            ELIF LE_{floor} = FLO + 1: groFloorHigh.append(LE)
        interCant = inter(groFloorLow + groFloorHigh) - inter(groFloorLow)
    C_{cant} = C_{cant} + interCant

Figure 34. Cantilever evaluator (Drawn by the author).

Circulation evaluator checks the area of circulation SPs. Circulation SPs are defined with zero area value by the user and their area is tried to be minimized by Ho-Gen. The total area of circulation SPs are given as a penalty.

FOR each SP:
    IF SP_{area} = 0:
        C_{circ} = C_{circ} + (SP_{dimX} * SP_{dimY})

Figure 35. Circulation evaluator (Drawn by the author).

View evaluator checks for obstacles in the given direction for SPs. If an initial view preference exists for one SP, the evaluator draws a rectangle from SPs farthest edge in that direction until the layout boundary. Any other SP that overlaps with the generated view rectangle is given a penalty relevant to the obstacle distance.

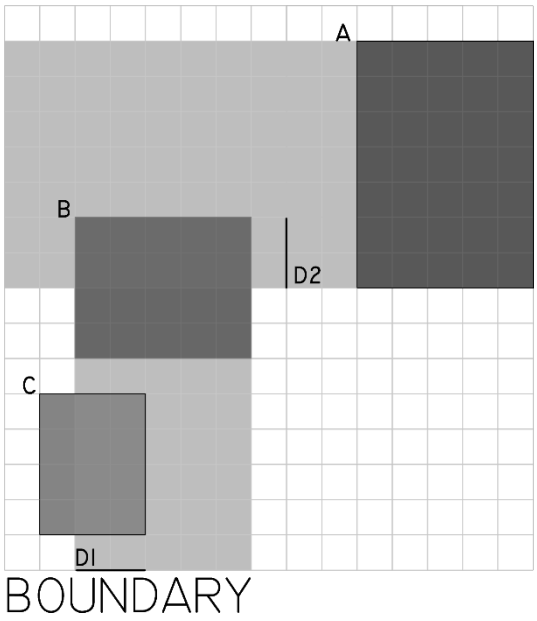


Figure 36. View evaluator (Drawn by the author).

```
FOR each SP:
IF dir = NORTH:
coorX = SP.coorX
                    coorY = SPcoorY + SPdimY
                    dimX = SPdimX
                    dimY = boundaryY - (SPcoorY + (SPdimY)
IF dir = EAST:
coorX = SPcoorX + SPdimX
                    coorY = SPcoorY
                    dimX = boundaryY – (SPcoorX + SPdimX)
                    dimY = SPdimY
IF dir = SOUTH:
coorX = SPcoorX
                    coorY = 0
                    dimX = SPdimX
                    dimY = SPcoorY
IF dir = WEST:
coorX = 0
                    coorY = SPcoorY
                    dimX = SPcoorX
                    dimY = SPdimY
VU = Unit(coorX,coorY,dimX,dimY)
FOR each SP:
distX=abs(SPx.centerX-VU.centerX)- (SPx.dimX-VU.dimX)/2
distY=abs(SPx.centerY-VU.centerY)- (SPx.dimY-VU.dimY)/2
IF distX>0: distX=0, IF distY>0: distY=0
IF dir=SOUTH OR dir=NORTH:
interView = interView - distX
IF dir=EAST OR dir=WEST:
interView = interView - distY
Cview = Cview + interView
```

Figure 37. View evaluator (Drawn by the author).

The assignment of evaluator weights is an essential process for Ho-Gen to generate layout solutions according to user requirements. Users define the relative importance of evaluators through the hierarchy in evaluator weights. Evaluators with considerably higher weight values correspond to hard sub-criteria. Hard sub-criteria are basically the red-lines of the design process. Architects, while in the form-finding process, do not generate every possible configuration such as a layout with

overlapping spaces. Similarly, Ho-Gen limits the possibility of the generation of certain instances from the exploration process. Layouts that violate hard sub-criteria are not killed off, but their penalty values increase.

Design is mostly about dealing with uncertain sub-criteria. Such uncertain sub-criteria are not strictly imposed rules, but their effect still matters for design. These criteria are defined as soft sub-criteria. The relative importance of soft sub-criteria can change with the problem and varying subjective decisions of the user. Such a layout task can require the area dimensions to match the user inputs while for another layout task the regularity of the overall form can be the primary concern.

CHAPTER 4

CASE STUDIES

In this chapter, Ho-Gen's performance in generating valid layouts is tested. Computational formulations to layout design are typically considered as "NP-complete". Therefore, finding optimal layout solutions require extended amounts of time even for small scale problems, as the solution space grows exponentially even with a low number of layout elements.[179] Thus, Ho-Gen is tested with two case studies with different levels of complexity in terms of the number of layout elements, adjacency relations, and user objectives. Every case study is also approached with different group relations and level of compactness in different parts to evaluate the effect of C_{comp} on the character of the layout solutions. In addition to these non-interactive case studies, the effects of user interaction on the generation process are tested in a separate part for every case study.

The main algorithm of HO-Gen is implemented in Rhino Python, while Grasshopper is used for the initial and intermediate states of user interaction. The user can also use Rhinoceros's main drafting interface to modify the generated layouts manually. Tests were conducted in 2017 by a 2.13 Core Duo computer with 4GB DDR Ram.

The non-interactive parts of the case studies are conducted to evaluate the success of Ho-Gen in generating alternative layout solutions for the same problem. Thus, the research presents six alternatives for every case study to observe the level of difference between the generated solutions. Non-interactive case studies also present intermediate phases from the generation process of the best layout alternative to

[179] Jo and Gero, "Space Layout Planning Using an Evolutionary Approach," 3.

show the gradual development of layouts. Table 4-1 shows the general aims of case studies.

Table 10 Case Study Table

<table>
<tr><th>CASE STUDY</th><th>INTERACTIVITY</th><th>COMPACTNESS</th><th>TARGET</th></tr>
<tr><td rowspan="2">1</td><td>AUTOMATED</td><td>D – Layout</td><td>-Divergence of results
-Effect of evaluator penalties</td></tr>
<tr><td>INTERACTIVE</td><td>D and B</td><td>User interaction mechanism</td></tr>
<tr><td rowspan="3">2</td><td rowspan="2">AUTOMATED</td><td>A – No groups</td><td rowspan="3">-Effect of increased complexity
-Divergence of results
-Effect of group relations
-Effect of evaluator penalties</td></tr>
<tr><td>C - Floor
D - Layout</td></tr>
<tr><td>INTERACTIVE</td><td>C - Floor</td><td>-Convergence of an initial layout sketch</td></tr>
</table>

The results of the case studies are presented together with the fitness graphs showing the development of the layout through generations. This graph does not show the individual's direct penalty score. Ho-Gen's evaluators work within different numerical ranges because of the magnitude of results. Thus, it is not directly possible to compare the differences in the evaluator penalties. The penalty scores are normalized within [0, 1] according to the following formula to make such comparison possible:

$$C_{normal} = C_{initial} / C_{maximum}$$

Figure 38. Formula to normalize constraint scores (Drawn by the author).

4.1 Case study inputs

4.1.1 Case study 1 – Small scale

First case study is conducted to test the capabilities of Ho-Gen in generating a diversity of solutions for a simple 2D layout problem. Layout problem consists of a 2-bedroom SFH in a single floor. The main entrance is arranged from the NORTH direction. A 10 m * 10 m rectangle is given as a boundary. Detailed information about the inputs is as follows:

Table 11. Main Space Inputs (Case Study 1)

Main Space / Input	MS_f	MS_{min}	MS_{ar}	MS_{max}	MS_{at}	MS_v	MS_g
MS^1 Living Room	0	3	25	8	-	-	1
MS^2 Kitchen	0	2.5	10	-	-	-	1
MS^3 Master Bedroom	0	2.5	18	5	-	-	2
MS^4 Bedroom 1	0	2.5	12	-	-	-	2
MS^5 Bathroom 1	0	1.5	7	-	-	-	2
MS^6 Entrance	0	1.5	8	8	-	1	0
MS^7 Circulation 1	0	1	-	8	-	-	2

Table 12. Patio Inputs (Case Study 1)

Patio / Input	Pa_{min}	Pa_{ar}	Pa_{max}	Pa_v	Pa_g
Pa^1 Patio 1	3	25	8	-	1

Table 13. Porch Inputs (Case Study 1)

Porch / Input	Po_{min}	Po_{ar}	Po_{max}	Po_v	Po_g
Po^1 Porch 1	3	25	8	1	0

Table 14. Garage Inputs (Case Study 1)

Garage / Input	Ga_c	Ga_v	Pa_g
Ga^1 Garage 1	1	1	0

Table 15. Space Adjacency Matrix (Case Study 1)

	MS^1	MS^2	MS^3	MS^4	MS^5	MS^6	MS^7	Pa^1	Po^1	Ga^1
MS^1		1	0	0	0	1	1	1	0	0
MS^2			0	0	0	0	0	0	0	0
MS^3				0	2	0	1	0	0	0
MS^4					2	0	1	0	0	0
MS^5						0	1	0	0	0
MS^6							0	0	1	1
MS^7								0	0	0
Pa^1									0	0
Po^1										0
Ga^1										

Table 16. Chimney Adjacency Matrix (Case Study 1)

	MS^1	MS^2	MS^3	MS^4	MS^5	MS^6	MS^7	Pa^1	Po^1	Ga^1
CHI^1	0	1	0	0	0	0	0	0	0	0

4.1.2 Case study 2 – Medium scale

Second case study is conducted in order to test Ho-Gen's capability to deal with

multi-floor layout problems. Multiple floors create a higher complexity for Ho-Gen because every floor is solved as a separate layout problem. This also reflects to the time requirements because Ho-Gen evaluates every floor separately. Another reason is the increase in the population in order to cope with the larger search space.

Layout problem consists of a 3-bedroom SFH in two floors. The main entrance is arranged from the NORTH direction. A 15 m X 15 m rectangle is given as a boundary. Detailed information about the inputs is as follows:

Table 17. Main Space Inputs (Case Study 2)

Main Space / Input	MS_f	MS_{min}	MS_{ar}	MS_{max}	MS_{at}	MS_v	MS_g
MS^1 Living Room	0	3	25	8	-	3	0
MS^2 Kitchen	0	2.5	10	-	-	-	0
MS^3 Master Bedroom	1	2.5	18	5	-	2	1
MS^4 Bedroom 1	0	2.5	12	-	-	-	0
MS^5 Bedroom 2	1	2.5	12	-	-	-	1
MS^6 Bathroom 1	0	1.5	7	-	-	-	0
MS^7 Bathroom 2	1	1.5	7	-	-	-	1
MS^8 Bathroom 3	1	1.5	7	-	-	-	1
MS^9 Entrance	0	1.5	8	8	-	1	0
MS^{10} Circulation 1	0	1	-	8	-	-	0
MS^{11} Circulation 2	1	1	-	8	-	-	1

Table 18. Patio Inputs (Case Study 2)

Patio / Input	Pa_{min}	Pa_{ar}	Pa_{max}	Pa_v	Pa_g
Pa^1 Patio 01	3	20	6	-	0

Table 19. Porch Inputs (Case Study 2)

Porch / Input	Po_{min}	Po_{ar}	Po_{max}	Po_v	Po_g
Po^1 Porch 01	3	10	4	1	0
Po^2 Porch 02	3	10	4	2	0

Table 20. Garage Inputs (Case Study 2)

Garage / Input	Ga_c	Ga_v	Pa_g
Ga^1 Garage 01	2	1	0

Table 21. Space Adjacency Matrix (Case Study 2)

	MS^1	MS^2	MS^3	MS^4	MS^5	MS^6	MS^7	MS^8	MS^9	MS^{10}	MS^{11}	Pa^1	Po^1	Po^2	Ga^1
MS^1		1	0	0	0	0	0	0	1	1	0	1	0	0	0
MS^2			0	0	0	0	0	0	0	0	0	0	0	1	0
MS^3				0	0	0	0	1	0	0	1	0	0	0	0
MS^4					0	0	2	0	0	1	0	0	0	0	0
MS^5						0	2	0	0	0	1	0	0	0	0
MS^6							0	0	0	1	0	0	0	0	0
MS^7								0	0	0	1	0	0	0	0
MS^8									0	0	0	0	0	0	0
MS^9										0	0	0	1	0	1
MS^{10}											0	0	0	0	0
MS^{11}												0	0	0	0
Pa^1													0	0	0
Po^1														0	0
Po^2															0
Ga^1															

Table 22. Stair Adjacency Matrix (Case Study 2)

	MS^1	MS^2	MS^3	MS^4	MS^5	MS^6	MS^7	MS^8	MS^9	MS^{10}	MS^{11}	Pa^1	Po^1	Po^2	Ga^1
STA1-a	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
STA1-b	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Table 23. Chimney Adjacency Matrix (Case Study 2)

	MS^1	MS^2	MS^3	MS^4	MS^5	MS^6	MS^7	MS^8	MS^9	MS^{10}	MS^{11}	Pa^1	Po^1	Po^2	Ga^1
CHI^1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

4.2 Case study results

4.2.1 Case study 1a

In this case study, the building program is tested as a one whole GRO, which corresponds to a compact rectangular result. The weights of the evaluators C_{ovf} to C_{view} are 3-8-3-2-3-0-0-1. Ho-Gen is run for 6 times with the given inputs to test the validity and formal variation of the generated results. During each run, Ho-Gen bakes the fittest member in every 10 generations. The figures that explain the development process of the fittest alternative in included in APPENDIX A. The generated results of six runs are as follows:

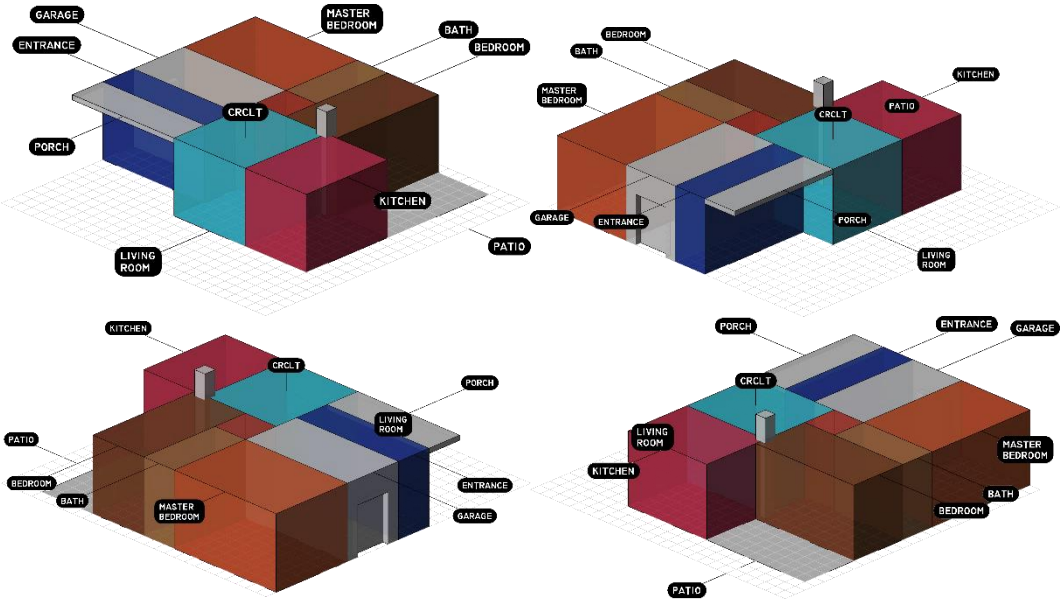


Figure 39. Alternative 1 - Case study 1 – Compactness D - Parallel projection from 4 sides. (Drawn by the author)

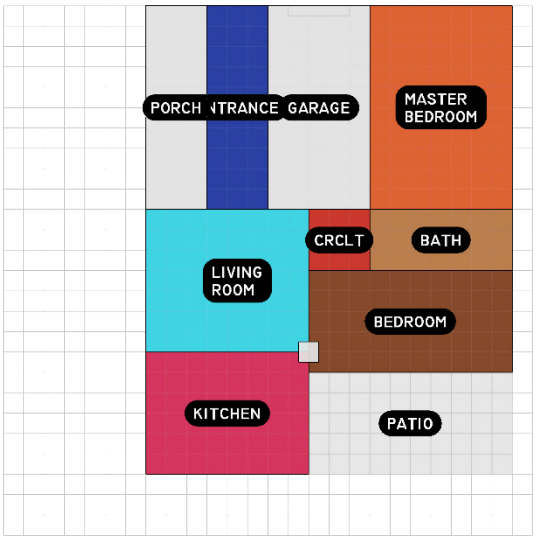


Figure 40. Alternative 1 - Case study 1 – Compactness D - Top view. (Drawn by the author)

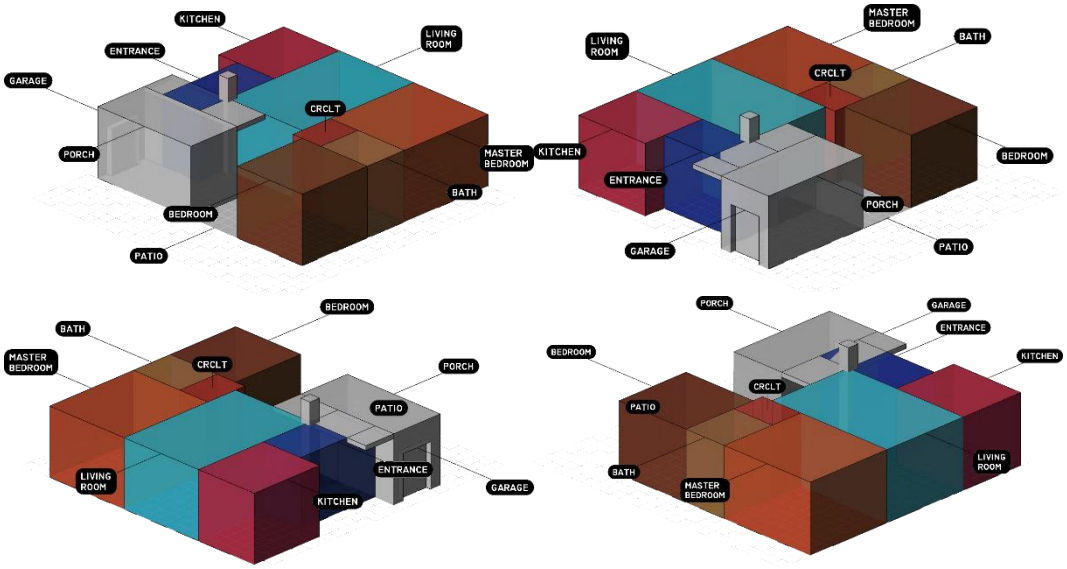


Figure 41. Alternative 2 - Case study 1 Compactness D - Parallel projection from 4 sides. (Drawn by the author)

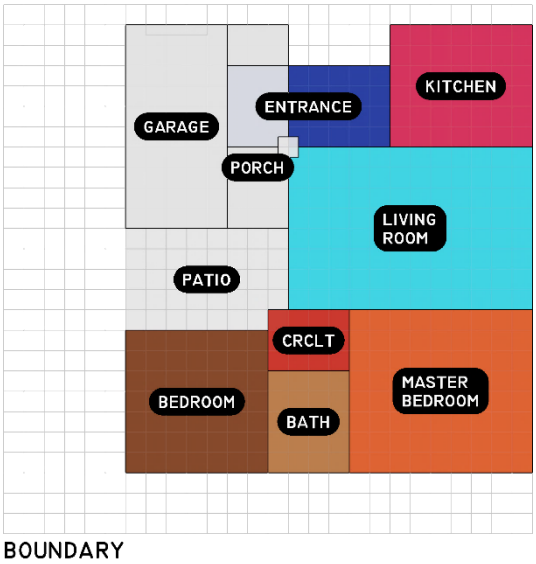


Figure 42. Alternative 2 - Case study 1 – Compactness D - Top view. (Drawn by the author)

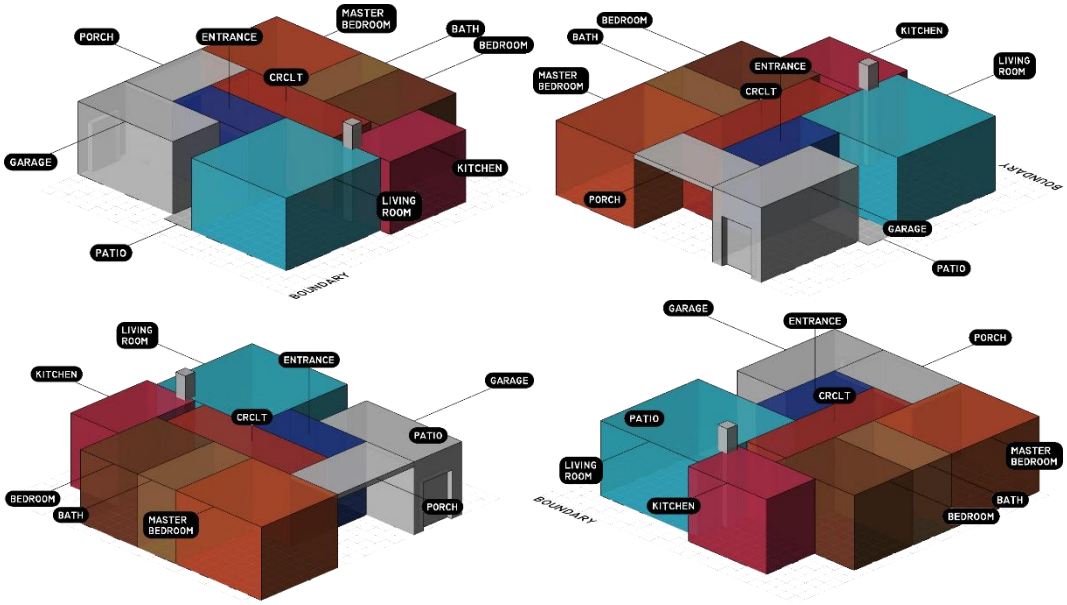


Figure 43. Alternative 3 - Case study 1 – Compactness D - Parallel projection from 4 sides. (Drawn by the author)

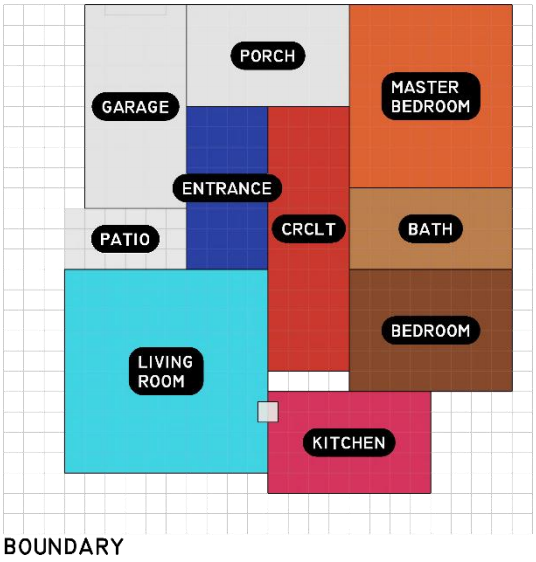


Figure 44. Alternative 3 - Case study 1 – Compactness D - Top view. (Drawn by the author)

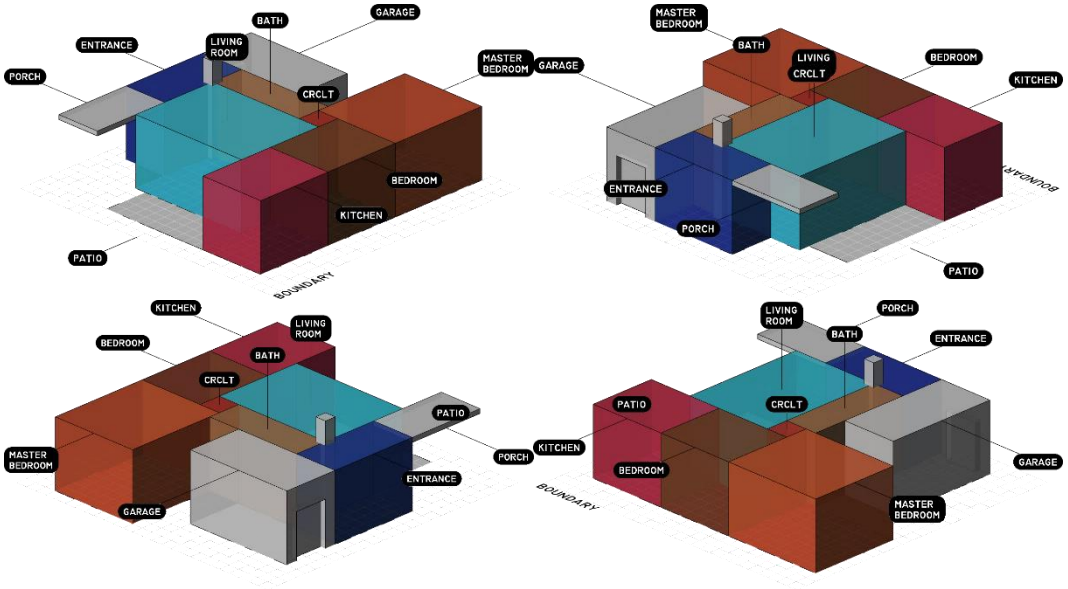


Figure 45. Alternative 4 - Case study 1 – Compactness D - Parallel projection from 4 sides. (Drawn by the author)

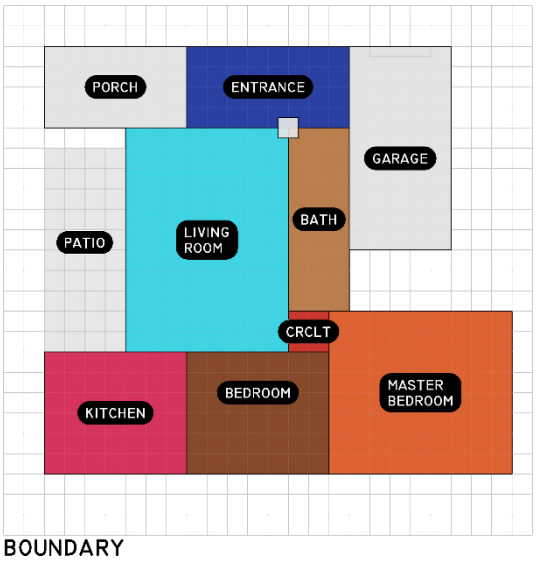


Figure 46. Alternative 4 - Case study 1 – Compactness D - Top view. (Drawn by the author)

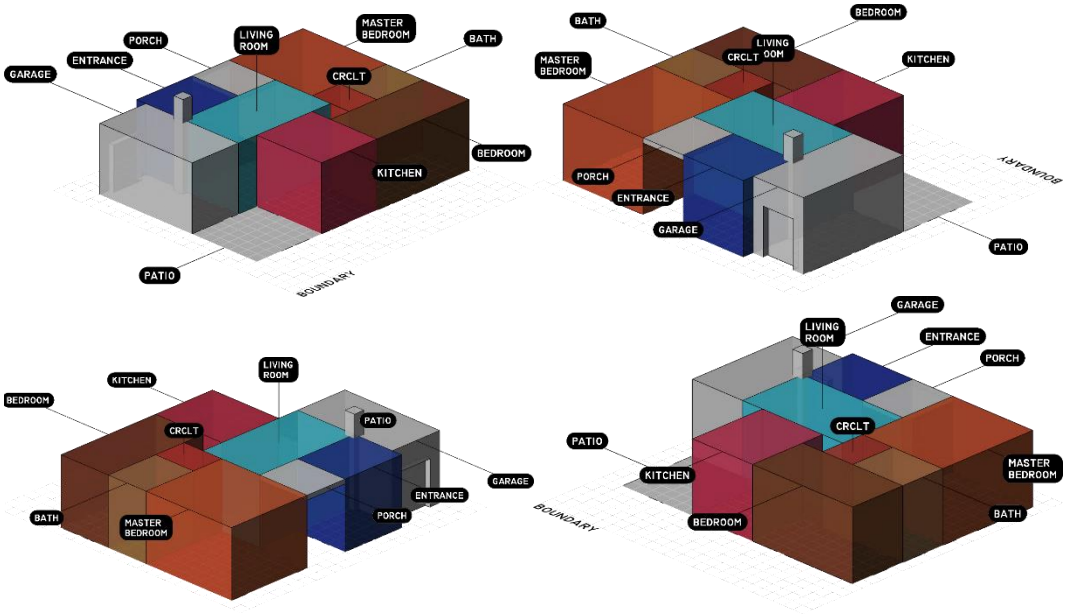


Figure 47. Alternative 5 - Case study 1 – Compactness D- Parallel projection from 4 sides. (Drawn by the author)

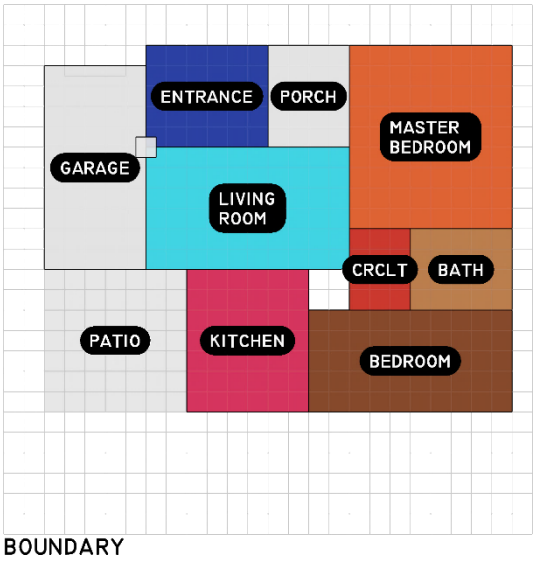


Figure 48. Alternative 5 - Case study 1 – Compactness D - Top view. (Drawn by the author)

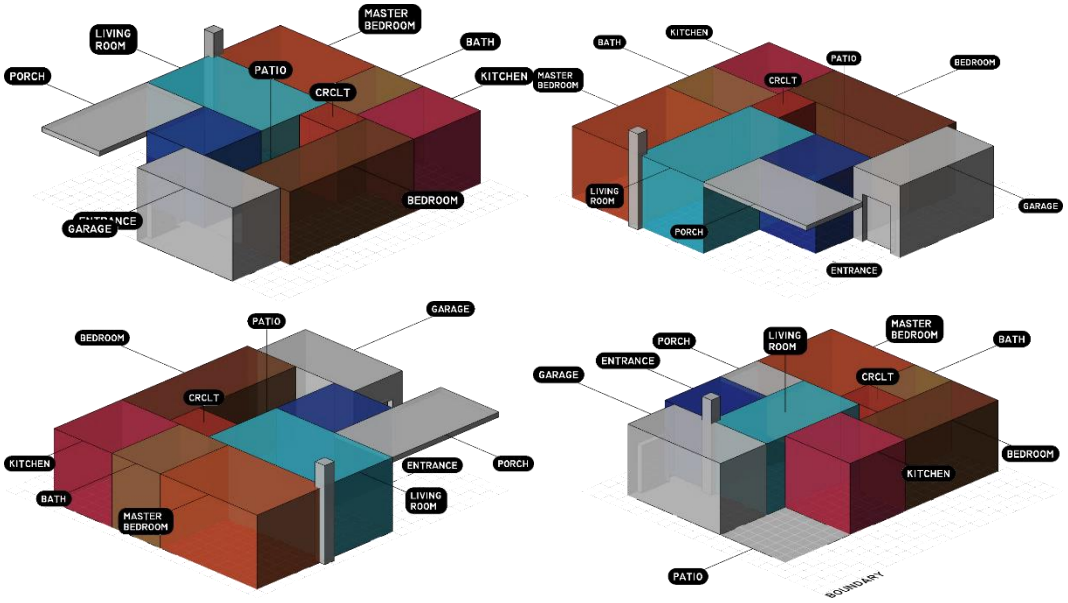


Figure 49. Alternative 6 - Case study 1 – Compactness D - Parallel projection from 4 sides. (Drawn by the author)

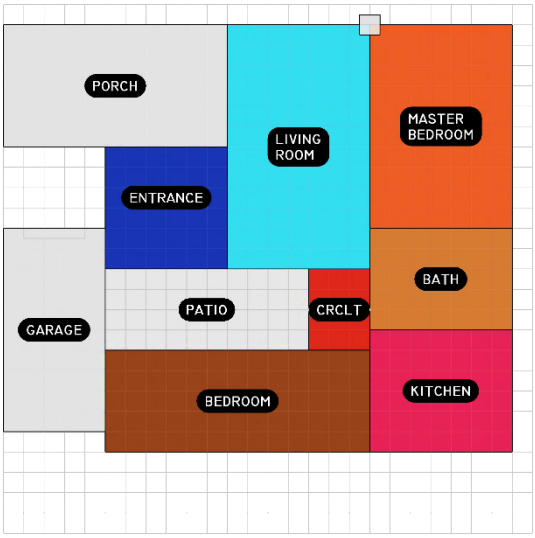


Figure 50. Alternative 6 - Case study 1 – Compactness D - Top view. (Drawn by the author)

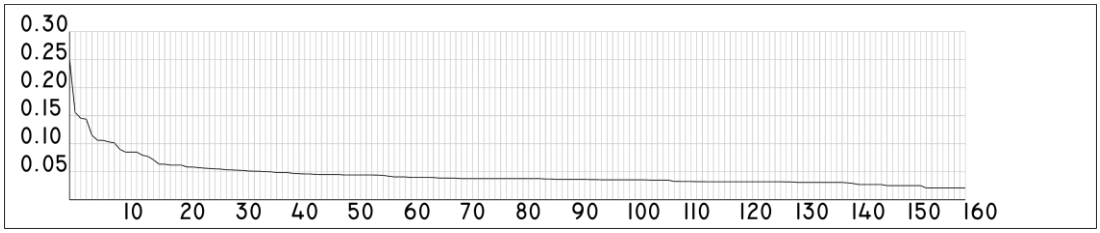


Figure 51. Best total fitness score - Case study 1 – Compactness D. (Drawn by the author)

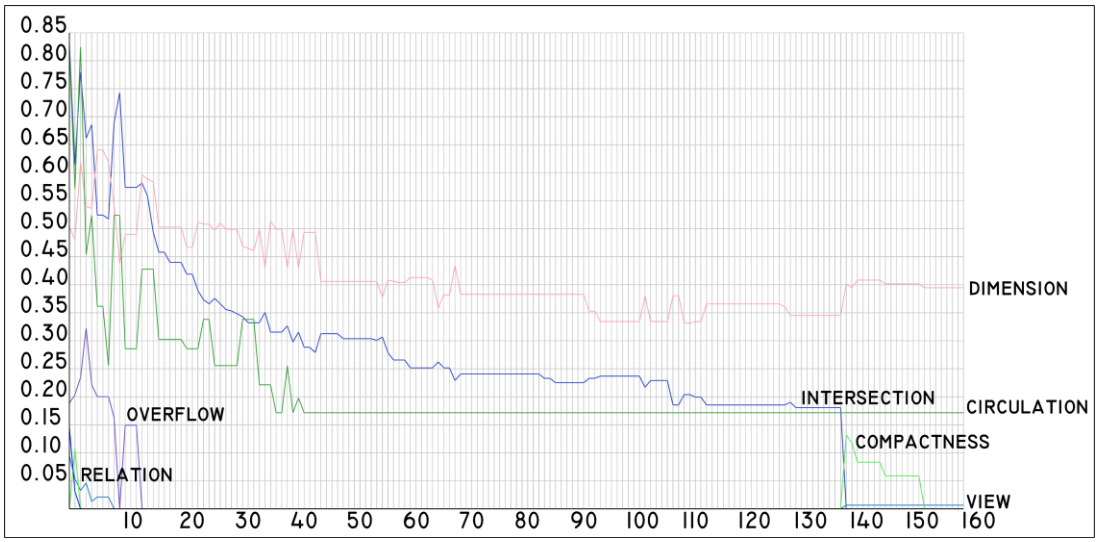


Figure 52. Best evaluator fitness score - Case study 1 –Compactness D. (Drawn by the author)

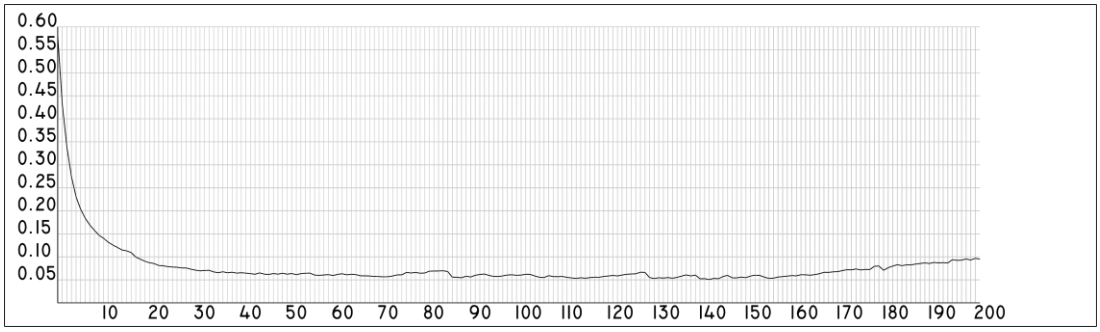


Figure 53. Average total fitness score - Case study 1 – Compactness D. (Drawn by the author)

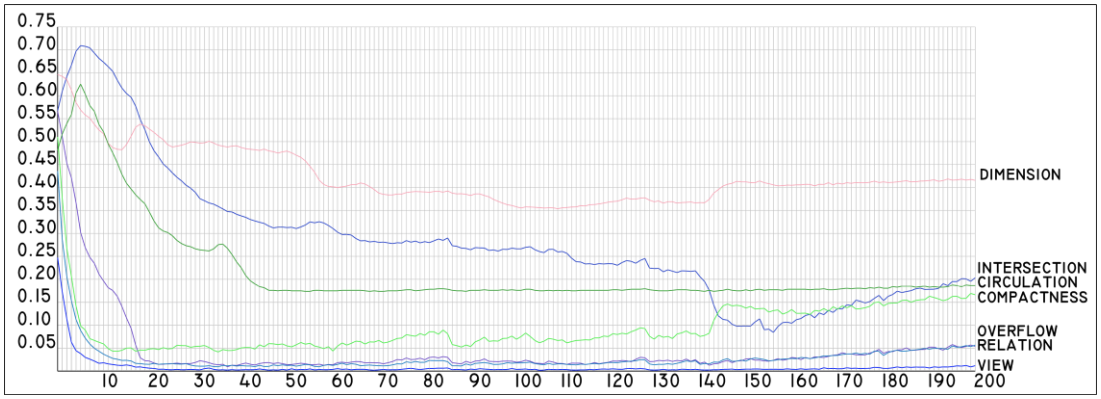


Figure 54. Average evaluator fitness score - Case study 1 – Compactness D. (Drawn by the author)

Table 24. Weighted fitness results for six alternatives - Case study 1 – Compactness D (Drawn by the author).

	Fitness	C_{ovf}	C_{int}	C_{dim}	C_{rel}	C_{comp}	C_{cant}	C_{circ}	C_{view}
1	0.023	0.000	0.000	0.391	0.011	0.000	0.000	0.172	0.000
2	0.044	0.000	0.000	0.234	0.000	0.000	0.000	0.244	0.000
3	0.047	0.000	0.000	0.212	0.022	0.236	0.000	0.402	0.000
4	0.033	0.000	0.000	0.027	0.018	0.256	0.000	0.118	0.000
5	0.024	0.000	0.000	0.391	0.000	0.106	0.000	0.198	0.000
6	0.042	0.000	0.000	0.255	0.000	0.000	0.000	0.150	0.000

A closer look into the generated alternatives and the evaluation scores indicate that Ho-Gen's automated run is successful in generating valid alternative solutions for layouts with high compactness. The variety in the generated solutions is found successful in terms of the placement of LEs and overall layout form. Alternative 1 (Figure 40) and Alternative 5 (Figure 48), as the first best and second best solutions, are different in their overall form and orientation. An interesting result of this study was the C_{rel} penalty in the Alternative 1. Normally C_{rel} is a hard constraint and its violation should result with an overall bad fitness score. The reason for the C_{rel} penalty in Alternative 1 can be observed in the best evaluator fitness score graph (Figure 52). According to the graph, a sudden decrease in C_{int} score caused a small increase in C_{rel}. Indeed, alternative 1 is found very organized and regular compared to the other solutions. In a way, this result verifies the suitability of the fitness function. Another interesting point is the emergent inner courtyard in Alternative 6 (Figure 50). The inner courtyard was not hard coded within the topologic description

of the layout, however, the description was open enough to include such a result within the solution space.

An examination of the average fitness score graph (Figure 53) shows small triangular differences. These hill-like figures indicate the points of stagnation in the search process. As mentioned in tool development, stagnation causes an increase in the mutation rate. Such an increase causes Ho-Gen to explore different solution options which usually causes the generation of many bad layouts. Therefore, it is possible to say that the search process has not come across such a long stagnation. Instead, Ho-Gen run encountered many quick improvements.

4.2.2 Case study 1b

In the interactive scenario, the user was expected to have less information about the configurational possibilities of the layout problem. Thus, the user has not arranged any separate groups within the layout at the start. The problem definition is changed as all LEs form one group. The weights of the evaluators C_{ovf} to C_{view} are 3-8-3-2-3-0-0-1.

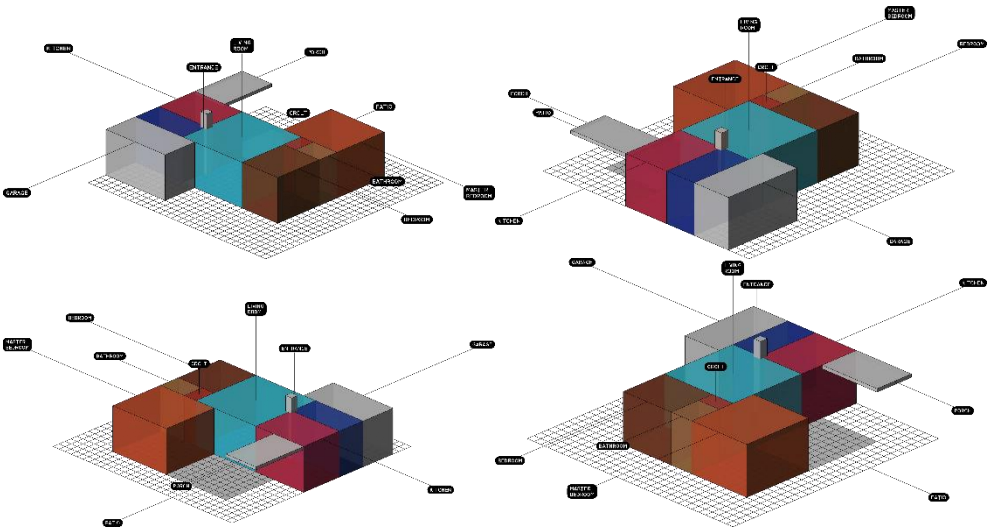


Figure 55. The results of Iteration 1 - Parallel projection from four sides - Case study 1– Interactive run. (Drawn by the author)

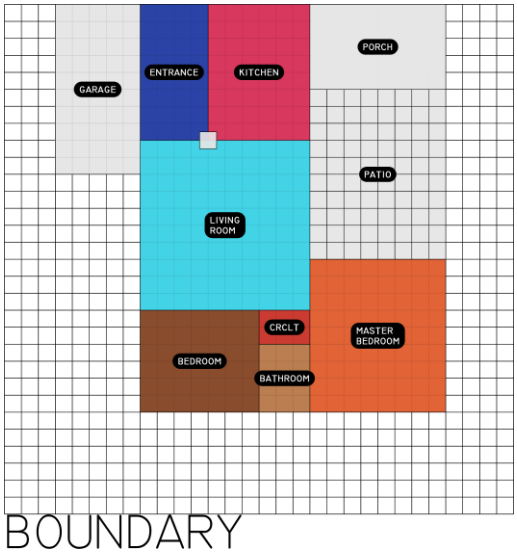


Figure 56. The results of Iteration 1 – Top view - Case study 1 – Interactive run. (Drawn by the author)

The user, after observing the results of the initial iteration, finds the bathroom small and decides that the required bathroom area is not possible with the current compactness arrangement. Therefore, the user separates the bathroom, corridor, bedroom, and master bedroom through defining them in a new group. The user also provides an initial layout to Ho-Gen by making certain arrangements in the current iteration.

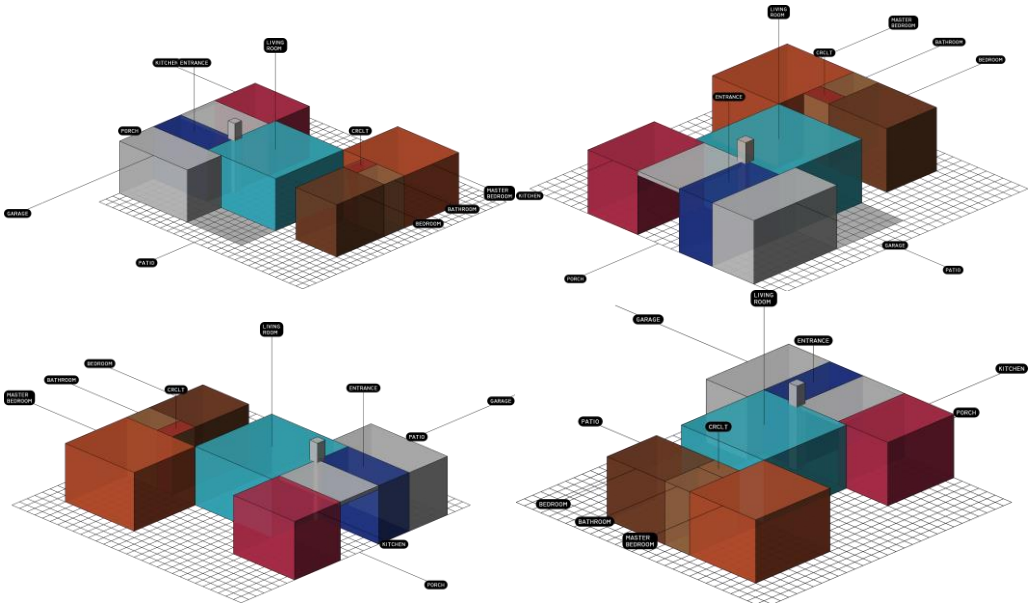


Figure 57. Given layout arrangement for Iteration 2 - Parallel projection from four sides - Case study 1 – Interactive run. (Drawn by the author)

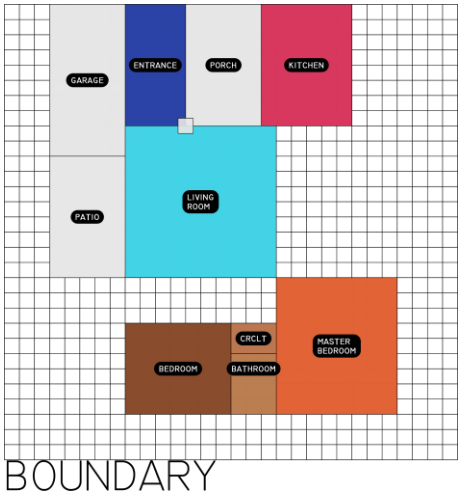


Figure 58. Given layout arrangement for Iteration 2 – Top View - Case study 1 – Interactive run. (Drawn by the author)

Ho-Gen generated the following layout through 43 generations:

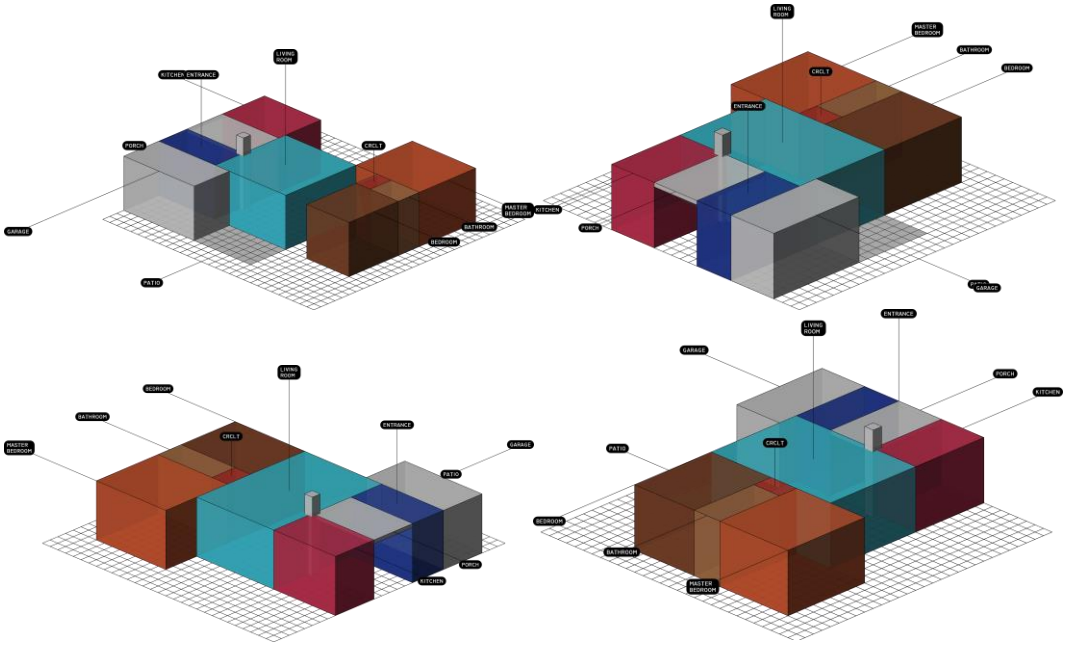


Figure 59. The results of Iteration 2 - Parallel projection from four sides - Case study 1 – Interactive run. (Drawn by the author)

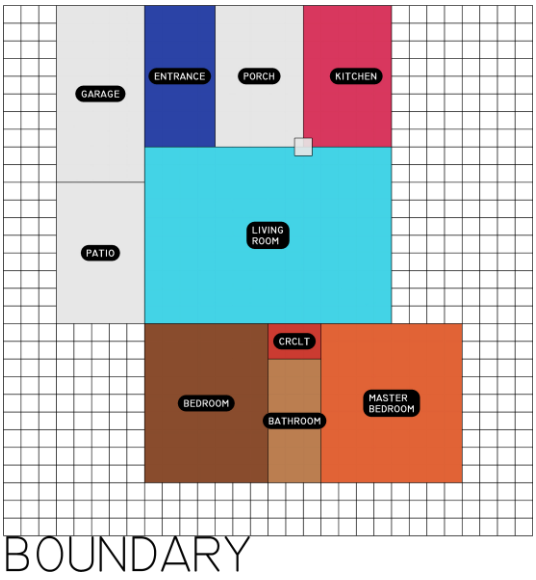


Figure 60. The results of Iteration 2 – Top view - Case study 1b – Interactive run. (Drawn by the author)

After Iteration 2, the user finds the living room too large and away from the initial inputs. The user decides that the hierarchy between C_{comp} and C_{dim} does not let Ho-Gen to develop better results in terms of LE dimensions. Thus, the user increases W_{dim} by one. Additionally, the user observes that the living room is largely obstructed by the surrounding LEs which causes a dark space living area throughout

the day. The user decides that this problem can be changed by giving living room an exposure on south direction. Lastly, master bedroom is also given an exposure on east direction to take the advantage of morning light. Ho-Gen is iterated again with the mentioned changes on the problem definition.

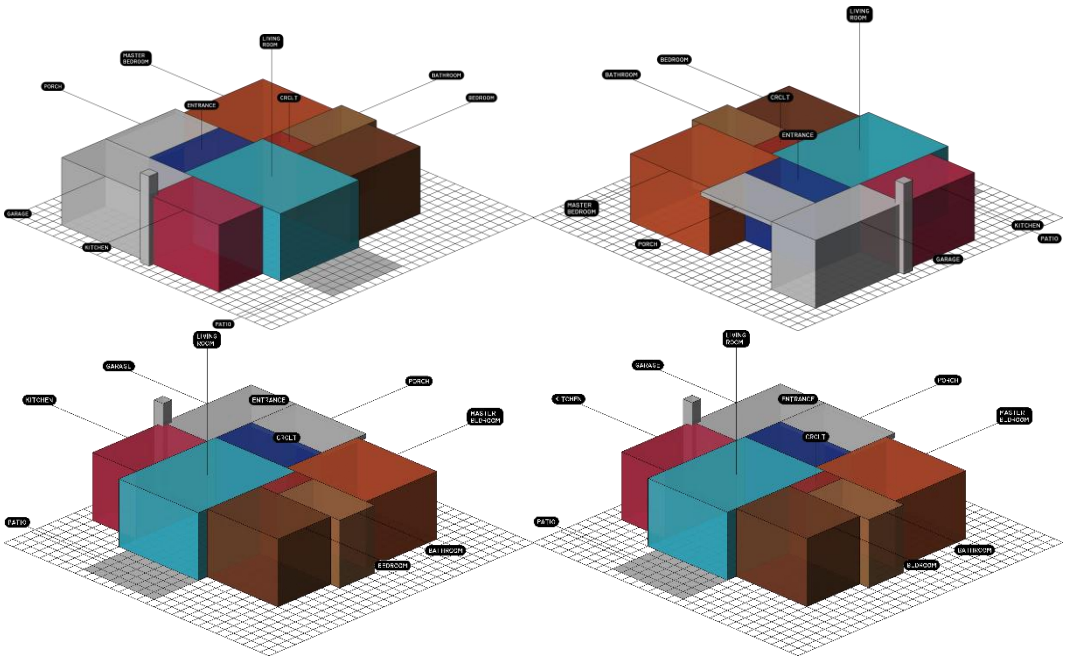


Figure 61. The results of Iteration 3 - Parallel projection from four sides - Case study 1 – Interactive run. (Drawn by the author)

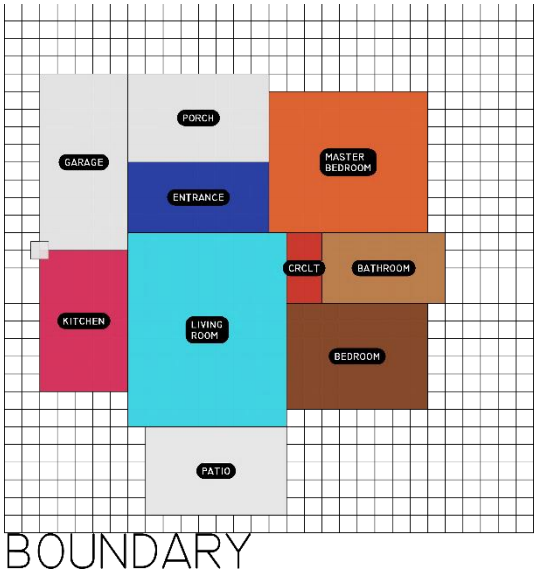


Figure 62. The results of Iteration 3 – Top view - Case study 1 – Interactive run. (Drawn by the author)

Ho-Gen successfully generated valid results for the changing problem definitions within the interactive run. Additionally, Ho-Gen showed success in generating a valid layout similar to the user's initial input in Iteration 2. The interactive run also generated better results in terms of LE dimensions because of user's intermediate interference to the generation process.

4.2.3 Case study 2a

This case study is conducted to test the capabilities of C_{comp} and different degrees of compactness in generating a variety of solutions. In this way, the building program of case study 2 is tested with three different compactness degrees. Ho-Gen is run 6 times for every compactness degree to test the validity and formal variation of the generated results. For a detailed look into the generation process, the rest of the figures are included in Appendix B, Appendix C, and Appendix D.

The case study with compactness degree A is conducted with a population of 2500 individuals. Every run is limited with a total stagnation of 60 generations. An average run took 1620 seconds. The weights of the evaluators C_{ovf} to C_{view} are 3-5-2-2-3-1-2-2.

The case study with compactness degree C is conducted with a population of 2500 individuals. Every run is limited with a total stagnation of 60 generations. An average run took 2056 seconds. The weights of the evaluators C_{ovf} to C_{view} are 3-8-2-2-3-1-2-2.

The case study with compactness degree D is conducted with a population of 2500 individuals. Every run is limited with a total stagnation of 60 generations. An average run took 2050 seconds. The weights of the evaluators C_{ovf} to C_{view} are 3-10-2-2-3-1-2-2.

The best layout alternative for every compactness degree is as follows:

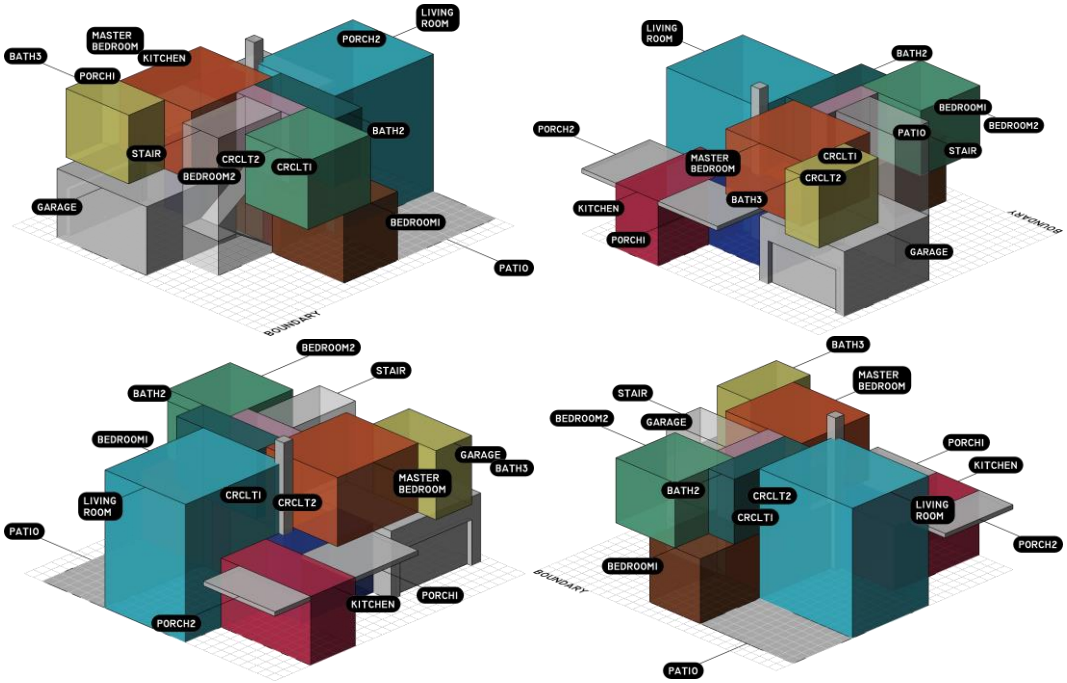


Figure 63. Alternative 6 - Case study 2 – Compactness A – Parallel projection from 4 sides (Drawn by the author).

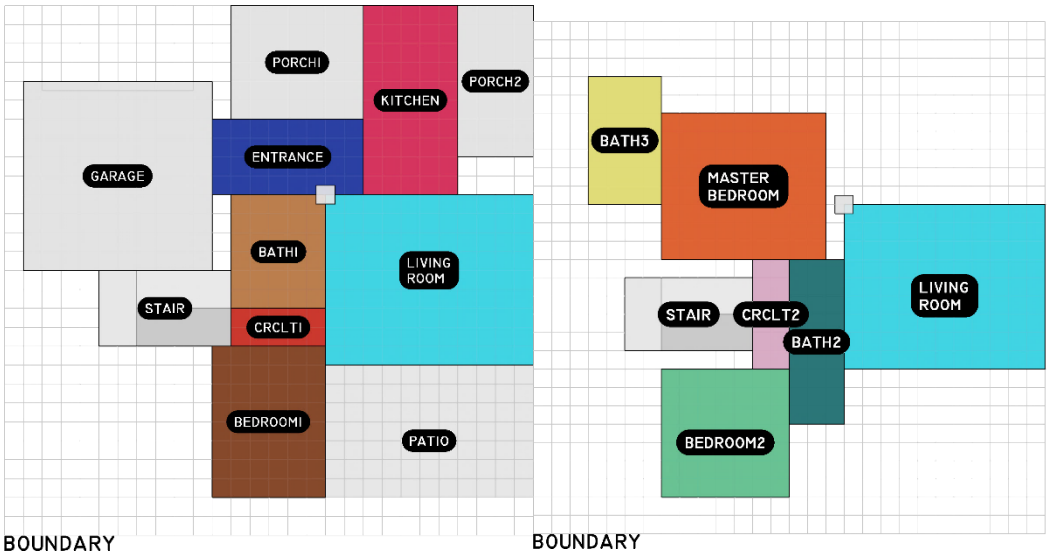


Figure 64. Alternative 6 - Case study 2a – Compactness A – Top view (Drawn by the author)

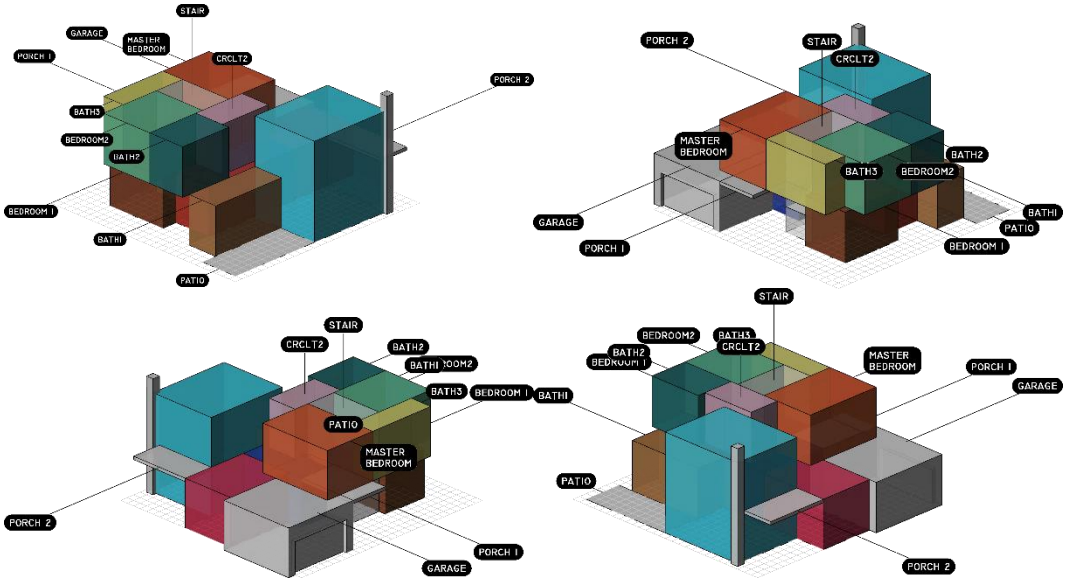


Figure 65. Alternative 5 - Case study 2 – Compactness C – Parallel projection from 4 sides. (Drawn by the author)

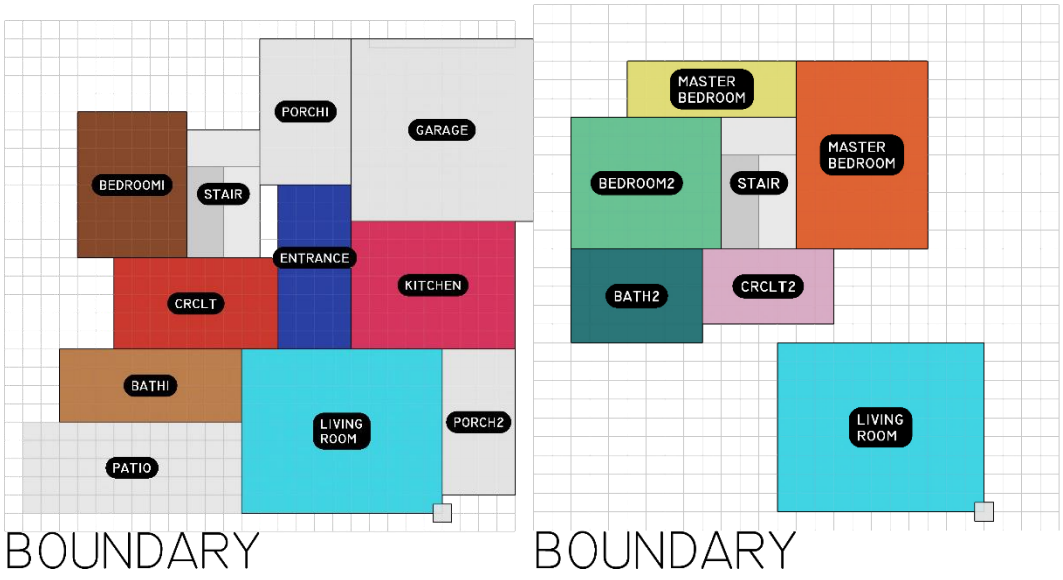


Figure 66. Alternative 5 - Case study 2 – Compactness C – Top view. (Drawn by the author)

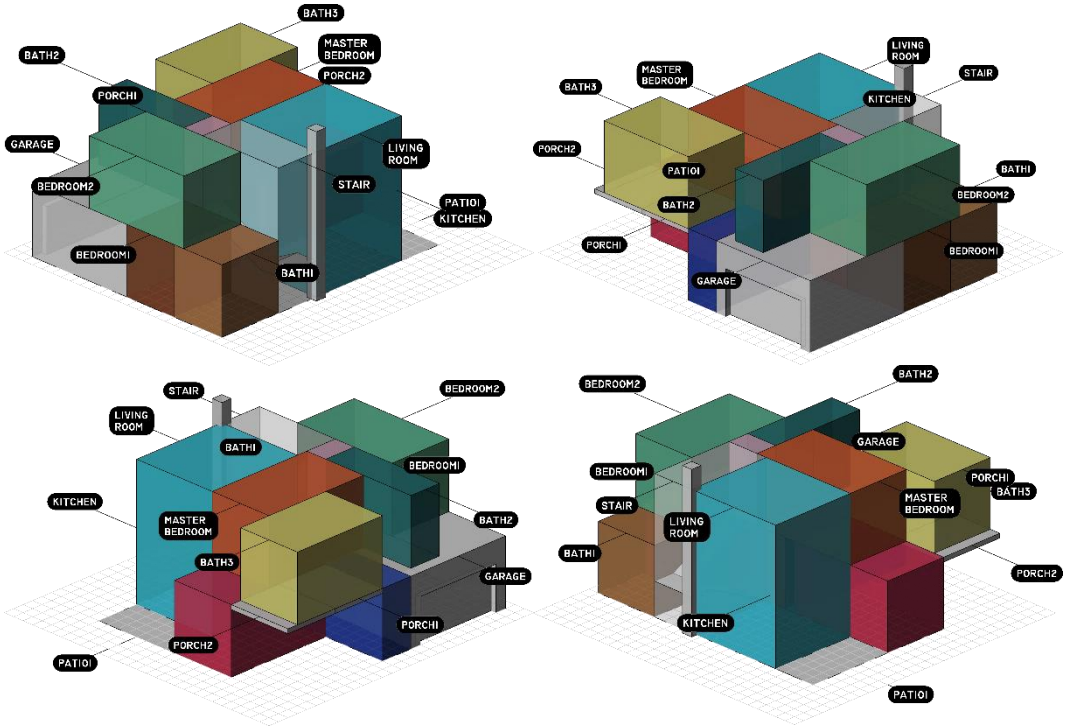


Figure 67. Alternative 5 - Case study 2 – Compactness D- Parallel projection from 4 sides. (Drawn by the author)

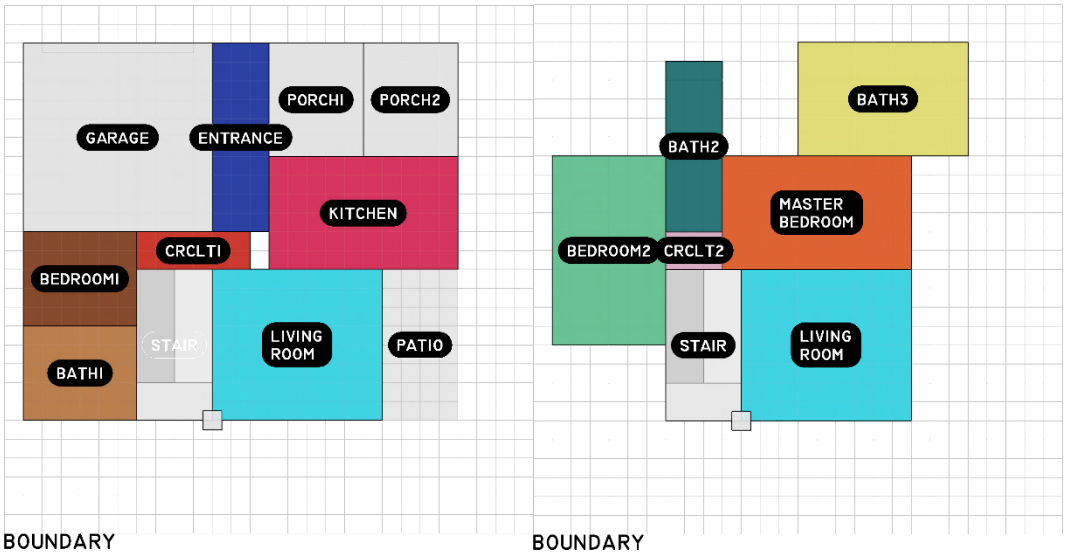


Figure 68. Alternative 5 - Case study 2 – Compactness D - Top view. (Drawn by the author)

Table 25 Weighted fitness result for six alternatives - Case study 2 – Compactness A.

	Fitness	C_{ovf}	C_{int}	C_{dim}	C_{rel}	C_{comp}	C_{cant}	C_{circ}	C_{view}
1	0.046	0.000	0.000	0.388	0.014	0.000	0.000	0.286	0.000
2	0.033	0.000	0.000	0.361	0.005	0.000	0.000	0.294	0.000
3	0.033	0.000	0.000	0.230	0.016	0.000	0.000	0.194	0.000
4	0.026	0.000	0.000	0.230	0.007	0.000	0.000	0.185	0.000
5	0.034	0.000	0.000	0.262	0.013	0.000	0.000	0.256	0.000
6	0.025	0.000	0.000	0.250	0.000	0.000	0.000	0.208	0.000

Table 26. Weighted fitness results for six alternatives - Case study 2 – Compactness C.

	Fitness	C_{ovf}	C_{int}	C_{dim}	C_{rel}	C_{comp}	C_{cant}	C_{circ}	C_{view}
1	0.056	0.000	0.000	0.406	0.013	0.301	0.000	0.222	0.000
2	0.072	0.000	0.184	0.483	0.007	0.000	0.000	0.227	0.000
3	0.054	0.000	0.000	0.509	0.007	0.258	0.000	0.227	0.000
4	0.073	0.111	0.000	0.605	0.033	0.126	0.000	0.309	0.000
5	0.052	0.000	0.000	0.292	0.000	0.352	0.000	0.359	0.000
6	0.057	0.000	0.000	0.287	0.037	0.259	0.000	0.153	0.000

Table 27.Weighted fitness result for six alternatives - Case study 2 – Compactness D.

	Fitness	C_{ovf}	C_{int}	C_{dim}	C_{rel}	C_{comp}	C_{cant}	C_{circ}	C_{view}
1	0.092	0.000	0.000	0.664	0.007	0.314	0.000	0.296	0.000
2	0.093	0.000	0.000	0.679	0.018	0.314	0.000	0.294	0.000
3	0.092	0.000	0.000	0.656	0.015	0.342	0.000	0.247	0.000
4	0.092	0.000	0.000	0.716	0.062	0.328	0.000	0.243	0.016
5	0.085	0.000	0.000	0.557	0.019	0.304	0.000	0.188	0.000
6	0.011	0.000	0.000	0.609	0.060	0.306	0.000	0.188	0.000

Different compactness degrees had a positive effect over the variety of results in this study. Improving the degree of exploration by a small change in the problem representation is found beneficial. One significant problem about the compactness degrees is its effect on C_{dim} penalties. The increase in compactness seems to push LEs to stretch or tighten in order to comply with the regularity of the layout. This issue can be problematic in terms of the validity of the layouts.

4.2.4 Case study 2b

This case study, similar to case study 1a, tests Ho-Gen's capability to generate diverse and valid results. An additional purpose of case study 2b is to test the effect of increased layout complexity over the performance of Ho-Gen.

In case study 2b, every floor is defined as a GRO. Ho-Gen is expected to generate compact floor layouts that are brought together with vertical circulation and maximum cantilever value.

As a medium scale layout problem, this study is conducted with a population of 2500 individuals. Every run is limited with a total stagnation of 60 generations. An average run took 2056 seconds. The weights of the evaluators C_{ovf} to C_{view} are 3-8-2-2-3-1-2-2. Ho-Gen is run for 6 times with the given inputs to test the validity and formal variation of the generated results. During each run, Ho-Gen bakes the fittest individual with an interval of 10 generations. This helps to explore the development of results and the effect of the evaluator weight hierarchy within the run. The figures that explain the development process of the fittest alternative in included in APPENDIX C. The generated results of six runs are as follows:

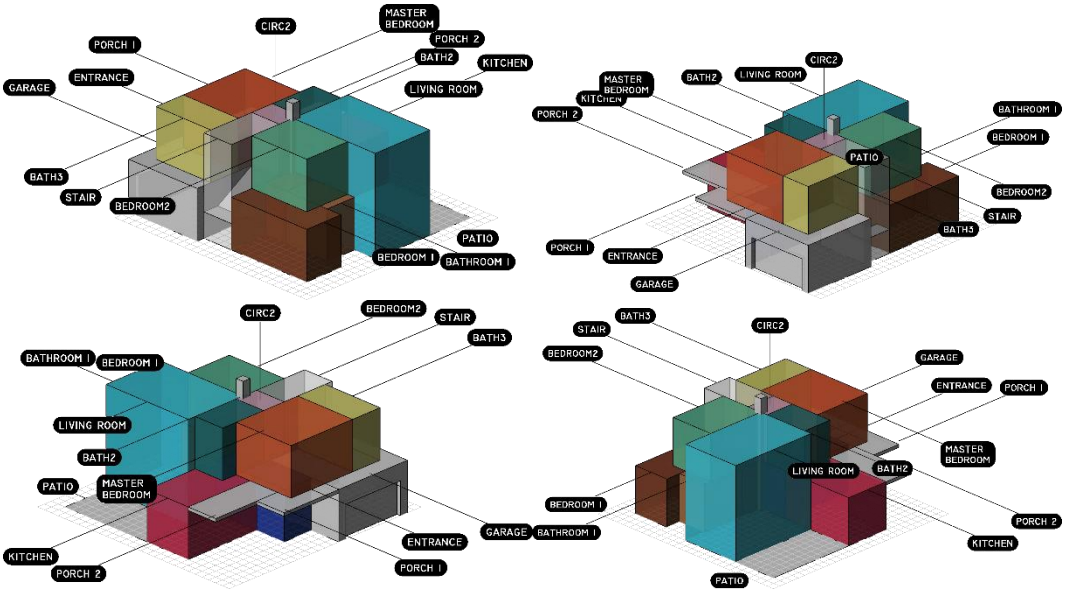


Figure 69. Alternative 1 - Case study 2 – Compactness C- Parallel projection from four sides. (Drawn by the author)

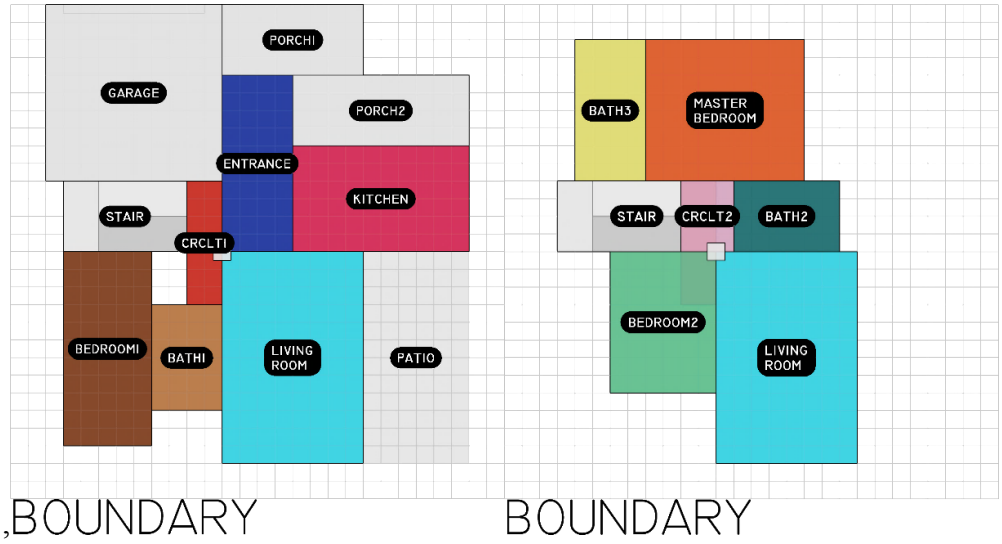


Figure 70. Alternative 1 - Case study 2 – Compactness C – Top view. (Drawn by the author)

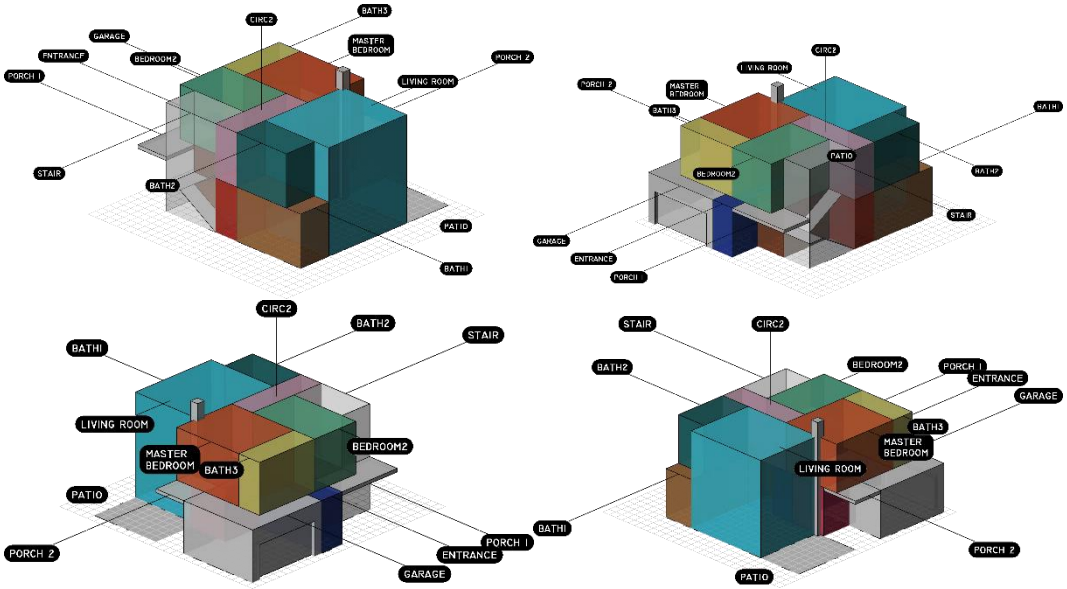


Figure 71. Alternative 2 - Case study 2 – Compactness C – Parallel projection from 4 sides. (Drawn by the author)

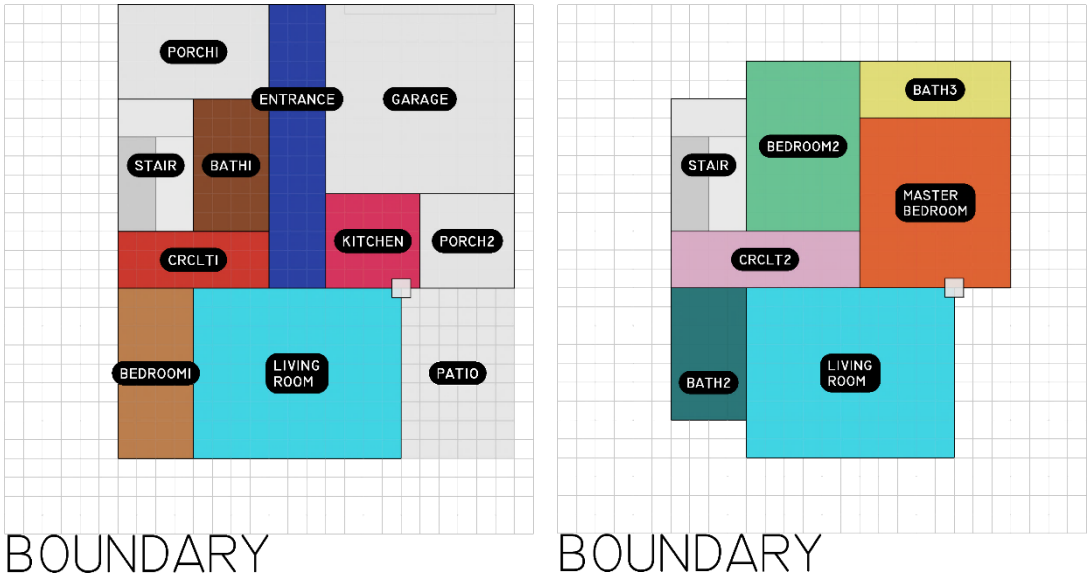


Figure 72. Alternative 2 - Case study 2 – Compactness B – Top view. (Drawn by the author)

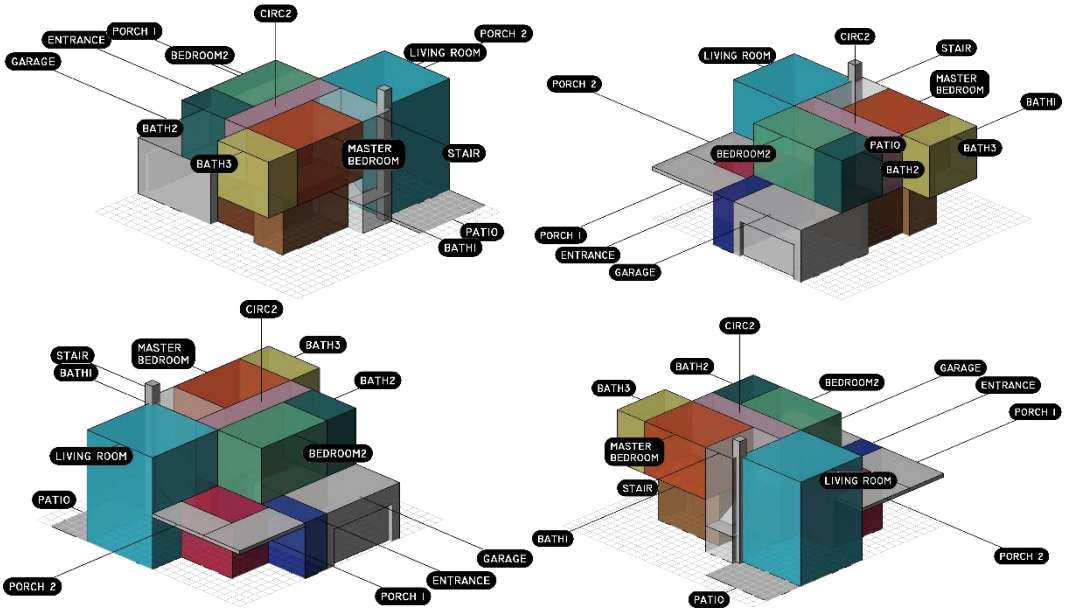


Figure 73. Alternative 3 - Case study 2 – Compactness C – Parallel projection from 4 sides. (Drawn by the author)

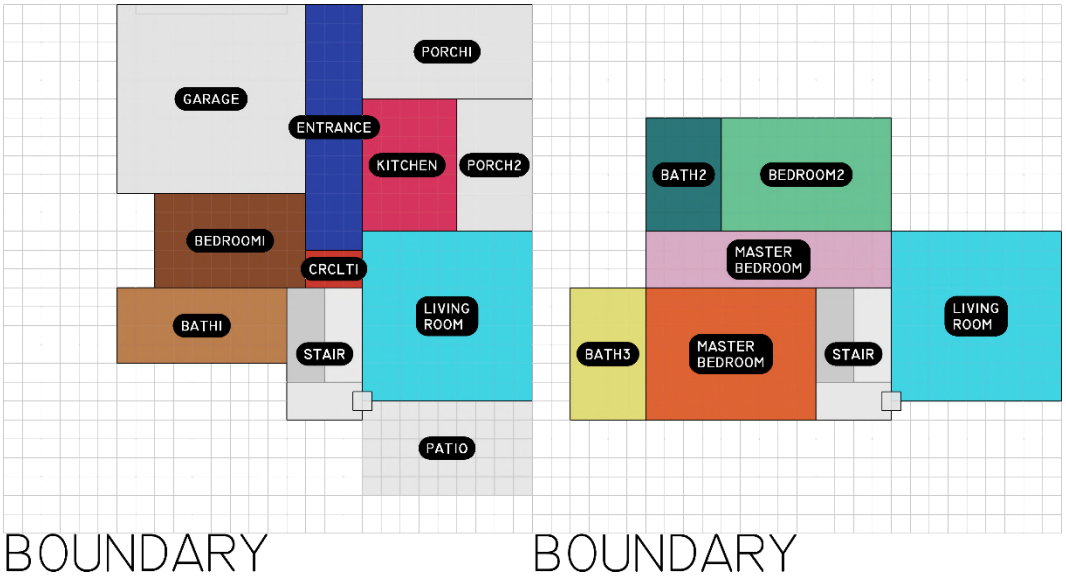


Figure 74. Alternative 3 - Case study 2 – Compactness C – Top view. (Drawn by the author)

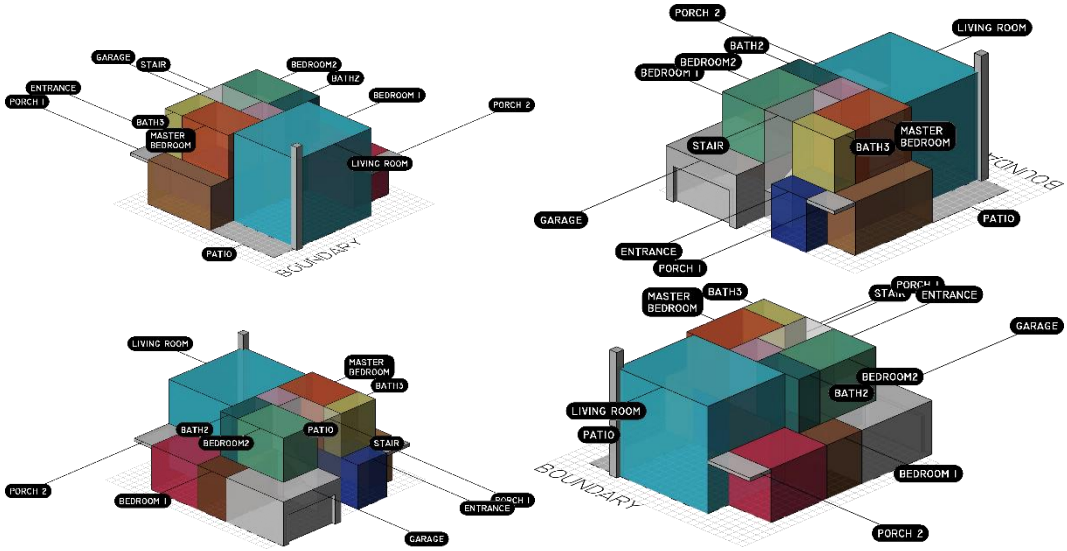


Figure 75. Alternative 4 - Case study 2 – Compactness C – Parallel projection from 4 sides. (Drawn by the author)

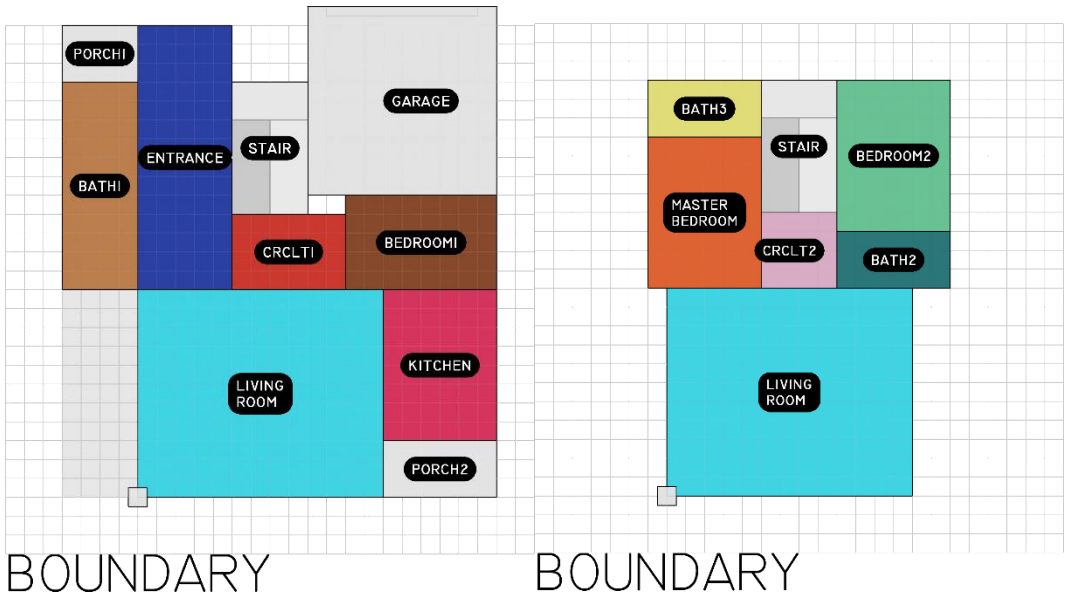


Figure 76. Alternative 4 - Case study 2 – Compactness C – Top view. (Drawn by the author)

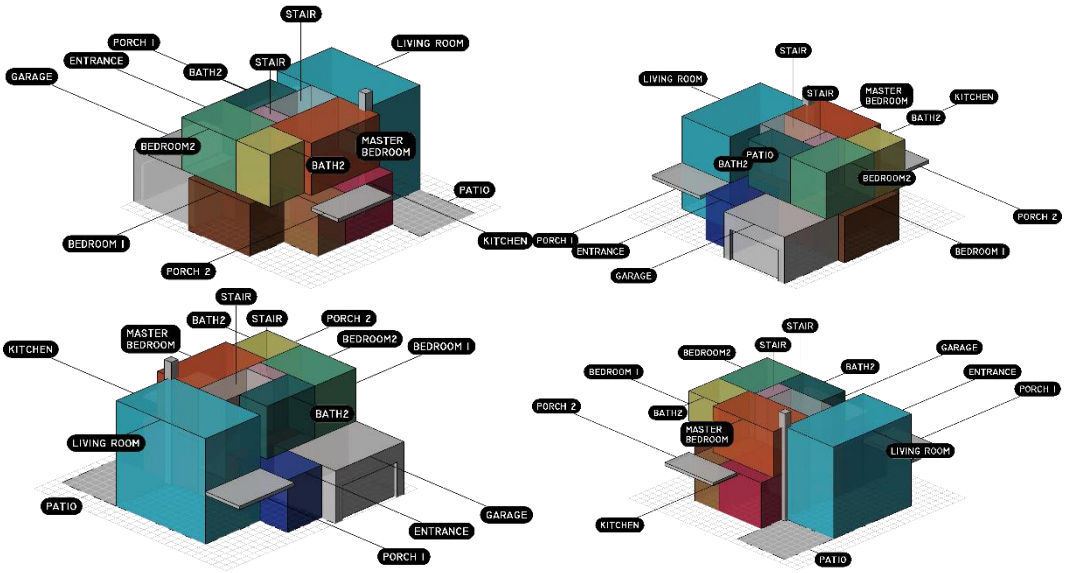


Figure 77. Alternative 6 - Case study 2 – Compactness C – Parallel projection from 4 sides. (Drawn by the author)

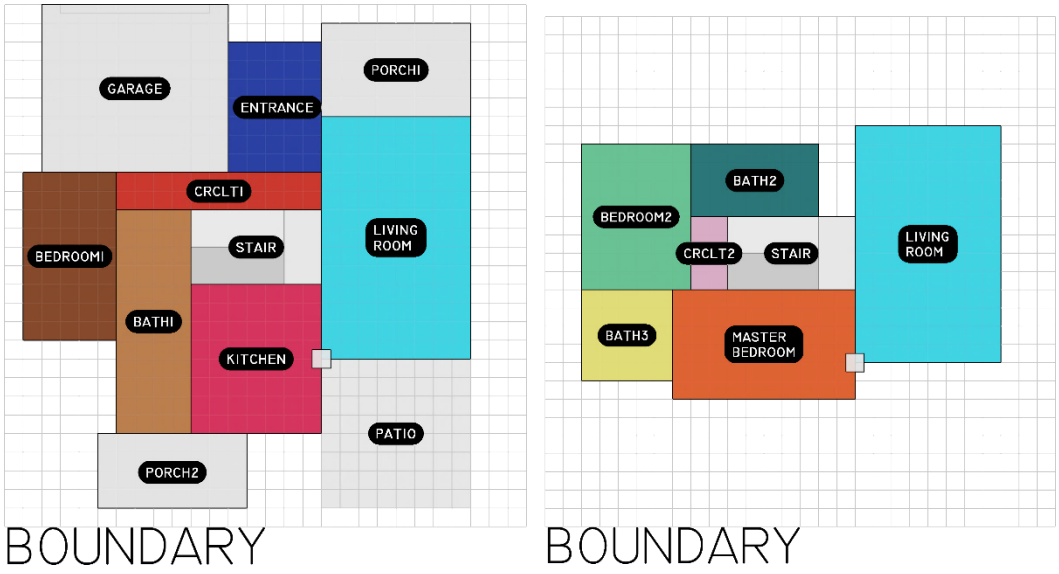


Figure 78. Alternative 6 - Case study 2 – Compactness C– Top view. (Drawn by the author)

Table 28. Weighted fitness results for six alternatives - Case study 2 – Compactness C.

	Fitness	C_{ovf}	C_{int}	C_{dim}	C_{rel}	C_{comp}	C_{cant}	C_{circ}	C_{view}
1	0.056	0.000	0.000	0.406	0.013	0.301	0.000	0.222	0.000
2	0.072	0.000	0.184	0.483	0.007	0.000	0.000	0.227	0.000
3	0.054	0.000	0.000	0.509	0.007	0.258	0.000	0.227	0.000
4	0.073	0.111	0.000	0.605	0.033	0.126	0.000	0.309	0.000
5	0.052	0.000	0.000	0.292	0.000	0.352	0.000	0.359	0.000
6	0.057	0.000	0.000	0.287	0.037	0.259	0.000	0.153	0.000

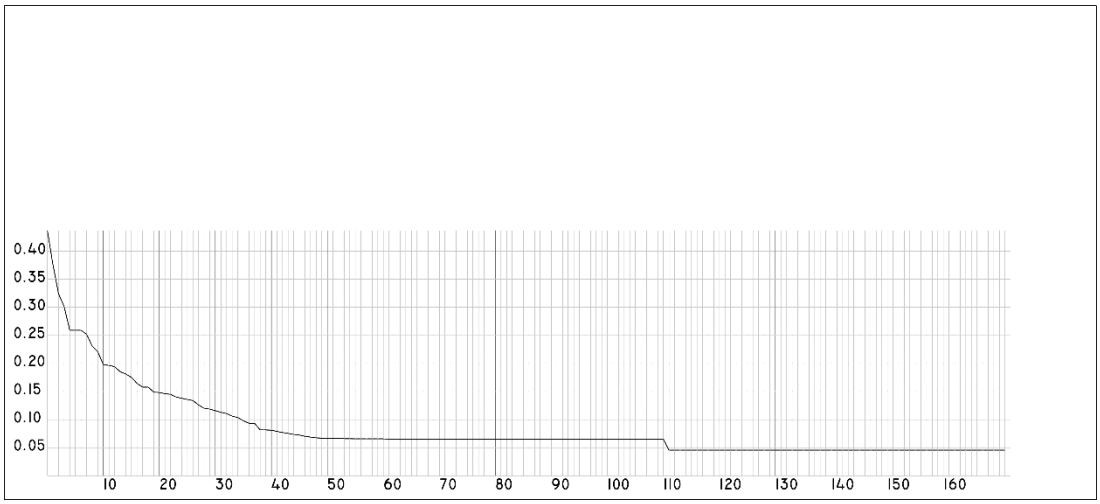


Figure 79. Best total fitness score - Case study 2 – Compactness C (Drawn by the author).

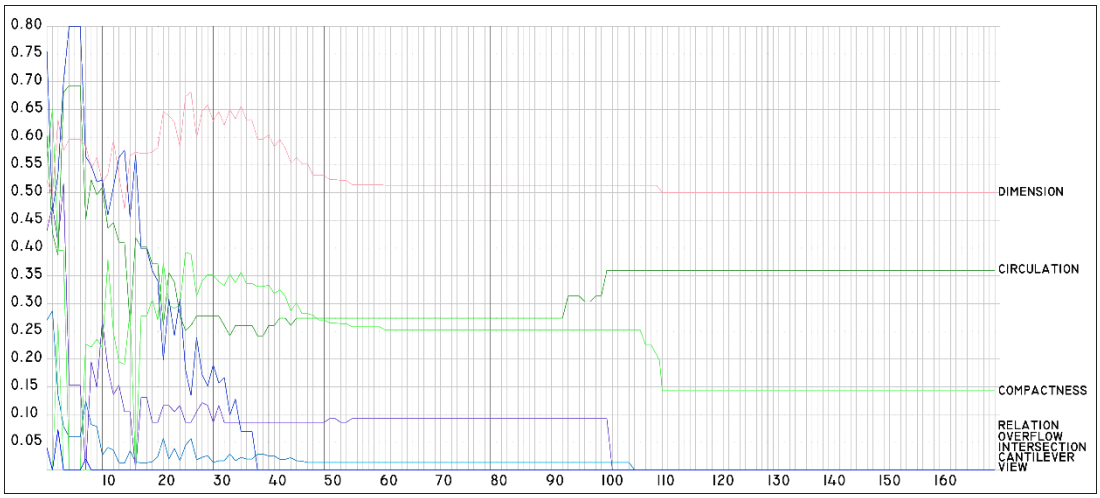


Figure 80. Best fitness score of evaluators - Case study 2 – Compactness C (Drawn by the author).

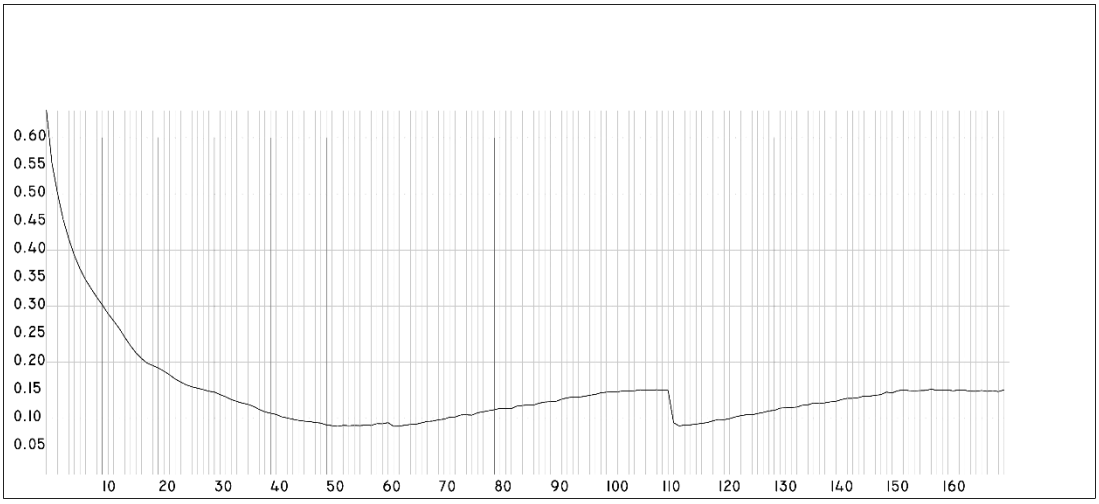


Figure 81. Average total fitness score - Case study 2 – Compactness C (Drawn by the author).

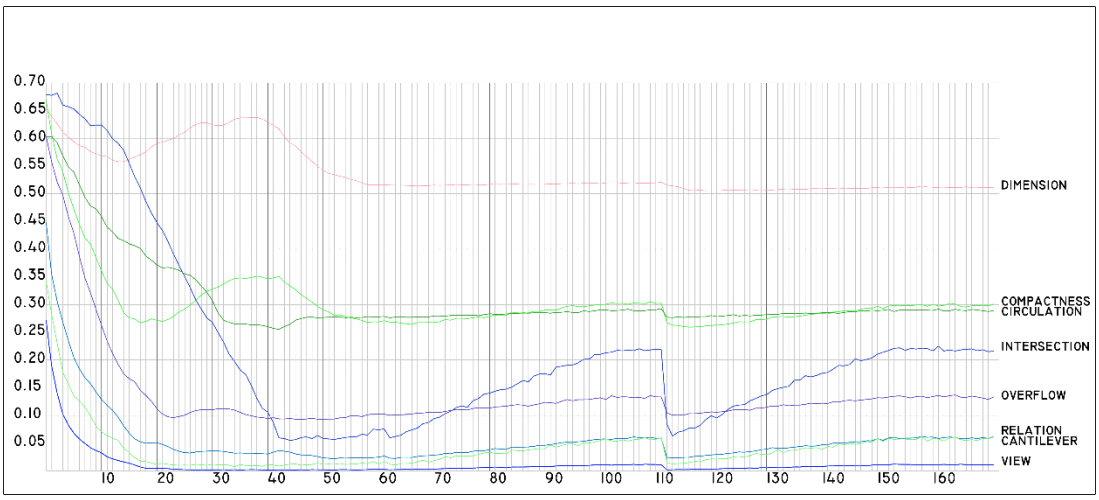


Figure 82. Average fitness score of evaluators - Case study 2 – Compactness C (Drawn by the author).

The increased amount of LEs caused an overall improvement in the variety of LE configurations. However, this also caused a reduction in the effect of search mechanism. According to average total fitness score graph (
Figure 81) the generation process has come across a long stagnation phase. This indicates that the overall increase in the automated divergence does not help every time to locate better solutions. This reduced performance can also be seen in the weighted fitness score table (Table 27). The penalty scores are relatively higher than case study 1a.

4.2.5 Case study 2c

In case study 2d, Ho-Gen is given an initial layout sketch to develop a similar but better performing result. Ho-Gen is expected to converge in a smaller time because of the smaller search space it requires to go through. The problem definition is the same with case study 2b. The generation process took 1220 seconds. The figures that explain the development process of the fittest alternative are included in APPENDIX E. The generated results of six runs are as follows:

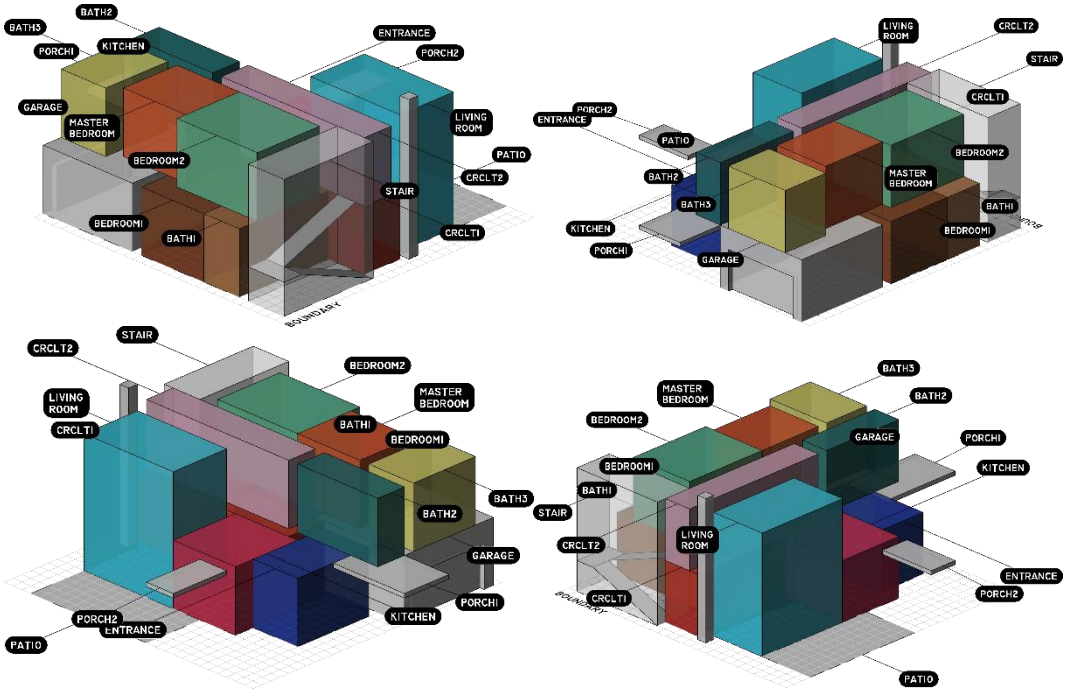


Figure 83. Initial sketch layout given to Ho-Gen – fitness: 0.176, C_{ovf}:0.019, C_{int}:0.068, C_{dim}:0.516, C_{rel}:0.075, C_{comp}:0.451, C_{cant}:0.000, C_{circ}:0.463, C_{view}:0.000 – Parallel projection from four sides - Case study 2d – Interactive run. (Drawn by the author)

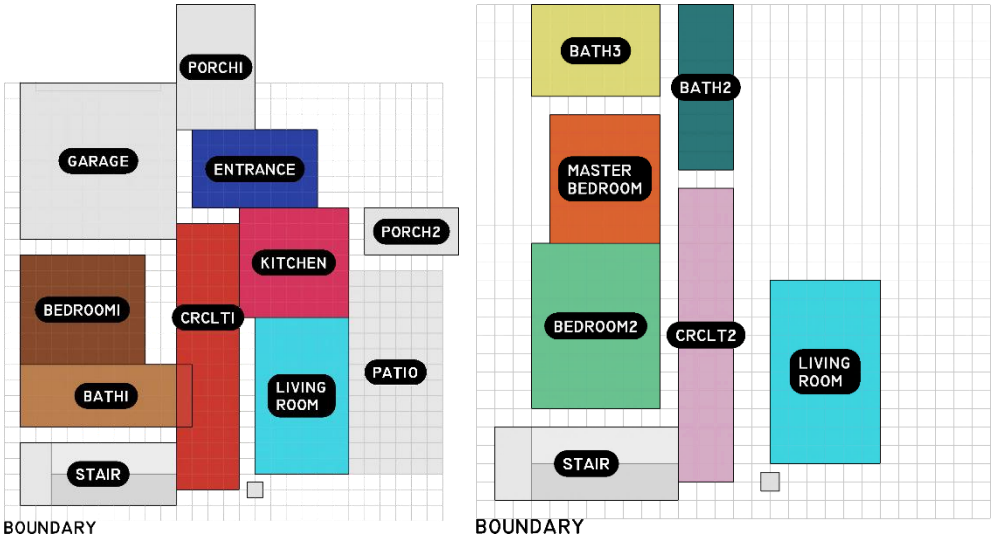


Figure 84. Initial sketch layout given to Ho-Gen – Top View - Case study 2d – Interactive run. (Drawn by the author)

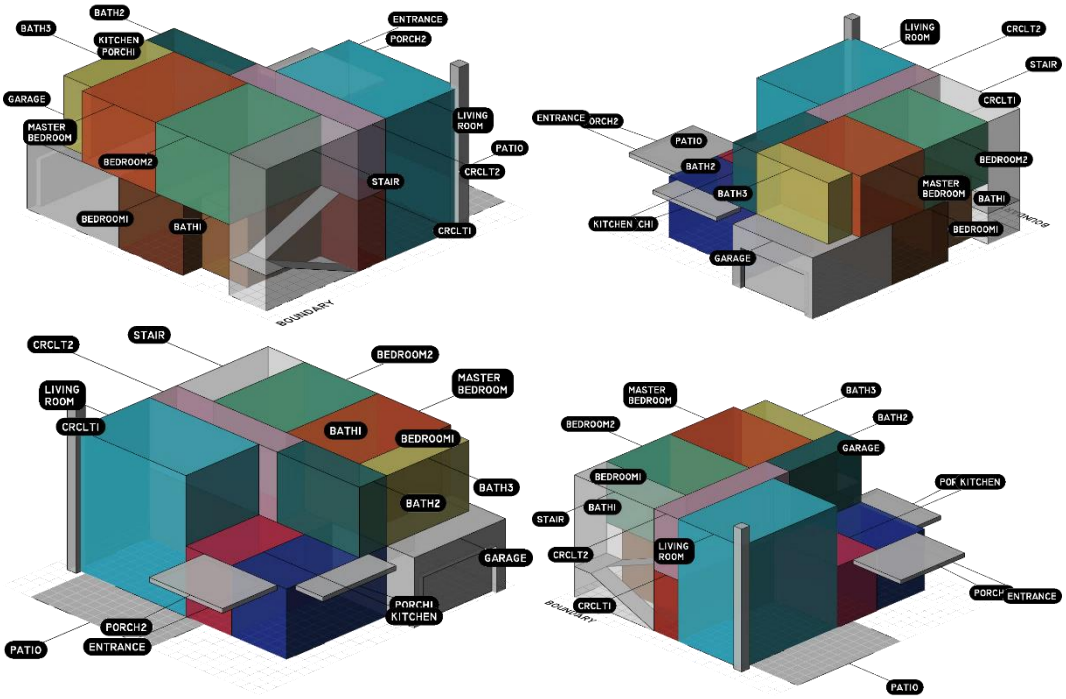


Figure 85. Best layout solution for generation 120 – fitness: 0.079, C_{ovf}:0.000, C_{int}:0.000, C_{dim}:0.516, C_{rel}:0.024, C_{comp}:0.163, C_{cant}:0.000, C_{circ}:0.414, C_{view}:0.000 – Parallel projection from four sides - Case study 2d – Interactive run. (Drawn by the author)

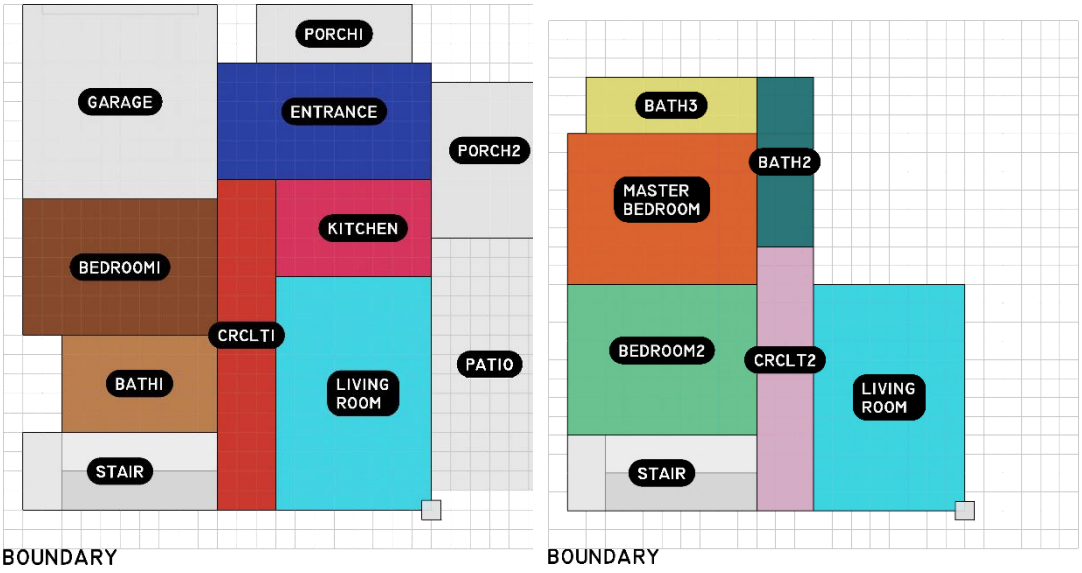


Figure 86. Best layout solution for generation 120 – Top View - Case study 2d – Interactive run. (Drawn by the author)

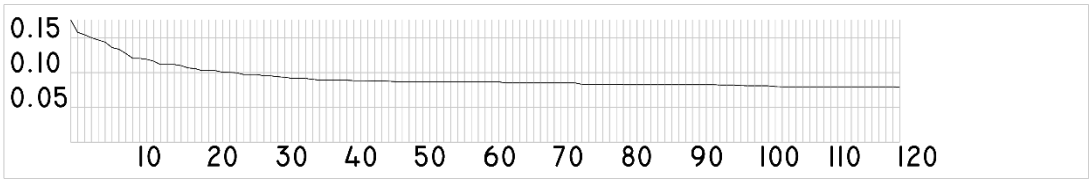


Figure 87. Best fitness score - Case study 2d (Drawn by the author).

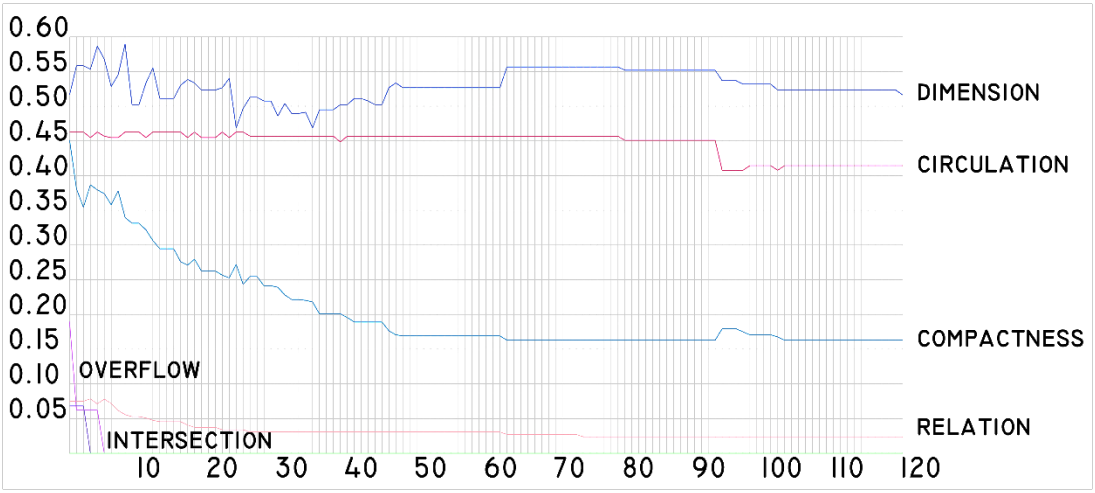


Figure 88. Best fitness scores of evaluators - Case study 2d (Drawn by the author).

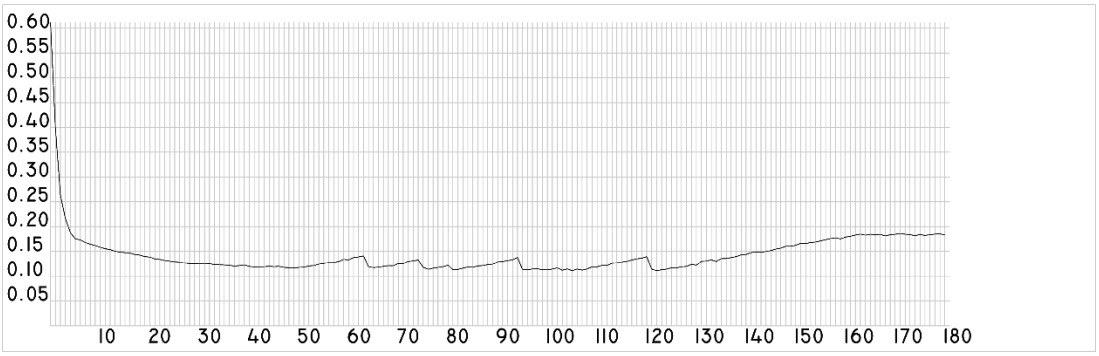


Figure 89. Average fitness score - Case study 2d (Drawn by the author).

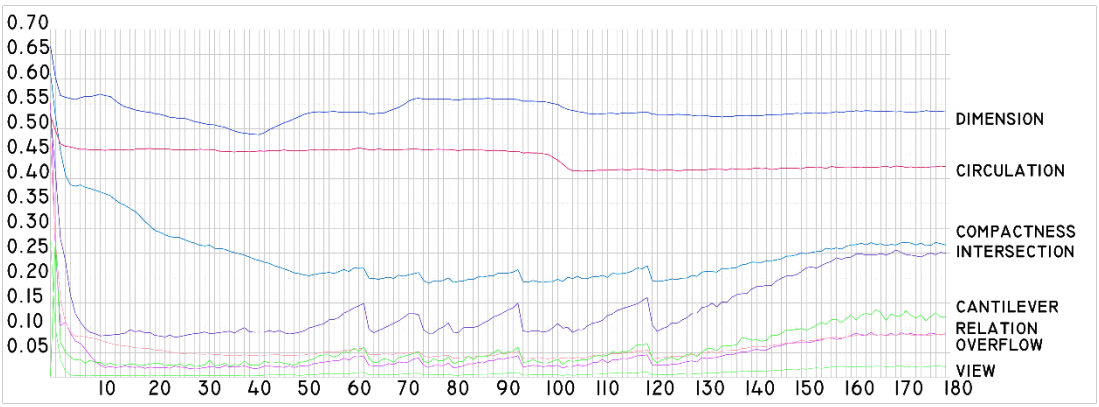


Figure 90. Average fitness scores of evaluators - Case study 2d (Drawn by the author).

Ho-Gen successfully developed a rough initial sketch into a functional building layout in lesser time. This time, however, Ho-Gen generated a layout with a worse fitness score than the non-interactive run. It is possible that the algorithm skipped a better initial layout option in the beginning because of the initial conditions provided by the user. Another reason is the higher C_{dim} penalty of Ho-Gen. An observation on the result shows that the reasons for the high C_{dim} penalty are trivial and can easily be fixed by the user in the end. Yet, Ho-Gen cannot iterate toward solutions with a better C_{dim} score. Ho-Gen, expectedly, disregards the LE dimensions to generate valid layouts at the start. However, it cannot fix it in the later stages because the later solutions with a better C_{dim} score violate important evaluators such as C_{int} or C_{comp}. This issue requires the user interference within the search process to the evaluator weights.

CHAPTER 5

CONCLUSION

The current state of the housing industry demands collaboration models that can increase the influence of the occupants in the design process. In this way, computational design tools for non-expert users generate a chance with the advancement of mass customization in construction and prevalent use of computers. The current state of non-expert design tools, however, is problematic due to the support they provide or the expertise they expect. Configurators present a little amount of choice to users, while drafting tools overwhelm the user with the amount of control they provide. Generative tools present a higher level of customization in terms of the solutions because of their dependence on user input. However, they offer little or no customization for their generative mechanisms which are either too bounded by the rules of its developer or requires an architectural or computational expertise. Therefore, the purpose of this study was to develop a new computational model that can enhance designer's control over the generation process.

The research has started with an investigation over the current literature on such subjects as computational non-expert design tools, design automation, non-expert and computation interaction, computational layout design, and genetic algorithms. In the first part of the review, a general research over the computational non-expert design tools brought out that such tools are popular approaches among the user-centered models for mass customization. Design companies, rather than funding market research techniques or lead user idea generation models to acquire a general standard in terms of the needs of the occupants, provide them the necessary tools for the design of their own house. The purpose of computational non-expert tools is to provide a user-friendly interface which requires little or no additional training beyond user's inherent design capabilities and personal requirements.

In the second part of the review, the research has shifted towards the need for

interactive interfaces for the development of computational tools. The research has taken the advantage of recent reports on automation to emphasize the lower rate of automation expectancy for design related occupations despite the current state of automation for many occupations. The research has shown that full automation requires well-defined problems which can be analyzed into clear objectives through objective and rational methods. On the contrary, the unavailability of such rational and objective analysis methods for design was explained over the "design methods movement" of 1960s and the deep criticism towards this movement. Instead, the research has acknowledged the need for designer's subjective interpretation on the design problem as a way to cope with the vaguely defined design problems with a high multiplicity of objectives.

The third part has examined the designer strategies to cope with the ill-defined nature of design problems in order to develop a computational model to support these activities for non-expert designers. This part revealed the importance of trial-and-error learning as a way to explore the design problem and the requirement for generating a high number of alternative solutions to reach better results. The assessment of an appropriate solution space for the design of SFH brought the research into layout design problem in architecture.

The forth part has evaluated the computational approaches to layout design problem. The investigation of the current computational approaches revealed that GA approaches bring certain advantages for design related problems. GA, as a metaheuristic, offers a general solution method that requires less problem specific information on the problem. In this way, metaheuristic approaches provide a general advantage for non-experts. As the solution method is guided by general rules away from expert knowledge, non-expert designers can interact with metaheuristics in an easier way. Another advantage of GA is their population-based working principle which improves the efficiency of exploration in the high multiplicity of solutions.

In the last section of the review, the research has identified a main problem in GA approaches. The general user interaction in GA is limited to the initial definition of variables and the observation of their results. Additionally, GA's capabilities in the exploration of large solution spaces is a computationally demanding process that

requires time. The limited interaction together with the large time requirements of GA causes latency between the problem definition and feedback mechanism. In this way, alternative interactive models are investigated in order to present a trial-and-error learning based interface to the user.

Therefore, this research was set out to develop a new computational approach, House Generator (Ho-Gen), for the interactive generation of 3D layout solutions specifically for SFH. Ho-Gen utilizes an interactive interface for genetic algorithms (GA) in order to combine GA's creative power in exploring complex problems with the advanced designer control over the generation mechanism. During a Ho-Gen run, a user can interfere with the GA run, observe the preliminary results, and alter the generation process by the following ways:

- Changing the problem definition through manipulating layout components and their topological relations.

- Making manual changes over the generated layout geometries. The user can also start with an initial layout in order to focus a significant part within the design space.

- Changing the evaluator weights to adjust their relative importance within the overall fitness function.

Ho-Gen is also developed with specific attention to the character of SFH layouts. Such properties are given as:

- Representing specific layouts elements under a group hierarchy. Groups can be specifically defined by the user or can be automatically defined under certain degrees of compactness.

- Defining the location of layout elements through their direction within the envelope. SFH is a free-standing building that is open on all sides. Designer can arrange the layout elements according to daylight requirements or other

environmental causes. This issue is tried to be implemented through the VIEW evaluator to give the required view for the layout elements.

- Generating multi-floor layouts with in-between vertical circulation elements. User can define layout elements in a floor hierarchy. The geometrical relation in between different floors is controlled by the CANTILEVER evaluator.

- Defining layout elements with different character. Open spaces are increasingly becoming important parts of SFH. Ho-Gen implements open and semi-open spaces with PORCH and PATIO components. Horizontal and vertical circulation elements are added into the overall layout. Some spaces are given the possibility to be double height spaces.

Ho-Gen is tested with two major layout problems with changing complexity regarding the number of layout elements, topological and geometric user criteria. Ho-Gen successfully generated valid layout options for a two floor SFH of 15 layout elements fewer than thirty minutes, however, tests are realized with a considerably low-end computer for the time. A better system can significantly reduce the current time requirements for such a problem. The case studies are also subjected to minor alterations in terms of group relations and compactness to check their effect on the variety of results. The generated variety by changing compactness degrees was found beneficial in terms of the ease of exploring different massing options through simple alterations. On the other hand, the extra time requirement of C_{comp} evaluator brings the need to develop a more efficient computation method for such action.

Additionally, every case study is tested with an interactive scenario. In the first scenario, the user started with a less specific problem definition and either added extra conditions or changed the existing ones through the observations within the generation process. User also made manual alterations on the generated layout. Ho-Gen successfully generated quick feedback for the changing problem definitions, thus allowing the designer to develop the design problem in a systematic and time efficient manner. The second scenario allowed the designer to sketch a quick initial layout to guide the generation process. In the end, Ho-Gen generated a layout in the

same topological structure with user's initial sketch. The generation of the results took a significantly lower time than the automated results.

5.1 Limitations and Future Work

The major limitation for this research is the insufficient number of interactive case studies. Despite the initial promise of user interaction such as better C_{dim} score in case study 1b and less time requirement in case study 2c, the tests for interactive case studies should be examined under increased detail. This examination is important for the decisions about the correct times and ways user interaction. Another limitation for this research is the lack of case studies with actual non-expert designers. Current case studies works for the validation of the model description acquired in the literature review. However, further case studies with non-experts are required to assess Ho-Gen's real performance in the support of occupants. As an example, testing the arrangement process of the evaluator weights with non-experts is a direct necessity. The interactive support of Ho-Gen can simplify the trial-and-error learning process, however, leaving a non-expert with 8 evaluator weights to control can be problematic at the start. Such problems can make way further simplifications in Ho-Gen such as providing an early set of evaluator weights based on the problem. In this way, Ho-Gen requires a real user interface that is both guiding and easy to operate.

One other important limitation of this research is the absence of a deep analysis into the precedents in single-family house. Such an analysis can help the development of general design concepts in terms of SFH. These concepts can be about functional requirements such as a home office setting or a holiday house. The general concepts can be turned into predefined input sets or combinations to provide a more user-friendly interface at the start. Occupants can use these concepts to develop early solutions immediately. One important point for the development of these concepts is their solution space. The set of inputs that is represented by concepts should not get too specific in order to keep divergent exploration capabilities.

Future work on the interactive genetic algorithm approach:

- Ho-Gen's generation process can be changed into a fully visible interface to

its user. User's interference to the run can be simplified to a button that halts the process. Current user interaction is only possible at the end of the search process. The duration of such process can be defined at the start. However, this can be problematic for the users that do not know the time requirements for such search.

- Whole search process can be utilized as a family of results which enables the user to turn back and try other alternatives.

- Much of the case studies have been done by a low-end computer for the time. GA approach for Ho-Gen can be developed to decrease the required time for exploration thus making possible to visualize user input more quickly.

Possible developments on the problem representation:

- The functional analysis can also utilize furnishing of LEs. LE geometries can be generated by the organization of furnishings, and then these resultant spaces can be configured similar to the hierarchical generation approaches. This can also aid the currently shallow state of C_{dim}. In a way, occupants can prefer to define a space by its functional setting such as a kitchen counter and dining table rather than an area value.

- Ho-Gen just considers rectangular geometries on the plan and section; irregular shapes are not estimated. Many SFH is made of such irregular shaped LEs consideration of cut angles or a degree of convex geometries can bring interesting solutions.

REFERENCE

"A New Way Forward for Mobility." Waymo. Accessed January 12, 2018. https://waymo.com/redirect/.

Abdelmohsen, Sherif, Ayman Assem, Sherif Tarabishy, and Ahmed Ibrahim. "A Heuristic Approach for the Automated Generation of Furniture Layout Schemes in Residential Spaces." In *Design Computing and Cognition '16*, 459–75. Springer, Cham, 2017. https://doi.org/10.1007/978-3-319-44989-0_25.

"About GAAP." Accessed January 14, 2018. http://www.accountingfoundation.org/cs/ContentServer?c=Page&cid=1176164538898&d=&pagename=Foundation%2FPage%2FFAFBridgePage.

"Accounting." *Merriam-Webster*. Accessed January 14, 2018. https://www.merriam-webster.com/dictionary/accounting.

Alexander, Christopher. *Notes on the Synthesis of Form*. Vol. 5. Harvard University Press, 1964.

Alpaydin, Ethem. *Introduction to Machine Learning*. 2nd ed. The MIT Press, 2010.

Armour, Gordon C., and Elwood S. Buffa. "A Heuristic Algorithm and Simulation Approach to Relative Location of Facilities." *Management Science*, no. 2 (1963): 294.

"Autopilot." Accessed January 12, 2018. https://www.tesla.com/autopilot.

Bahrehmand, Arash, Thomas Batard, Ricardo Marques, Alun Evans, and Josep Blat. "Optimizing Layout Using Spatial Quality Metrics and User Preferences." *Graphical Models* 93, no. Supplement C (September 1, 2017): 25–38. https://doi.org/10.1016/j.gmod.2017.08.003.

Bayazit, Nigan. "Investigating Design: A Review of Forty Years of Design Research." *Design Issues* 20, no. 1 (January 1, 2004): 16–29. https://doi.org/10.1162/074793604772933739.

Bazaraa, M.s. "Computerized Layout Design: A Branch and Bound Approach." *AIIE Transactions* 7, no. 4 (01 1975): 432–38. https://doi.org/10.1080/05695557508975028.

Belk, Russell W. "Possessions and the Extended Self." *Journal of Consumer Research* 15, no. 2 (1988): 139–168.

Bentley, Peter J., and David W. Corne. "Introduction to Creative Evolutionary

Systems." In *Creative Evolutionary Systems*, edited by Peter J. Bentley and David W. Corne, 1–75. Morgan Kaufmann Publishers Inc., 2002.

Birattari, Mauro, Luis Paquete, Thomas Stützle, and K. Varrentrapp. "Classification of Metaheuristics and Design of Experiments for the Analysis of Components," 2001. http://hdl.handle.net/2013/ULB-DIPOT:oai:dipot.ulb.ac.be:2013/77018.

Blaser, Werner. *Mies van Der Rohe: Farnsworth House-Weekend House*. 1 edition. Basel ; Boston: Birkhauser, 1999.

Bozer, Yavuz A., Russell D. Meller, and Steven J. Erlebacher. "An Improvement-Type Layout Algorithm for Single and Multiple-Floor Facilities." *Management Science*, no. 7 (1994): 918.

Burmeister, Jay, and Janet Wiles. "The Challenge of Go as a Domain for AI Research: A Comparison between Go and Chess." In *Intelligent Information Systems, 1995. ANZIIS-95. Proceedings of the Third Australian and New Zealand Conference On*, 181–186. IEEE, 1995.

Cagan, J, Mi Campbell, S Finger, and T Tomiyama. "A Framework for Computational Design Synthesis: Model and Applications." *JOURNAL OF COMPUTING AND INFORMATION SCIENCE IN ENGINEERING* 5, no. 3 (September 2005): 171–81.

Campbell, Murray, A. Joseph Hoane, and Feng-hsiung Hsu. "Deep Blue." *Artificial Intelligence* 134, no. 1 (January 1, 2002): 57–83. https://doi.org/10.1016/S0004-3702(01)00129-1.

Chow, R.Y. "House Form and Choice." *Traditional Dwellings and Settlements Review* 9, no. 2 (1998): 51–62.

Cole, N., S. J. Louis, and C. Miles. "Using a Genetic Algorithm to Tune First-Person Shooter Bots." In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, 1:139–145 Vol.1, 2004. https://doi.org/10.1109/CEC.2004.1330849.

Corne, David, and Peter Bentley. *Creative Evolutionary Systems*. The Morgan Kaufmann Series in Artificial Intelligence. San Francisco, CA: Morgan Kaufmann, 2002.

Cross, Nigel. "Design Cognition: Results from Protocol and Other Empirical Studies of Design Activity." In *Design Knowing and Learning: Cognition in Design Education.*, edited by Charles M. Eastman, W. Michael McCracken, Wendy C. Newstetter, Charles M. Eastman (Ed), W. Michael McCracken (Ed), and Wendy C. Newstetter (Ed), 79–103. Oxford, England: Elsevier Science Ltd, 2001. https://doi.org/10.1016/B978-008043868-9/50005-X.

———. *Designerly Ways of Knowing*. London : Springer-Verlag London Limited, 2006., 2006.

"Customize a Green, Modern, Affordable Home by Postgreen Homes - CUSTOMIZE - Passive Houses." Accessed December 25, 2017. http://customize.postgreenhomes.com/?s=0.

Darke, Jane. "The Primary Generator and the Design Process." *Design Studies* 1, no. 1 (1979): 36–44.

Davis, Lawrence. "Handbook of Genetic Algorithms," 1991.

Davis, Stanley M. *Future perfect.* Reading, Mass. [u.a.]: Addison-Wesley Publ. Co., 1987.

Dino, Ipek Gürsel. "An Evolutionary Approach for 3D Architectural Space Layout Design Exploration." *Automation in Construction* 69 (2016): 131–150.

Donelson, Dain C., John McInnis, and Richard D. Mergenthaler. "Explaining Rules-Based Characteristics in US GAAP: Theories and Evidence." *Journal of Accounting Research* 54, no. 3 (2016): 827–861.

Dorst, Kees. "The Problem of Design Problems." In *Expertise in Design*, 135–147. Sydney, Australia, 2003.

Doulgerakis, A. "Genetic Programming + Unfolding Embryology in Automated Layout Planning." Masters, UCL (University College London), 2007. http://discovery.ucl.ac.uk/4981/.

Dreyfus, Hubert L. "Intelligence Without Representation–Merleau-Ponty's Critique of Mental Representation the Relevance of Phenomenology to Scientific Explanation." *Phenomenology and the Cognitive Sciences* 1, no. 4 (2002): 367–383.

Duarte, José Pinto. "Customizing Mass Housing : A Discursive Grammar for Siza's Malagueira Houses." Thesis, Massachusetts Institute of Technology, 2001. http://dspace.mit.edu/handle/1721.1/8189.

Dugan, Andrew, and Bailey Nelson. "3 Trends That Will Disrupt Your Workplace Forever." Gallup.com, June 8, 2017. http://news.gallup.com/businessjournal/211799/trends-disrupt-workplace-forever.aspx.

"Express Modular." Accessed December 10, 2017. http://expressmodular.com/dragonfly_editor.php.

Feng, Sheng-Yu, and Chuan-Kang Ting. "Painting Using Genetic Algorithm with Aesthetic Evaluation of Visual Quality." In *Technologies and Applications of*

Artificial Intelligence, 124–135. Springer, 2014. http://link.springer.com/chapter/10.1007/978-3-319-13987-6_12.

Flack, Robert W. J. "Evolution of Architectural Floor Plans." Masters, Brock University, 2011. http://dr.library.brocku.ca/handle/10464/3409.

Flemming, Ulrich, Can A. Baykan, Robert F. Coyne, and Mark S. Fox. "Hierarchical Generate-and-Test vs Constraint-Directed Search." In *Artificial Intelligence in Design'92*, 817–838. Springer, 1992. https://link.springer.com/chapter/10.1007/978-94-011-2787-5_41.

"Forget Cars, Self-Driving Shuttles Are the Future of Transportation." WIRED. Accessed January 12, 2018. https://www.wired.com/story/las-vegas-shuttle-crash-self-driving-autonomous/.

"Forget Self-Driving Cars. Automated Public Transportation Is Coming." Roadshow. Accessed January 12, 2018. https://www.cnet.com/roadshow/news/self-driving-cars-automated-public-transport-bus/.

France-Presse, Agence. "World's Best Go Player Flummoxed by Google's 'Godlike' AlphaGo AI." *The Guardian*, May 23, 2017, sec. Technology. http://www.theguardian.com/technology/2017/may/23/alphago-google-ai-beats-ke-jie-china-go.

Frey, Carl Benedikt, and Michael A. Osborne. "The Future of Employment: How Susceptible Are Jobs to Computerisation?" *Technological Forecasting and Social Change* 114 (2017): 254–280.

Futuyma, Douglas. *Evolution*. Sinauer, 2013.

Gartman, David. *From Autos to Architecture: Fordism and Architectural Aesthetics in the Twentieth Century*. New York: Princeton Architectural Press, 2010.

Gibson, P. M., and J. A. Byrne. "NEUROGEN, Musical Composition Using Genetic Algorithms and Cooperating Neural Networks." In *1991 Second International Conference on Artificial Neural Networks*, 309–13, 1991.

Goldberg, David E. "Genetic Algorithms as a Computational Theory of Conceptual Design." In *Applications of Artificial Intelligence in Engineering VI*, 3–16. Springer, 1991. https://link.springer.com/chapter/10.1007/978-94-011-3648-8_1.

Goldberg, David Edward. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Boston [u.a.: Addison-Wesley, 2012.

Goldschmidt, Gabriela. "The Dialectics of Sketching." *Creativity Research Journal* 4, no. 2 (January 1, 1991): 123–43. https://doi.org/10.1080/10400419109534381.

Guo, Zifeng, and Biao Li. “Evolutionary Approach for Spatial Architecture Layout Design Enhanced by an Agent-Based Topology Finding System.” *Frontiers of Architectural Research* 6, no. 1 (March 1, 2017): 53–62. https://doi.org/10.1016/j.foar.2016.11.003.

Habraken, N. John. “The Control of Complexity.” *Places* 4, no. 2 (1987).

Hassan, Mohsen MD, Gary L. Hogg, and Donald R. Smith. “SHAPE: A Construction Algorithm for Area Placement Evaluation.” *International Journal of Production Research* 24, no. 5 (1986): 1283–1295.

Hillier, Bill. “The Art of Place and the Science of Space.” *World Architecture* 185 (2005): 96–102.

Holland, John H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT press, 1992.

Huang, Chuen-huei Joseph, and Robert Krawczyk. “A Choice Model of Consumer Participatory Design for Modular Houses,” 2007.

Jagielski, Romuald, and John S. Gero. “A Genetic Programming Approach to the Space Layout Planning Problem.” In *CAADFutures 1997: Proceedings of the 7th International Conference on Computer Aided Architectural Design Futures*, 875–84. CAAD Futures. Kluwer Academic Publishers, 1997.

Jo, Jun H., and John S. Gero. “Space Layout Planning Using an Evolutionary Approach.” *Artificial Intelligence in Engineering* 12, no. 3 (1998): 149–162.

Kalay, Yehuda E. *Architecture’s New Media: Principles, Theories, and Methods of Computer-Aided Design*. MIT Press, 2004.

Knecht, Katja, and Reinhard König. “Generating Floor Plan Layouts with Kd Trees and Evolutionary Algorithms.” In *Generative Art Conf*, 238–253, 2010.

Koopmans, Tjalling C., and Martin Beckmann. “Assignment Problems and the Location of Economic Activities.” *Econometrica: Journal of the Econometric Society*, 1957, 53–76.

Koza, John R., Forrest H. Bennett, David Andre, Martin A. Keane, and Frank Dunlap. “Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming.” *IEEE Transactions on Evolutionary Computation* 1, no. 2 (1997): 109–128.

Lawson, Bryan. *How Designers Think : The Design Process Demystified*. Oxford ; Burlington, MA : Elsevier/Architectural, 2006., 2006.

Le Corbusier. *Towards a New Architecture*. London, Architectural Press [1946],

1946.

Lee, Geun-Cheol, and Yeong-Dae Kim. “Algorithms for Adjusting Shapes of Departments in Block Layouts on the Grid-Based Plane.” *Omega* 28, no. 1 (February 1, 2000): 111–22. https://doi.org/10.1016/S0305-0483(99)00034-1.

Lee, Yanki. “Design Participation Tactics: Redefining User Participation in Design.” In *Design Research Society International Conference*, 2006.

Liang, Lou Y., and Wen C. Chao. “The Strategies of Tabu Search Technique for Facility Layout Optimization.” *Automation in Construction* 17, no. 6 (2008): 657–669.

Liu, Y.-C., A. Chakrabarti, and T. Bligh. “Towards an ‘Ideal’Approach for Concept Generation.” *Design Studies* 24, no. 4 (2003): 341–355.

Madrigal, Alexis C. “Inside Waymo’s Secret World for Training Self-Driving Cars.” *The Atlantic*, August 23, 2017. https://www.theatlantic.com/technology/archive/2017/08/inside-waymos-secret-testing-and-simulation-facilities/537648/.

Manyika, James, Michael Chui, Mehdi Miremadi, Jacques Bughin, Katy George, Paul Willmott, and Martin Dewhurst. “Harnessing Automation for a Future That Works.” *New York: McKinsey Global Institute*, 2017.

Marcus, Clare Cooper. *Easter Hill Village: Some Social Implications of Design*. Free Press, 1975.

McGrady, Vanessa. “New Study: Artificial Intelligence Is Coming For Your Job, Millennials.” Forbes, June 9, 2017. https://www.forbes.com/sites/vanessamcgrady/2017/06/09/millennial-jobs/.

McLeish, Thomas John. “A Platform for Consumer Driven Participative Design of Open (Source) Buildings.” Thesis, Massachusetts Institute of Technology, 2003. http://dspace.mit.edu/handle/1721.1/32250.

Meller, Russell D., and Kai-Yin Gau. “The Facility Layout Problem: Recent and Emerging Trends and Perspectives.” *Journal of Manufacturing Systems* 15, no. 5 (1996): 351–366.

Michalek, Jeremy, Ruchi Choudhary, and Panos Papalambros. “Architectural Layout Design Optimization.” *Engineering Optimization* 34, no. 5 (2002): 461–484.

Mittal, Sanjay, and Felix Frayman. “Towards a Generic Model of Configuraton Tasks.” In *IJCAI*, 89:1395–1401, 1989.

“New Construction Homes for Sale | Toll Brothers® Luxury Homes.” Accessed December 25, 2017. https://www.tollbrothers.com/.

Nourian, Pirouz. “Configraphics: Graph Theoretical Methods for Design and Analysis of Spatial Configurations.” *A+ BE| Architecture and the Built Environment* 6, no. 14 (2016): 1–348.

“Quinta Monroy / ELEMENTAL.” ArchDaily, December 31, 2008. http://www.archdaily.com/10775/quinta-monroy-elemental/.

Quiroz, J. C., S. J. Louis, A. Banerjee, and S. M. Dascalu. “Towards Creative Design Using Collaborative Interactive Genetic Algorithms.” In *2009 IEEE Congress on Evolutionary Computation*, 1849–56, 2009. https://doi.org/10.1109/CEC.2009.4983166.

Renee Y. Chow. “House Form and Choice.” *Traditional Dwellings and Settlements Review*, no. 2 (1998): 51.

Rittel, Horst WJ, and Melvin M. Webber. “Dilemmas in a General Theory of Planning.” *Policy Sciences* 4, no. 2 (1973): 155–69.

Rodrigues, Eugénio. “Automated Floor Plan Design: Generation, Simulation, and Optimization; Desenho Automático de Plantas: Geração, Simulação e Optimização.” Universidade de Coimbra, 2014. http://oatd.org/oatd/record?record=handle%5C%3A10316%5C%2F25438.

Rodrigues, Eugénio, Adélio Rodrigues Gaspar, and Álvaro Gomes. “An Approach to the Multi-Level Space Allocation Problem in Architecture Using a Hybrid Evolutionary Technique.” *Automation in Construction* 35 (2013): 482–498.

Rosenman, M. A. “The Generation of Form Using an Evolutionary Approach.” In *Evolutionary Algorithms in Engineering Applications*, 69–85. Springer, 1997. http://link.springer.com/chapter/10.1007/978-3-662-03423-1_4.

Russell, Stuart J., and Peter Norvig. *Artificial Intelligence : A Modern Approach*. Prentice Hall Series in Artificial Intelligence. Englewood Cliffs, N.J. : Prentice Hall, 2010., 1995.

Russell, Stuart, and Peter Norvig. “A Modern Approach.” *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs* 25 (1995): 27.

Schön, Donald A. *The Reflective Practitioner : How Professionals Think in Action /*. New York : Basic Books, c1983.

Shea, Kristina, Andrew Sedgwick, and Giulio Antonuntto. “Multicriteria Optimization of Paneled Building Envelopes Using Ant Colony Optimization.” *Intelligent Computing in Engineering and Architecture*, 2006, 627–636.

Silver, David, and Demis Hassabis. “AlphaGo: Mastering the Ancient Game of Go with Machine Learning.” *Research Blog* (blog), January 27, 2016.

https://research.googleblog.com/2016/01/alphago-mastering-ancient-game-of-go.html.

Simon, Herbert A. “Rational Choice and the Structure of the Environment.” *Psychological Review* 63, no. 2 (1956): 129.

“Simplifying and Improving GAAP.” Accessed January 14, 2018. http://www.accountingfoundation.org/jsp/Foundation/Page/FAFBridgePage&cid=1176164540272.

Skiena, Steven S. *The Algorithm Design Manual*. 2nd ed. Springer Publishing Company, Incorporated, 2008.

Stiny, George. *Shape : Talking about Seeing and Doing*. Cambridge, Massachusetts : The MIT Press, [2006], 2006.

Talbi, El-Ghazali. *Metaheuristics : From Design to Implementation*. Hoboken, N.J. : John Wiley & Sons, c2009., 2009.

Ueda, Kazuhiro, Hitoshi Kitazawa, and Ikuo Harada. “CHAMP: Chip Floor Plan for Hierarchical VLSI Layout Design.” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 4, no. 1 (1985): 12–22.

Van Camp, Drew J., Michael W. Carter, and Anthony Vannelli. “A Nonlinear Optimization Approach for Solving Facility Layout Problems.” *European Journal of Operational Research* 57, no. 2 (1992): 174–189.

Vardouli, Theodora. “Who Designs?” In *Empowering Users through Design*, 13–41. Springer, 2015.

Venturi, Robert. *Complexity and Contradiction in Architecture*. The Museum of Modern Art Papers on Architecture. New York : Museum of Modern Art ; Boston : distributed by New York Graphic Society, 1977., 1977.

Verma, Manisha, and Manish K. Thakur. “Architectural Space Planning Using Genetic Algorithms.” In *Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference On*, 2:268–275. IEEE, 2010. http://ieeexplore.ieee.org/abstract/document/5451497/.

Virirakis, Lefteris. “GENETICA: A Computer Language That Supports General Formal Expression with Evolving Data Structures.” *IEEE Transactions on Evolutionary Computation* 7, no. 5 (2003): 456–481.

Von Hippel, Eric, and Ralph Katz. “Shifting Innovation to Users via Toolkits.” *Management Science* 48, no. 7 (2002): 821–833.

Whitehouse, Mark, Mira Rojanasakul, and Cedric Sam. “Is Your Job About To Disappear?: Quicktake.” *Bloomberg.Com*, June 22, 2017.

https://www.bloomberg.com/graphics/2017-jobs-automation-risk/.

“WikiHouse.” WikiHouse. Accessed December 19, 2017. https://wikihouse.cc/.

Williams, Reid E. (Reid Edward). “Training Architectural Computational Critics by Example.” Thesis, Massachusetts Institute of Technology, 2003. http://dspace.mit.edu/handle/1721.1/16691.

Yi, Hwang, and Yun Kyu Yi. “Performance Based Architectural Design Optimization: Automated 3D Space Layout Using Simulated Annealing.” American Society of Heating, Refrigeration, and Air-Conditioning Engineers (ASHRAE), 2014. https://experts.illinois.edu/en/publications/performance-based-architectural-design-optimization-automated-3d-.

Zha, Xuanfang, and Wen F. Lu. “Knowledge Support for Customer-Based Design for Mass Customization.” In *Artificial Intelligence in Design ’02*, 407–29. Springer, Dordrecht, 2002. https://doi.org/10.1007/978-94-017-0795-4_20.

APPENDIX A

FIGURES FOR CASE STUDY 1 – COMPACTNESS D

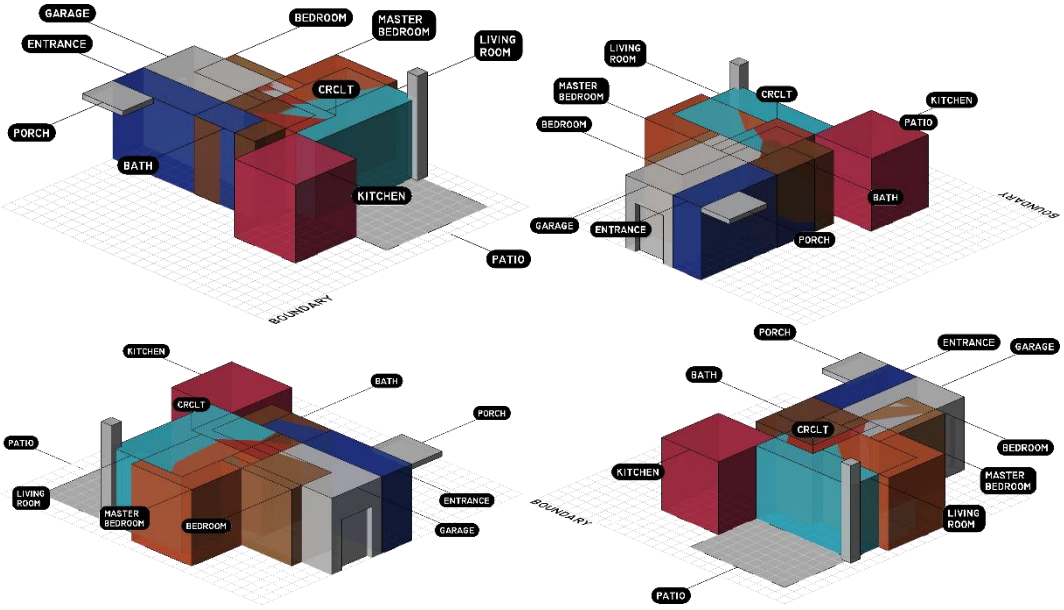


Figure 91. Best layout solution for generation 20 – fitness: 0.062, C_{ovf}:0.000, C_{int}:0.440, C_{dim}:0.500, C_{rel}:0.000, C_{comp}:0.00, C_{cant}:0.00, C_{circ}:0.300, C_{view}:0.000 – Parallel projection from four sides - Case study 1 – Compactness D. (Drawn by the author)

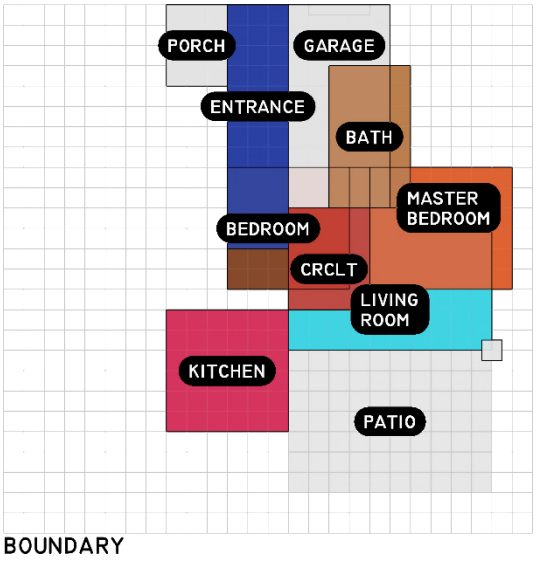


Figure 92. Best layout solution for generation 20 – Top View - Case study 1– Compactness D. (Drawn by the author)

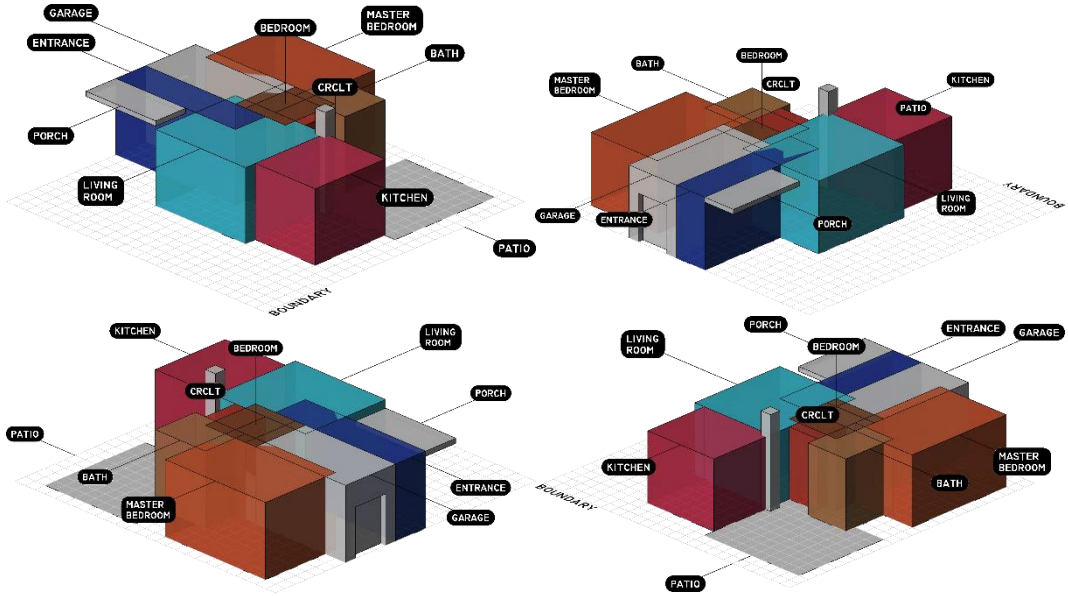


Figure 93. Best layout solution for generation 40 – fitness: 0.047, C_{ovf}:0.000, C_{int}:0.200, C_{dim}:0.300, C_{rel}:0.000, C_{comp}:0.00, C_{cant}:0.00, C_{circ}:0.172, C_{view}:0.000 – Parallel projection from four sides - Case study 1 – Compactness D. (Drawn by the author)

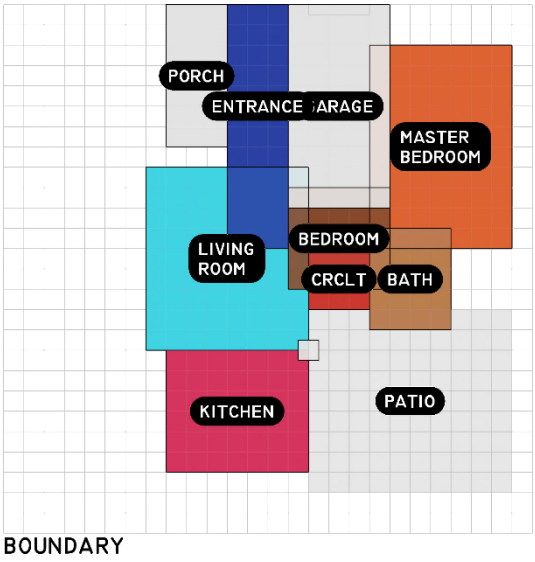


Figure 94. Best layout solution for generation 40 – Top View - Case study 1 – Compactness D. (Drawn by the author)

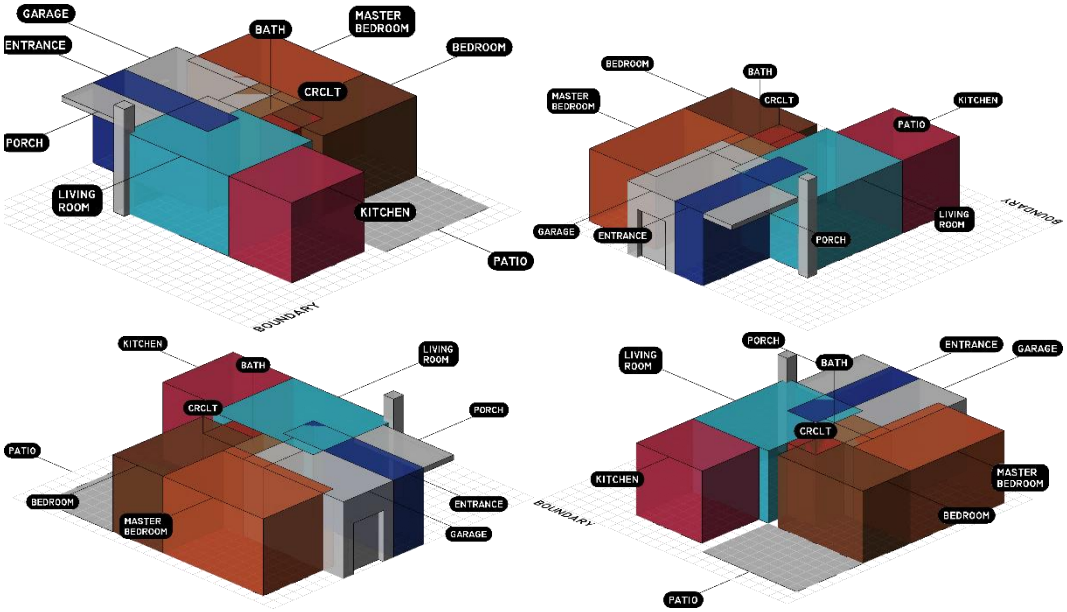


Figure 95. Best layout solution for generation 60 – fitness: 0.041, C_{ovf}:0.000, C_{int}:0.266, C_{dim}:0.401, C_{rel}:0.000, C_{comp}:0.000, C_{cant}:0.000, C_{circ}:0.172, C_{view}:0.000 – Parallel projection from four sides - Case study 1 – Compactness D. (Drawn by the author)

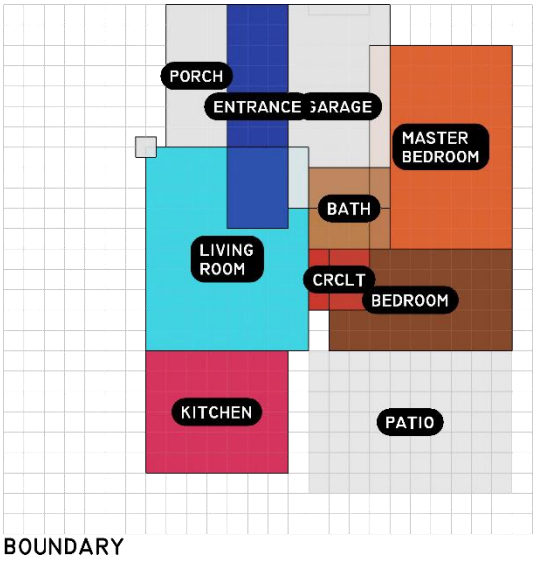


Figure 96. Best layout solution for generation 60 – Top View - Case study 1 – Compactness D. (Drawn by the author)

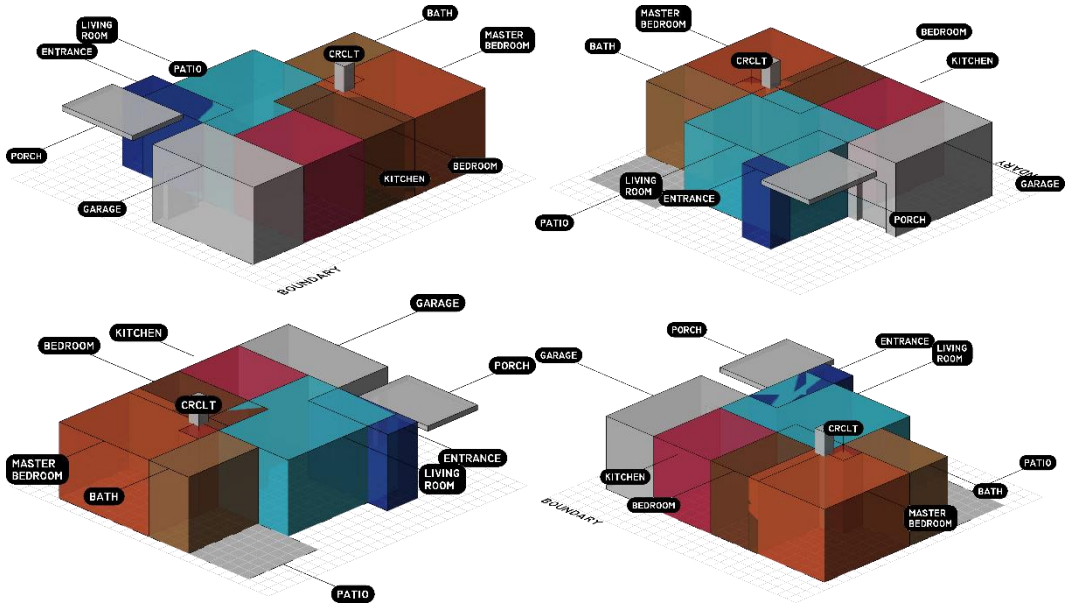


Figure 97. Best layout solution for generation 80 – fitness: 0.038, C_{ovf}:0.000, C_{int}:0.241, C_{dim}:0.383, C_{rel}:0.05, C_{comp}:0.000, C_{cant}:0.000, C_{circ}:0.172, C_{view}:0.000 – Parallel projection from four sides - Case study 1 – Compactness D. (Drawn by the author)

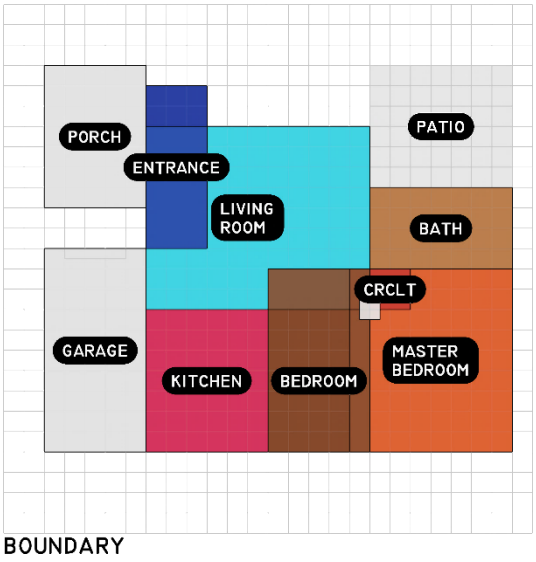


Figure 98. Best layout solution for generation 80 – Top View - Case study 1 – Compactness D. (Drawn by the author)

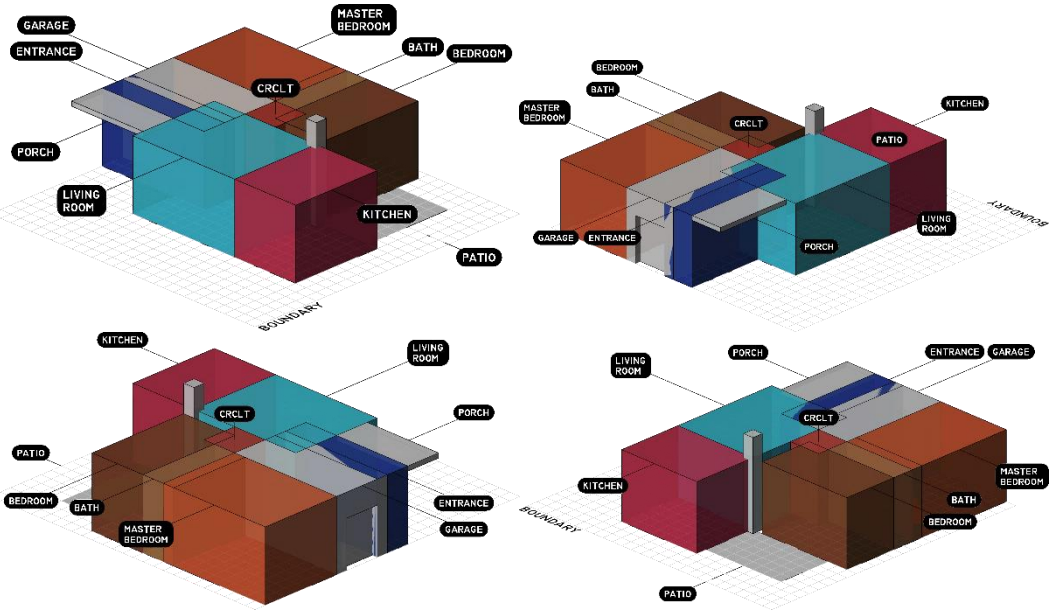


Figure 99. Best layout solution for generation 100 – fitness: 0.037, C_{ovf}:0.000, C_{int}:0.237, C_{dim}:0.334, C_{rel}:0.000, C_{comp}:0.000, C_{cant}:0.000, C_{circ}:0.172, C_{view}:0.000 – Parallel projection from four sides - Case study 1 – Compactness D. (Drawn by the author)

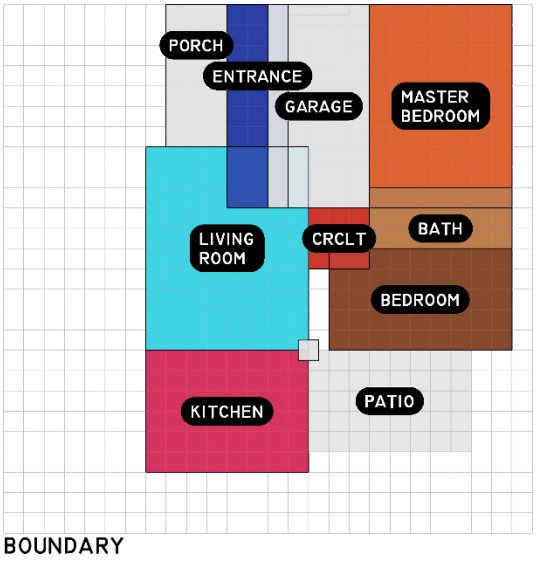


Figure 100. Best layout solution for generation 100 – Top View - Case study 1 – Compactness D. (Drawn by the author)

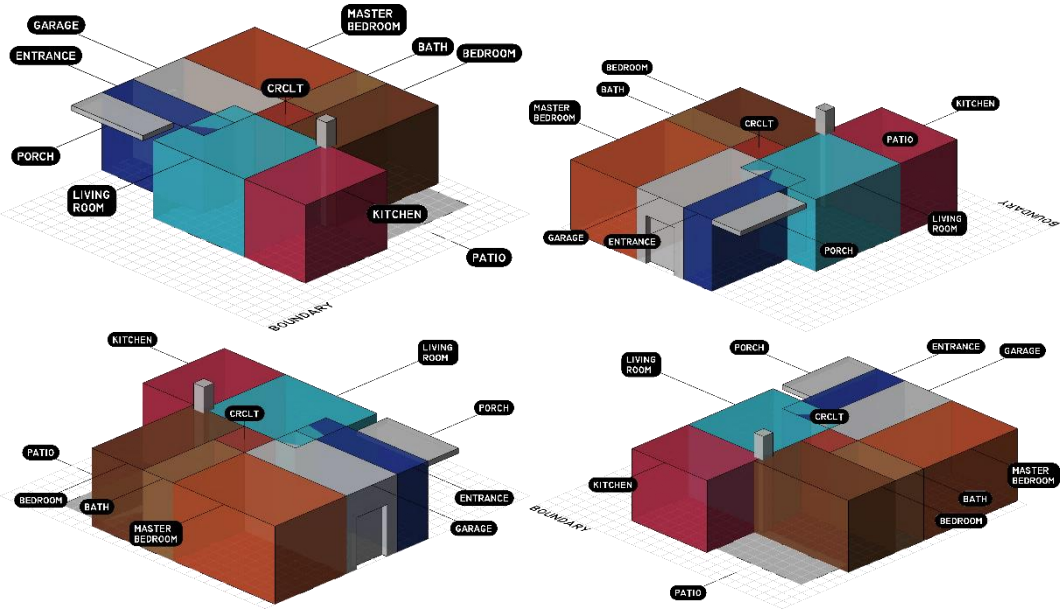


Figure 101. Best layout solution for generation 120 – fitness: 0.039, C_{ovf}:0.00, C_{int}:0.185, C_{dim}:0.366, C_{rel}:0.000, C_{comp}:0.000, C_{cant}:0.000, C_{circ}:0.172, C_{view}:0.000 – Parallel projection from four sides - Case study 1 – Compactness D. (Drawn by the author)

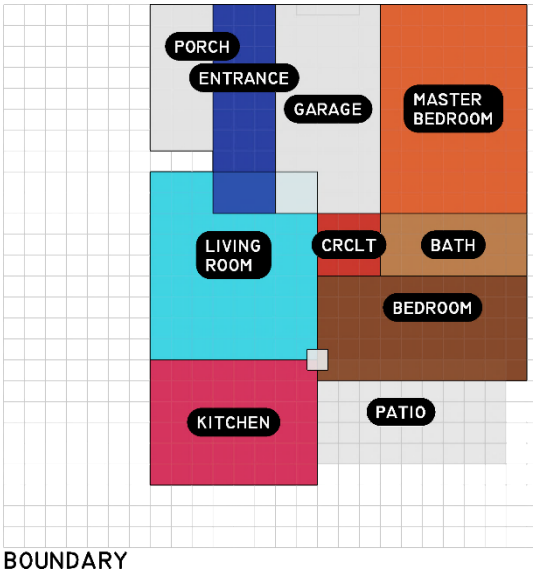


Figure 102. Best layout solution for generation 120 – Top View - Case study 1 – Compactness D. (Drawn by the author)

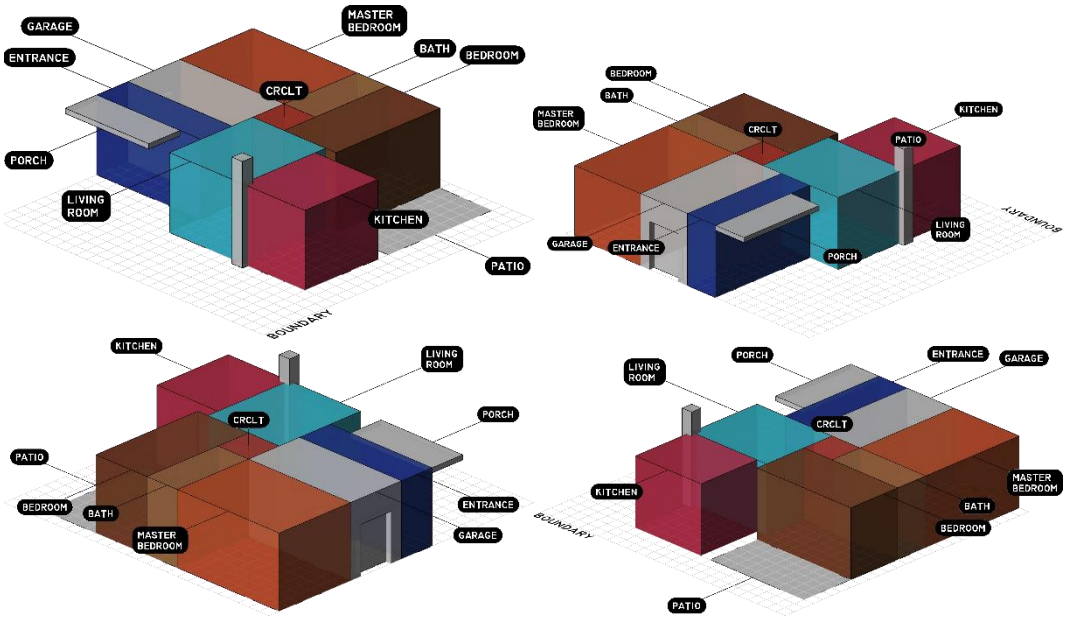


Figure 103. Best layout solution for generation 140 – fitness: 0.39, C_{ovf}:0.00, C_{int}:0.000, C_{dim}:0.395, C_{rel}:0.000, C_{comp}:0.118, C_{cant}:0.00, C_{circ}:0.172, C_{view}:0.000 – Parallel projection from four sides - Case study 1 – Compactness D. (Drawn by the author)

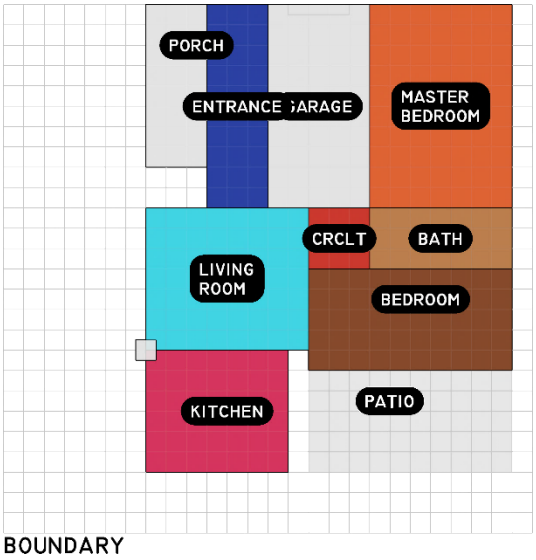


Figure 104. Best layout solution for generation 140 – Top View - Case study 1 – Compactness D. (Drawn by the author)

APPENDIX B

FIGURES FOR CASE STUDY 2 – COMPACTNESS A

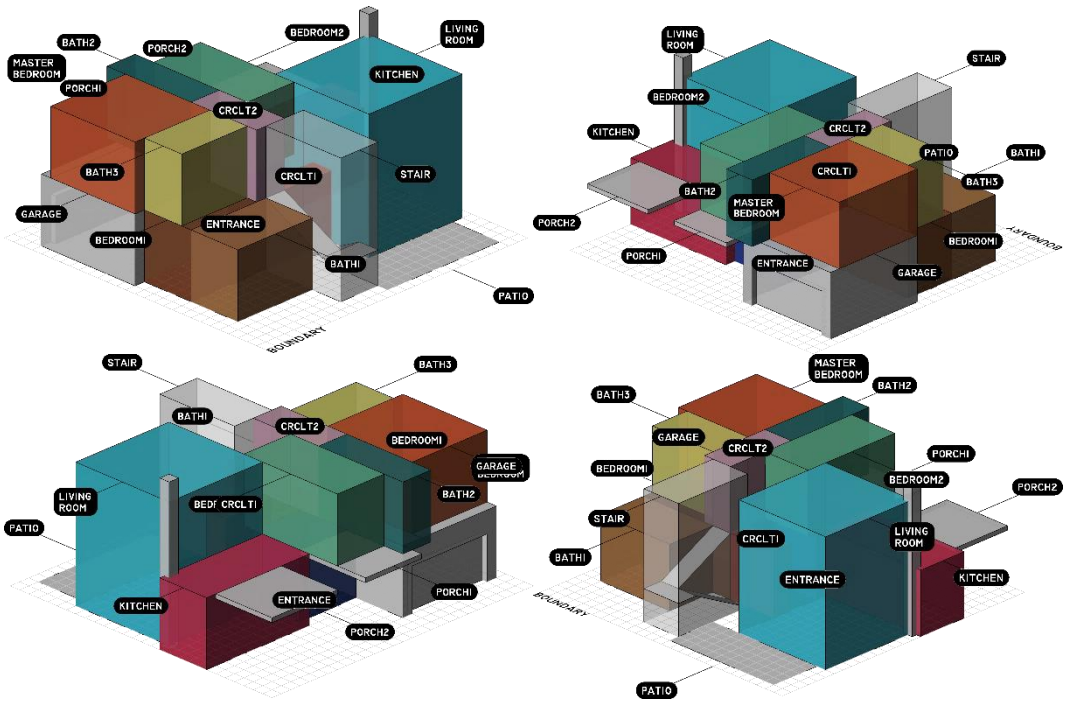


Figure 105. Best layout solution for generation 0 – fitness:0.390, C_{ovf}:0.695, C_{int}:0.646, C_{dim}:0.739, C_{rel}:0.160, C_{comp}:0.000, C_{cant}:0.000, C_{circ}:0.632, C_{view}:0.091 – Parallel projection from four sides - Case study 2 – Compactness A (Drawn by the author).

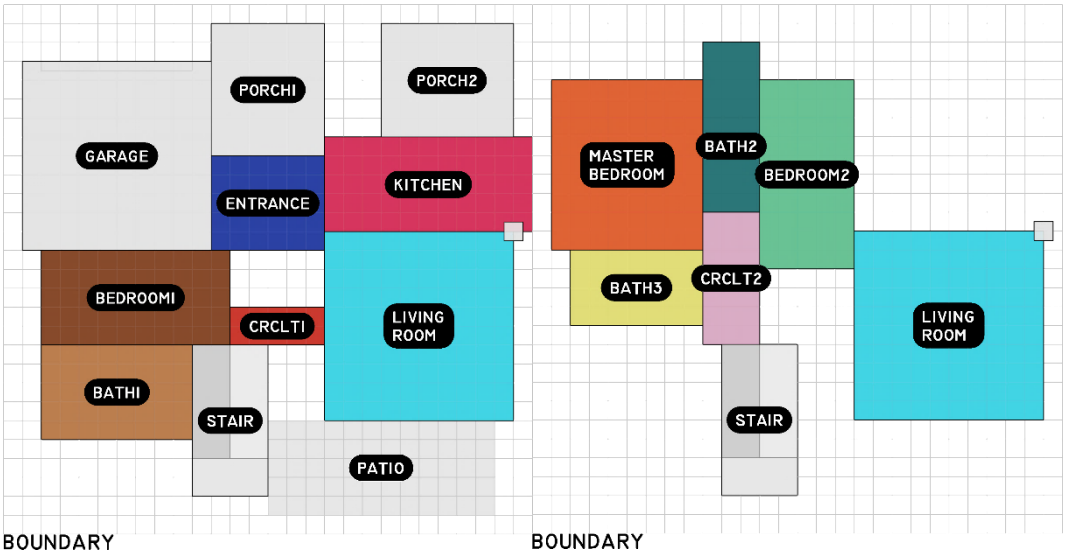


Figure 106. Best layout solution for generation 0 – Top View - Case study 2 – Compactness A. (Drawn by the author)

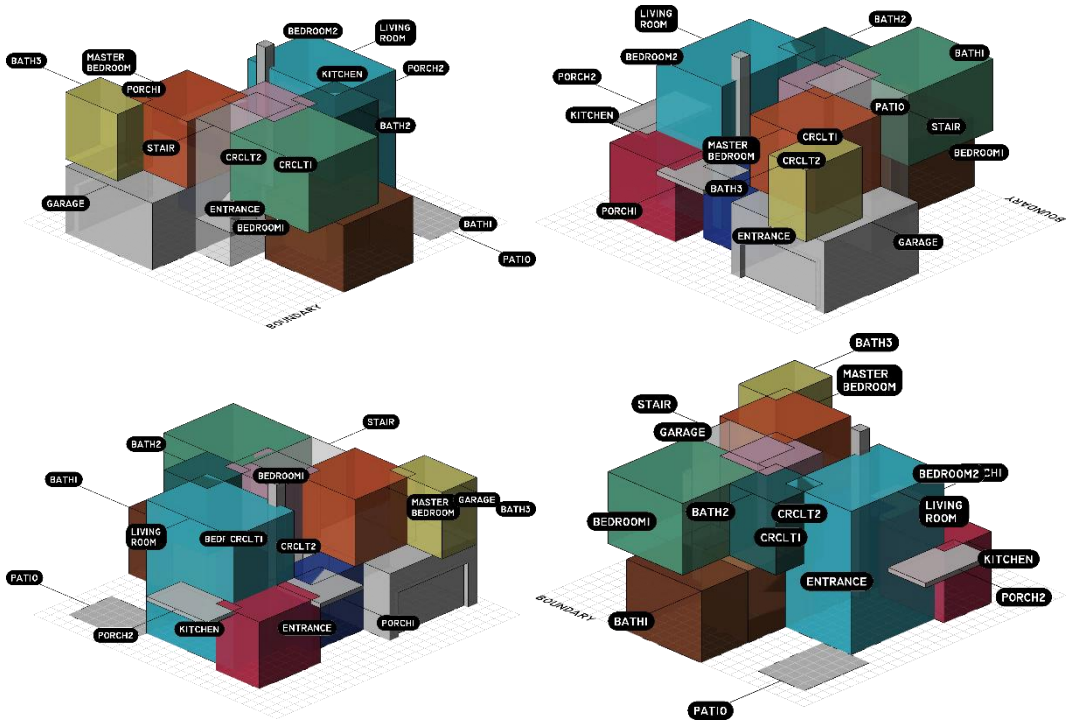


Figure 107. Best layout solution for generation 20 – fitness:0.105, C_{ovf}:0.000, C_{int}:0.311, C_{dim}:0.628, C_{rel}:0.007, C_{comp}:0.000, C_{cant}:0.000, C_{circ}:0.287, C_{view}:0.000 – Parallel projection from four sides - Case study 2 – Compactness A (Drawn by the author).

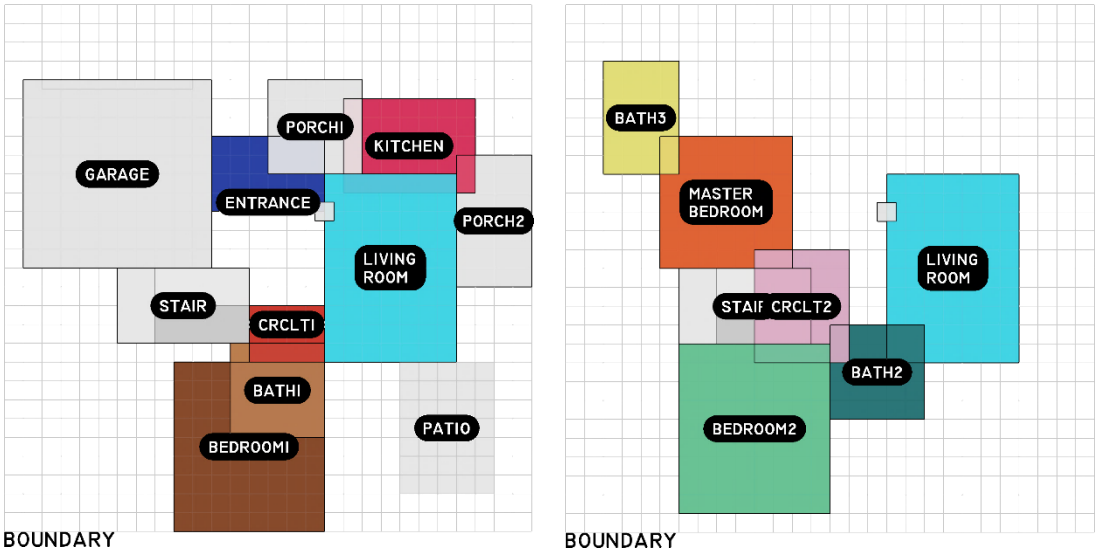


Figure 108. Best layout solution for generation 20 – Top View - Case study 2 – Compactness A. (Drawn by the author)

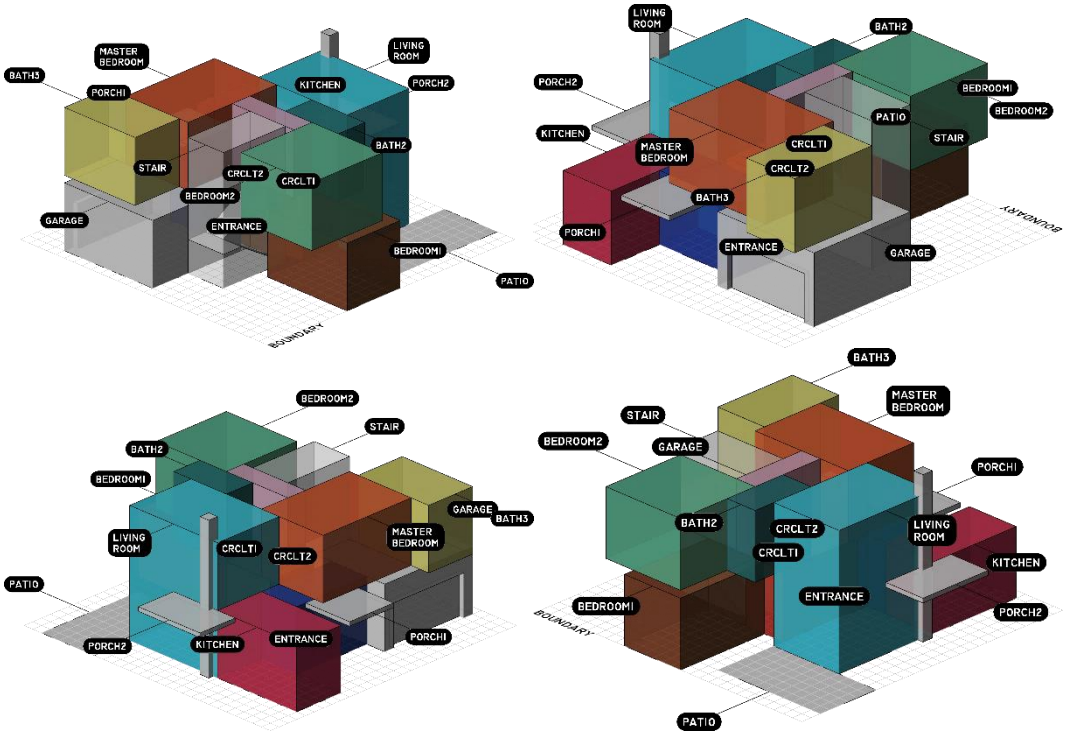


Figure 109. Best layout solution for generation 40 – fitness:0.048, C_{ovf}:0.000, C_{int}:0.000, C_{dim}:0.504, C_{rel}:0.001, C_{comp}:0.000, C_{cant}:0.000, C_{circ}:0.226, C_{view}:0.000 – Parallel projection from four sides - Case study 2 – Compactness A (Drawn by the author).

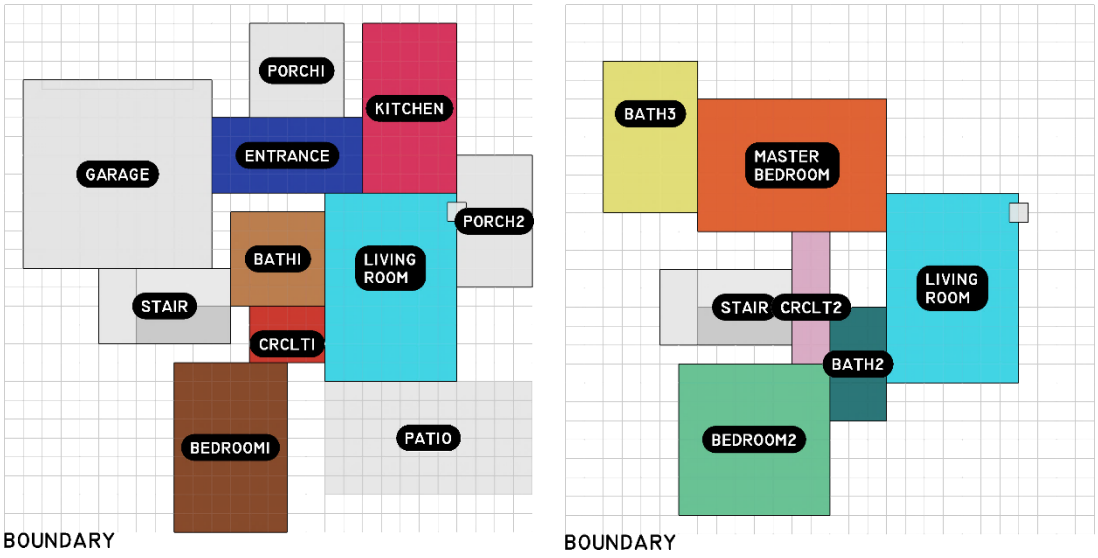


Figure 110. Best layout solution for generation 40 – Top View - Case study 2 – Compactness A. (Drawn by the author)

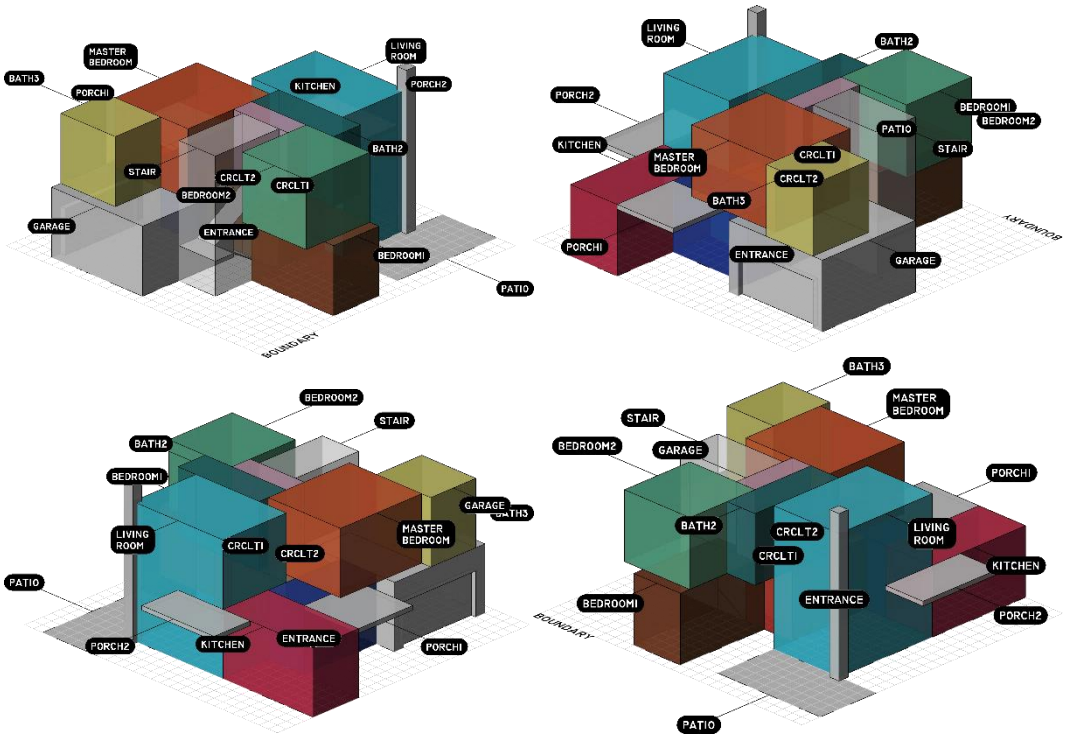


Figure 111. Best layout solution for generation 70 – fitness:0.033, C_{ovf}:0.000, C_{int}:0.000, C_{dim}:0.379, C_{rel}:0.000, C_{comp}:0.000, C_{cant}:0.000, C_{circ}:0.208, C_{view}:0.000 – Parallel projection from four sides - Case study 2 – Compactness A (Drawn by the author).

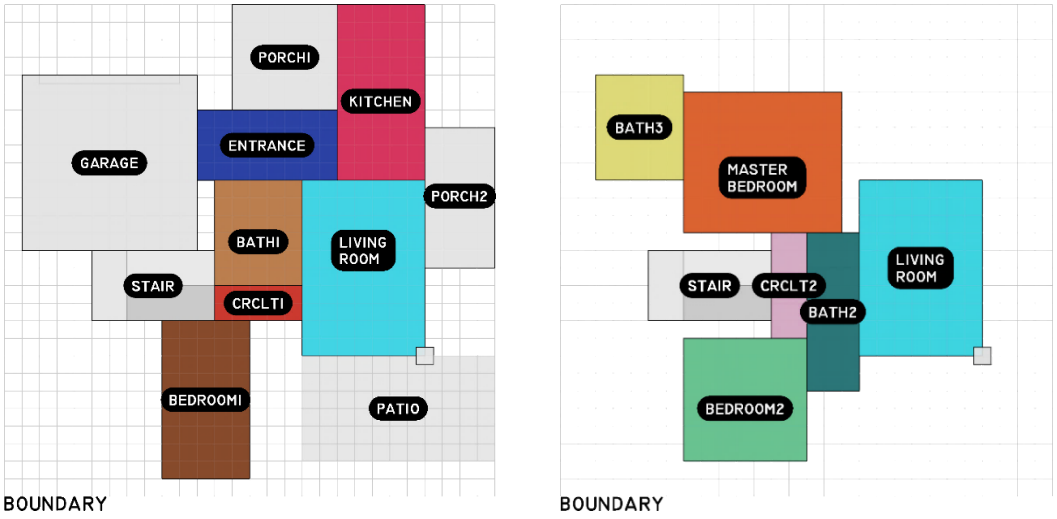


Figure 112. Best layout solution for generation 70 – Top View - Case study 2 – Compactness A (Drawn by the author).

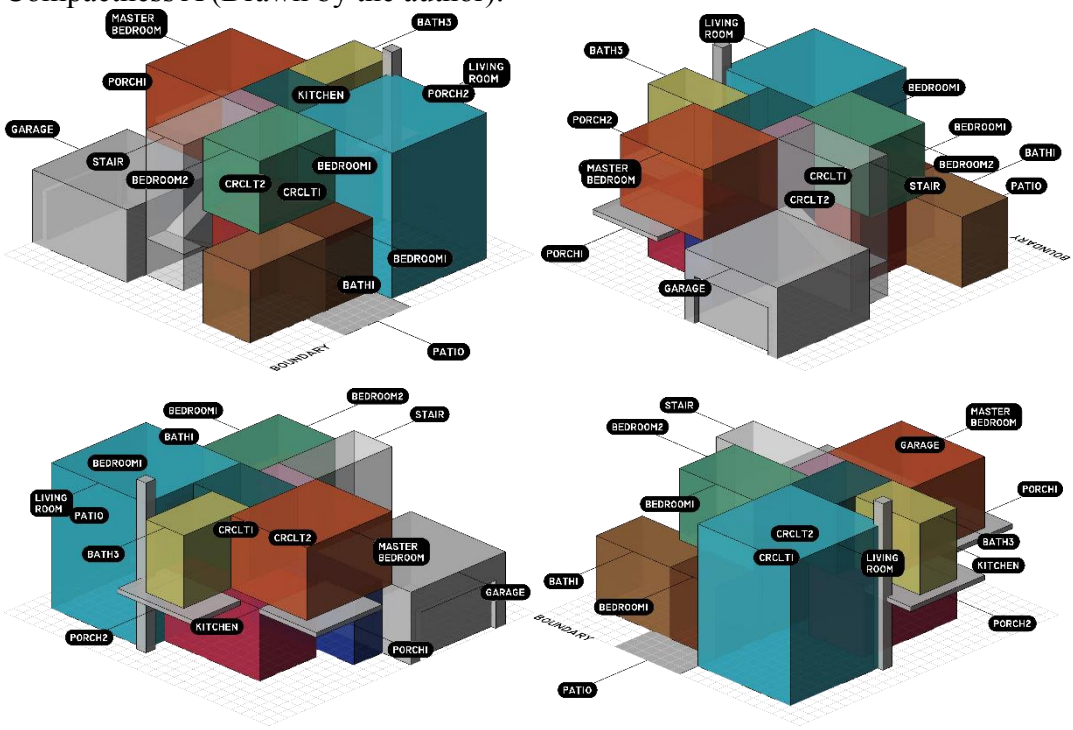


Figure 113. Alternative 1 - Case study 2a – Compactness A – Parallel projection from 4 sides (Drawn by the author).

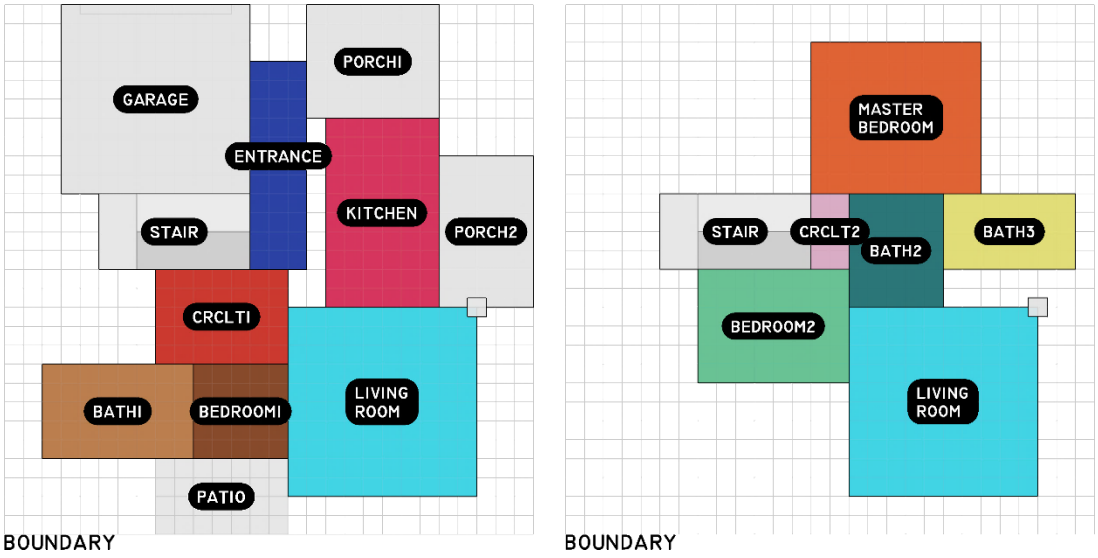


Figure 114. Alternative 1 - Case study 2a – Compactness A – Top view (Drawn by the author).

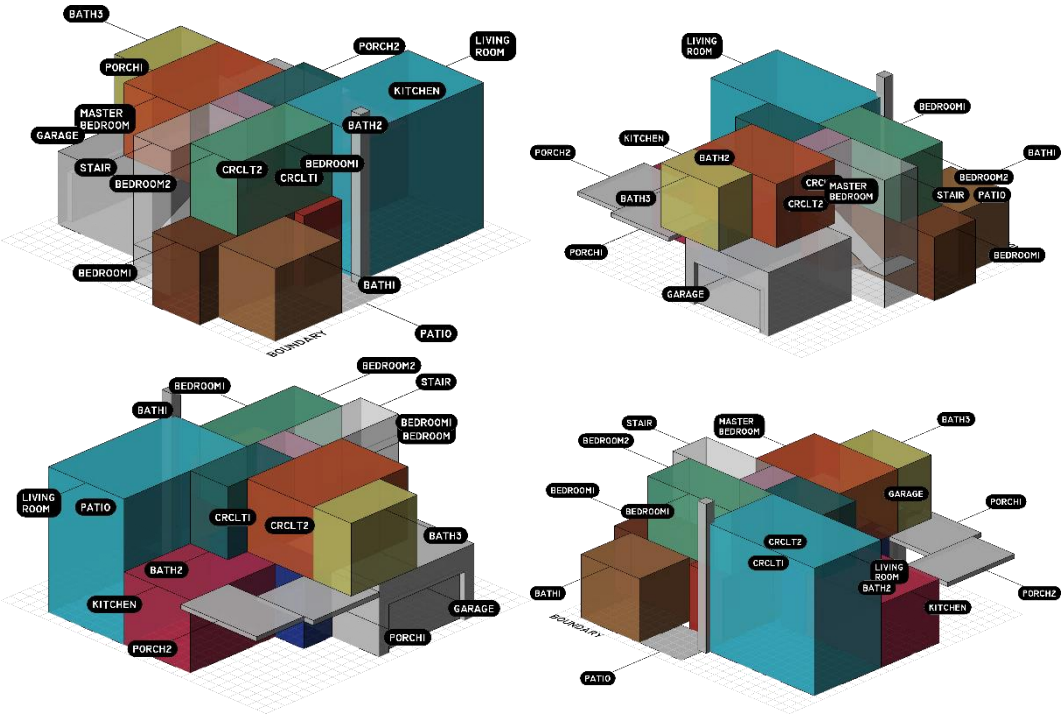


Figure 115. Alternative 2 - Case study 2a – Compactness A – Parallel projection from 4 sides (Drawn by the author).

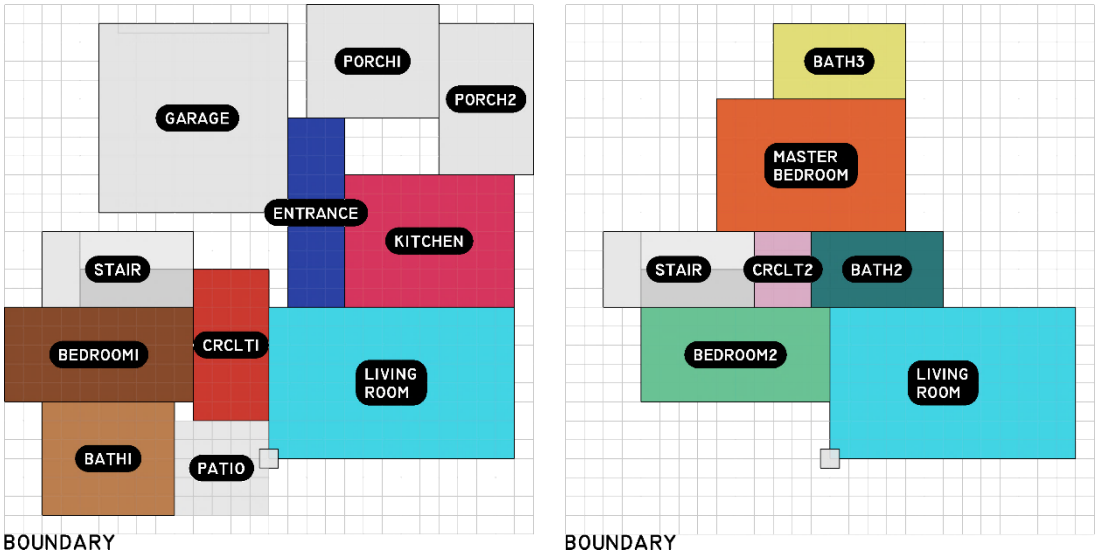


Figure 116. Alternative 2 - Case study 2a – Compactness A – Top view. (Drawn by the author)

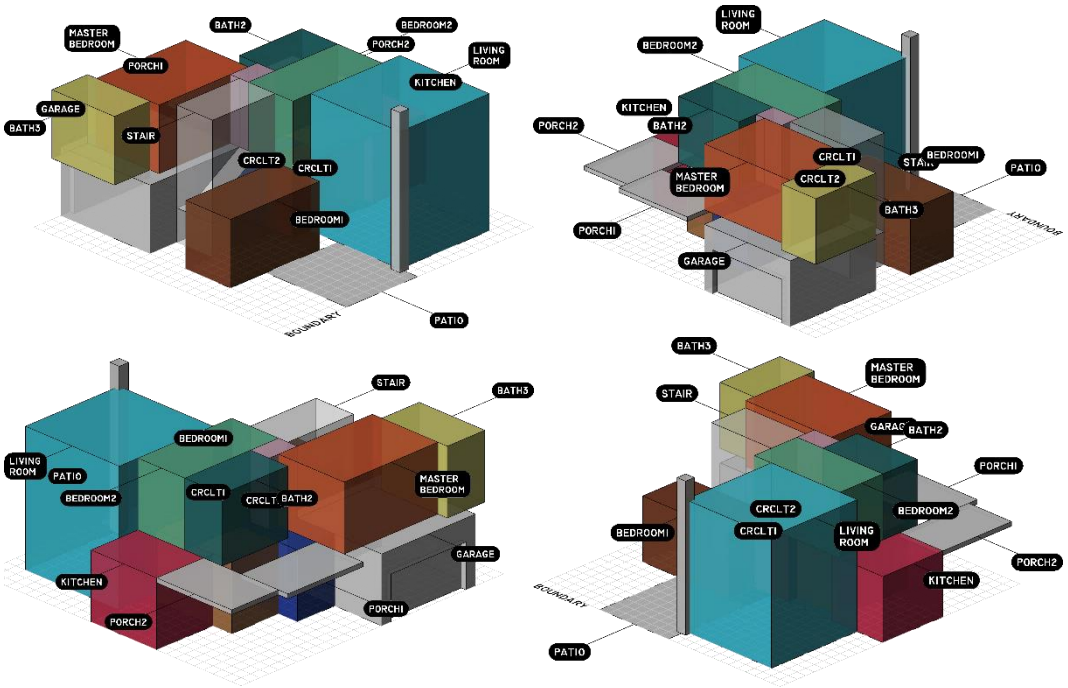


Figure 117. Alternative 3 - Case study 2a – Compactness A – Parallel projection from 4 sides (Drawn by the author).

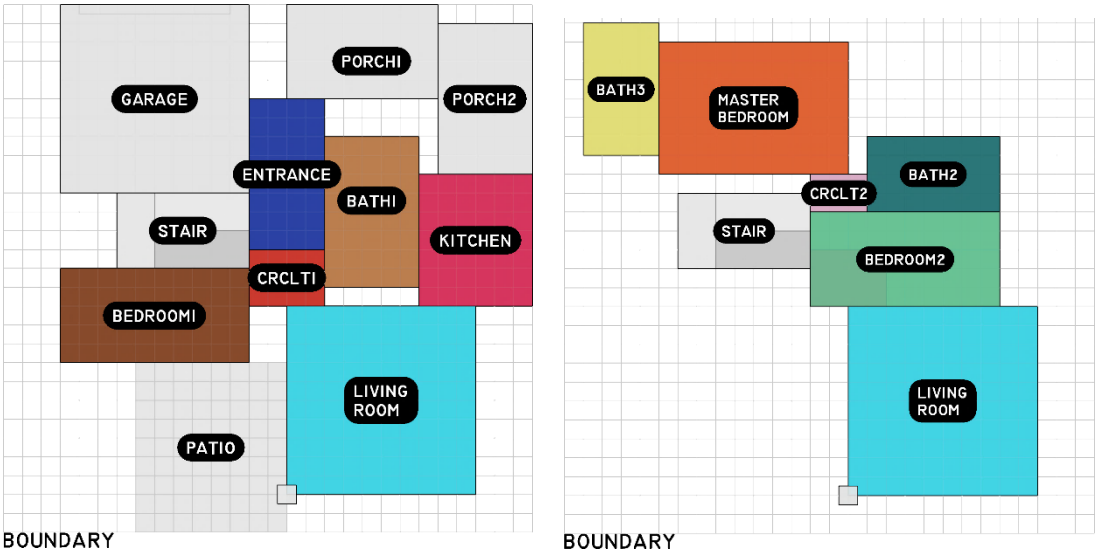


Figure 118. Alternative 3 - Case study 2a – Compactness A – Top view (Drawn by the author).

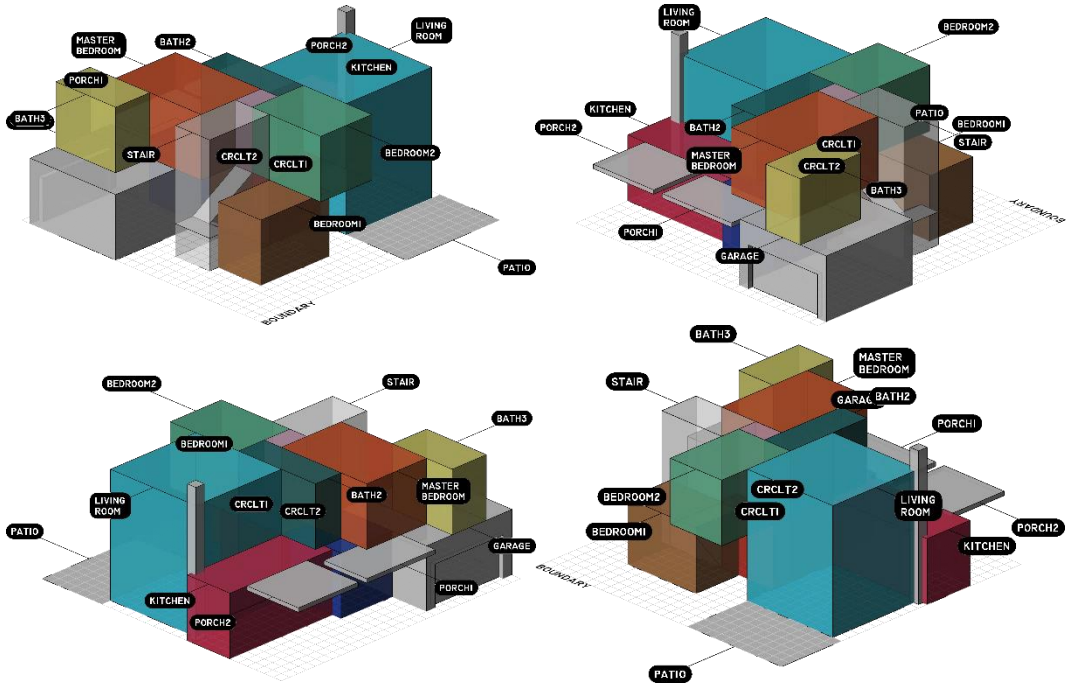


Figure 119. Alternative 4 - Case study 2a – Compactness A – Parallel projection from 4 sides (Drawn by the author).

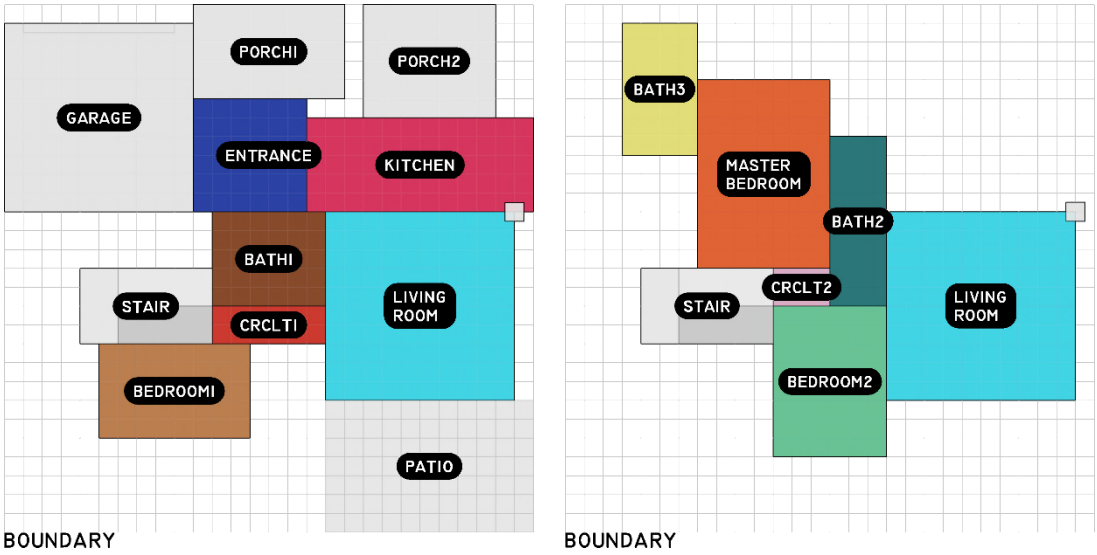


Figure 120. Alternative 4 - Case study 2a – Compactness A – Top view (Drawn by the author).

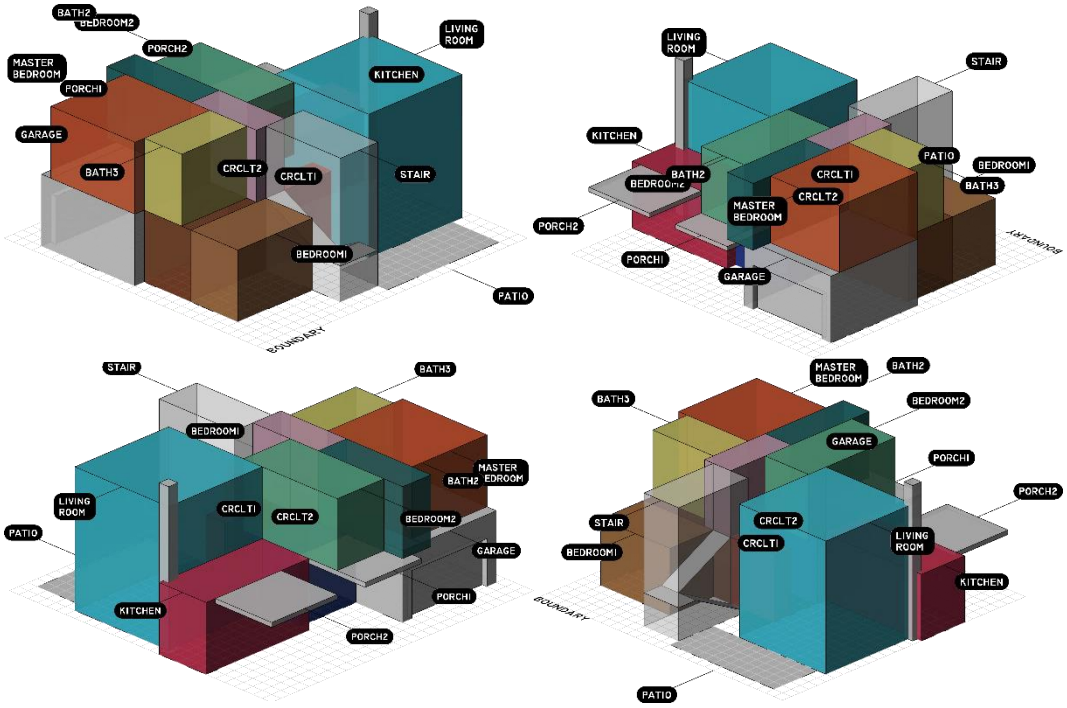


Figure 121. Alternative 5 - Case study 2a – Compactness A – Parallel projection from 4 sides (Drawn by the author).

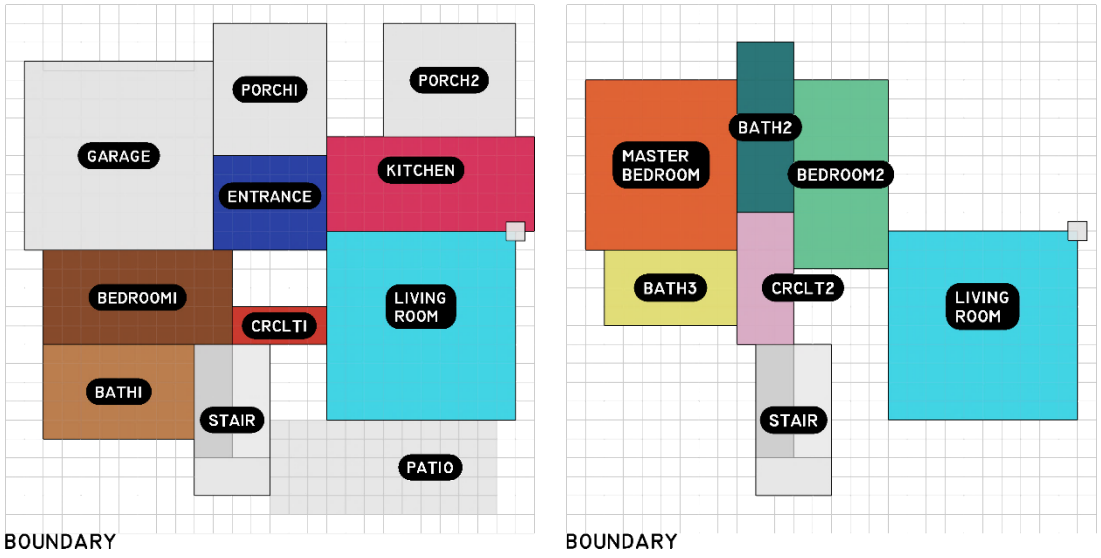


Figure 122. Alternative 5 - Case study 2 – Compactness A – Top view (Drawn by the author).

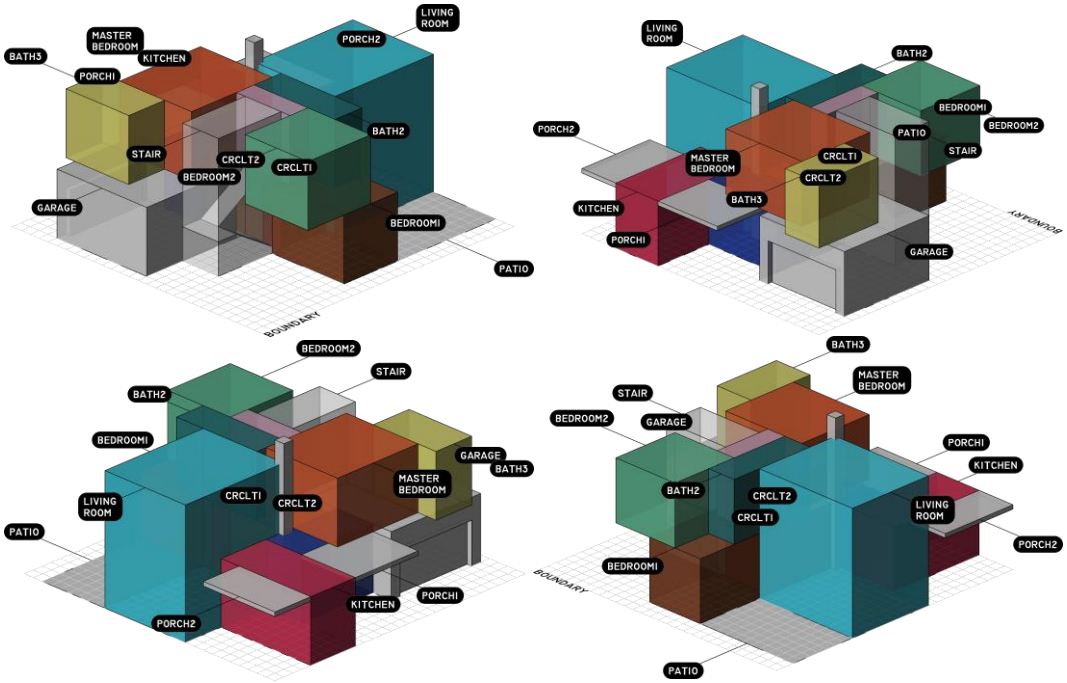


Figure 123. Alternative 6 - Case study 2 – Compactness A – Parallel projection from 4 sides (Drawn by the author).

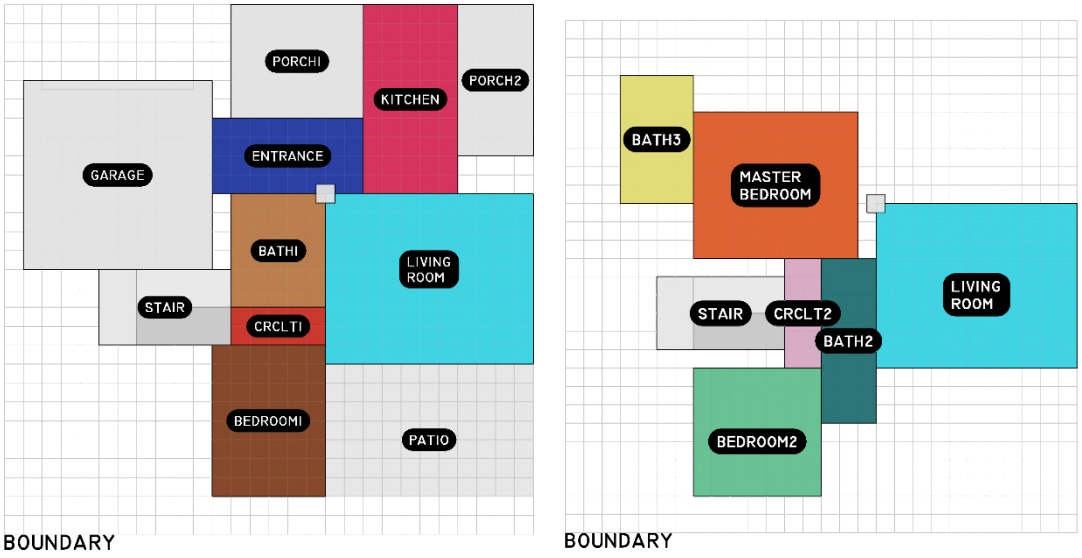


Figure 124. Alternative 6 - Case study 2 – Compactness A – Top view (Drawn by the author)

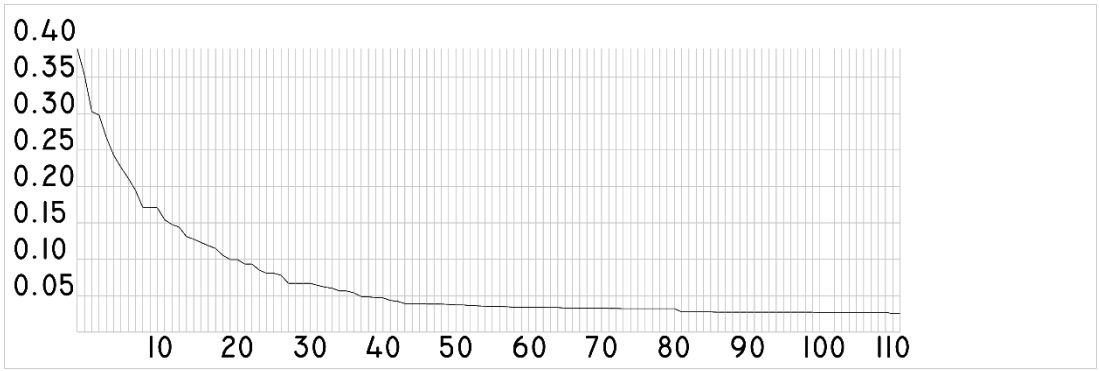


Figure 125. Best fitness score of total fitness - Case study 2 – Compactness A (Drawn by the author).

Figure 126. Best fitness score of evaluators - Case study 2 – Compactness A (Drawn by the author).

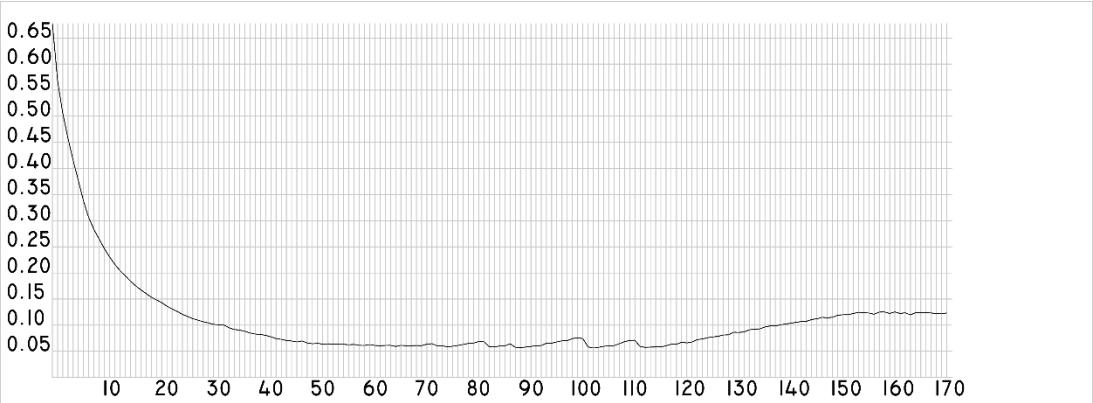


Figure 127. Average fitness score for total fitness - Case study 2 – Compactness A (Drawn by the author).

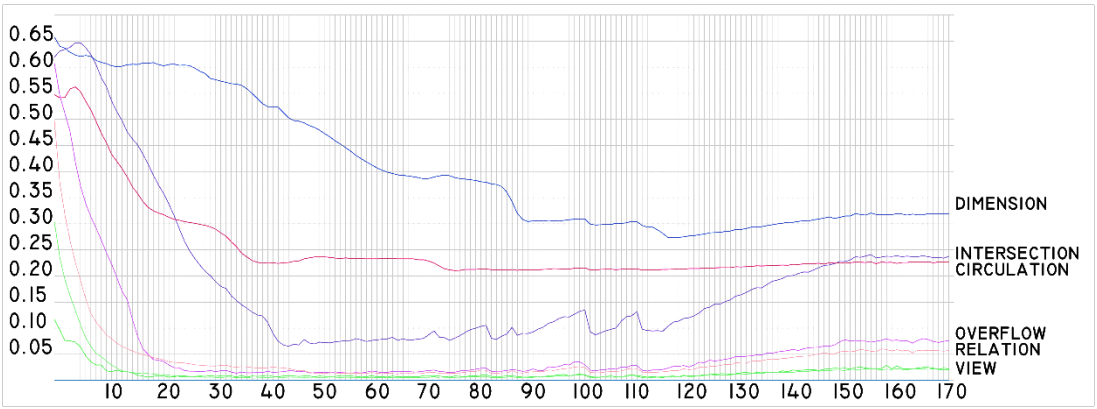


Figure 128. Average fitness score of evaluators - Case study 2 – Compactness A (Drawn by the author).

APPENDIX C

FIGURES FOR CASE STUDY 2 – COMPACTNESS C

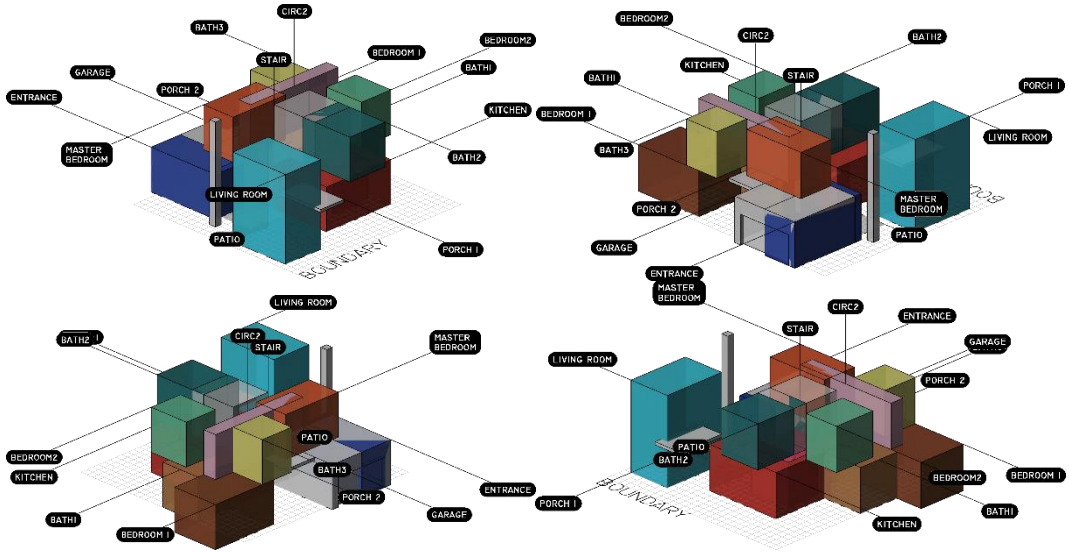


Figure 129. Best layout solution for generation 0 – fitness:0.44, C_{ovf}:0.43, C_{int}:0.75, C_{dim}:0.52, C_{rel}:0.27, C_{comp}:0.57, C_{cant}:0.00, C_{circ}:0.60, C_{view}:0.04 – Parallel projection from four sides - Case study 2 – Compactness C. (Drawn by the author)

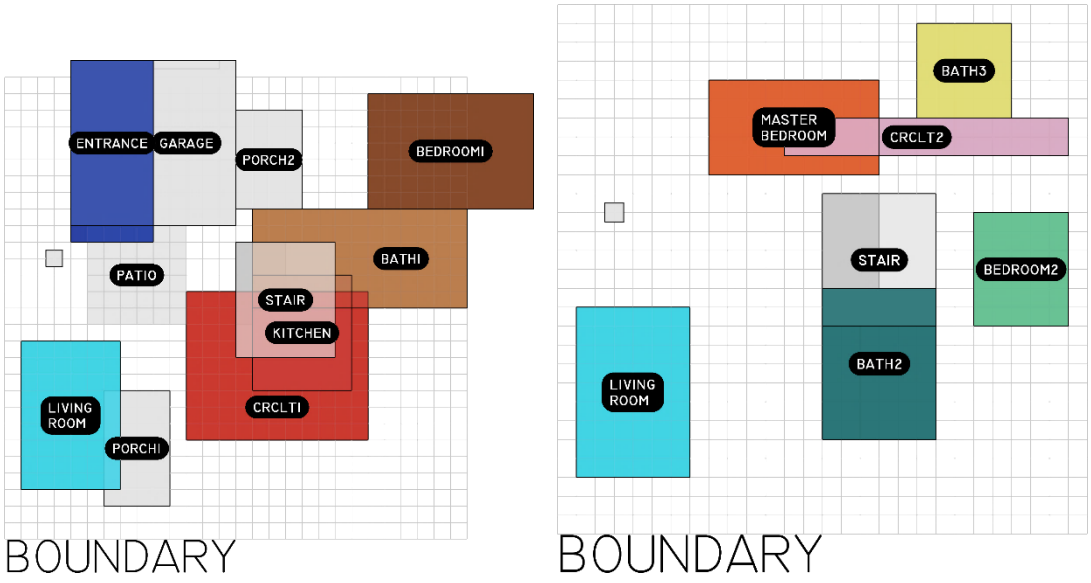


Figure 130. Best layout solution for generation 0 – Top View - Case study 2 – Compactness C. (Drawn by the author)

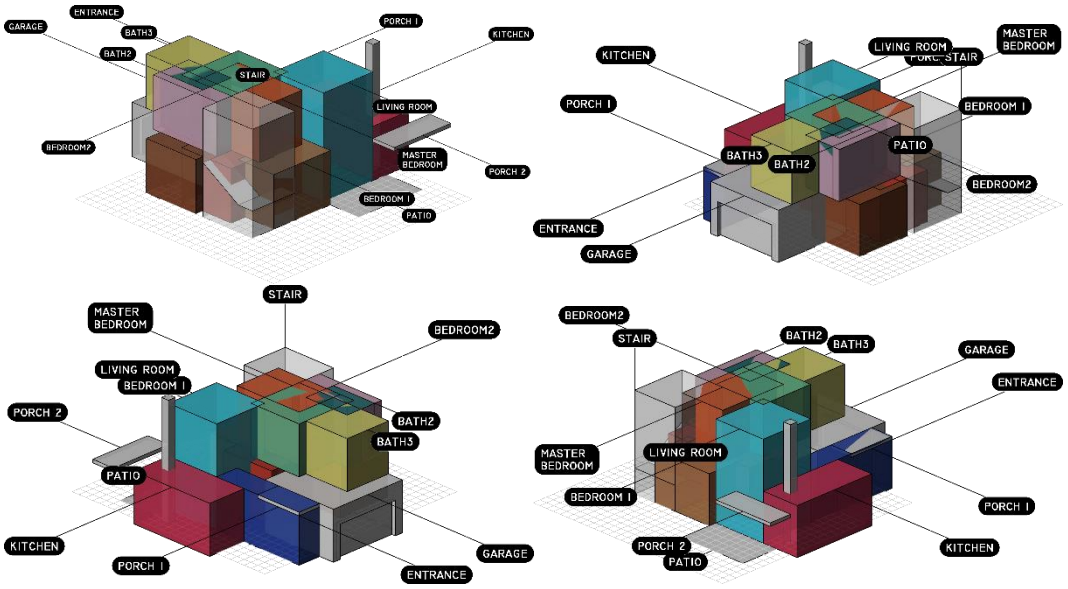


Figure 131. Best layout solution for generation 10 – fitness:0.20, C_{ovf}:0.27, C_{int}:0.52, C_{dim}:0.52, C_{rel}:0.03, C_{comp}:0.22, C_{chim}:0.39, C_{cant}:0.00, C_{circ}:0.51, C_{view}:0.00 – Parallel projection from four sides - Case study 2 – Compactness C. (Drawn by the author)

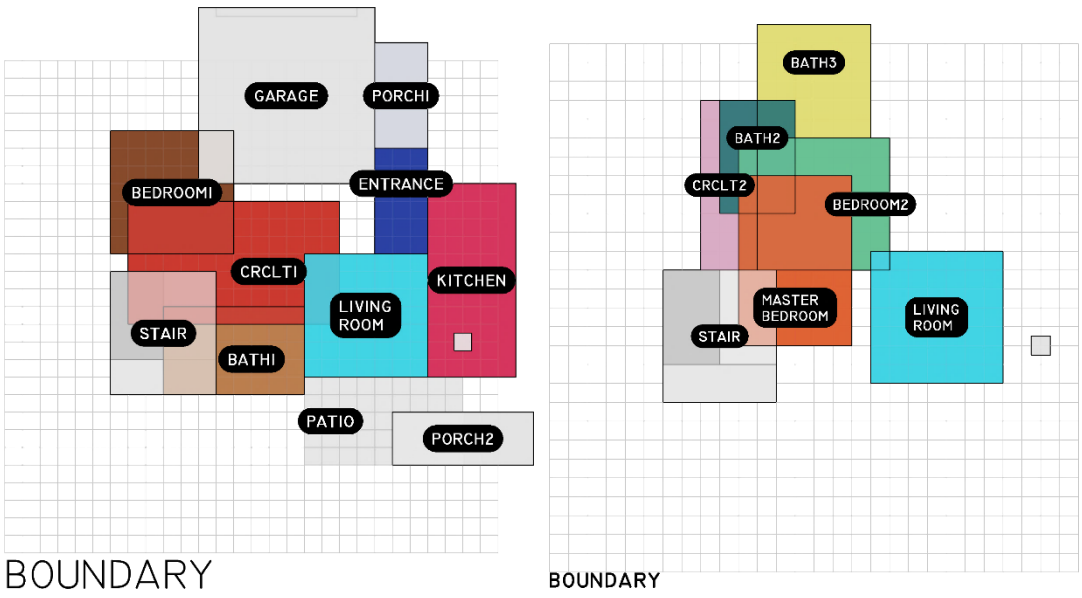


Figure 132. Best layout solution for generation 10 – Top View - Case study 2 – Compactness C. (Drawn by the author)

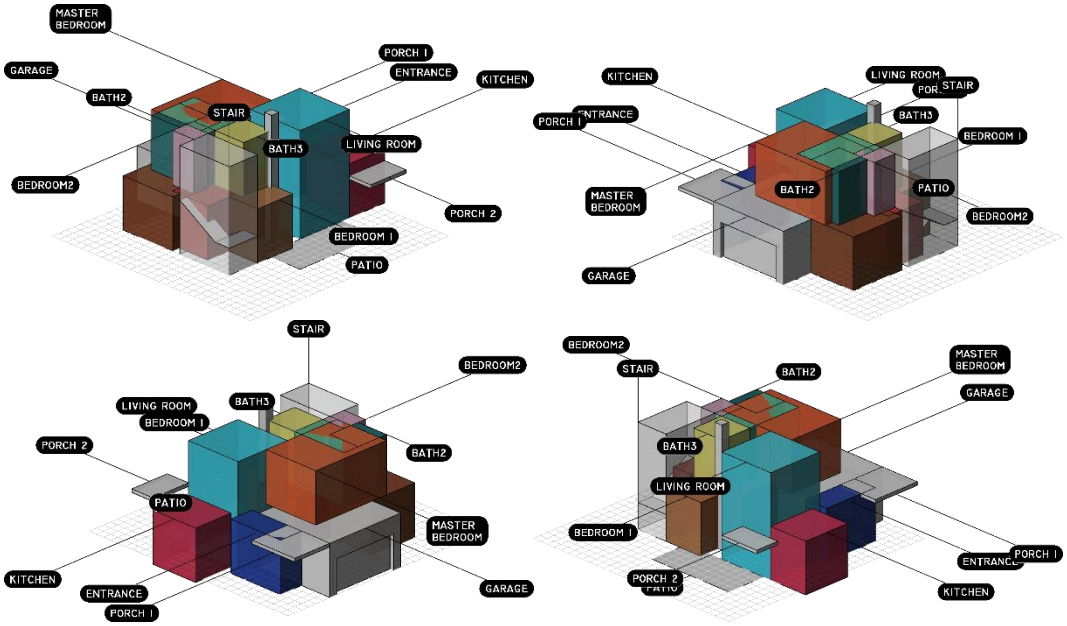


Figure 133. Best layout solution for generation 20 – fitness:0.15, C_{ovf}:0.09, C_{int}:0.34, C_{dim}:0.39, C_{rel}:0.02, C_{comp}:0.27, C_{cant}:0.00, C_{circ}:0.37, C_{view}:0.00 – Parallel projection from four sides - Case study 2 – Compactness C. (Drawn by the author)

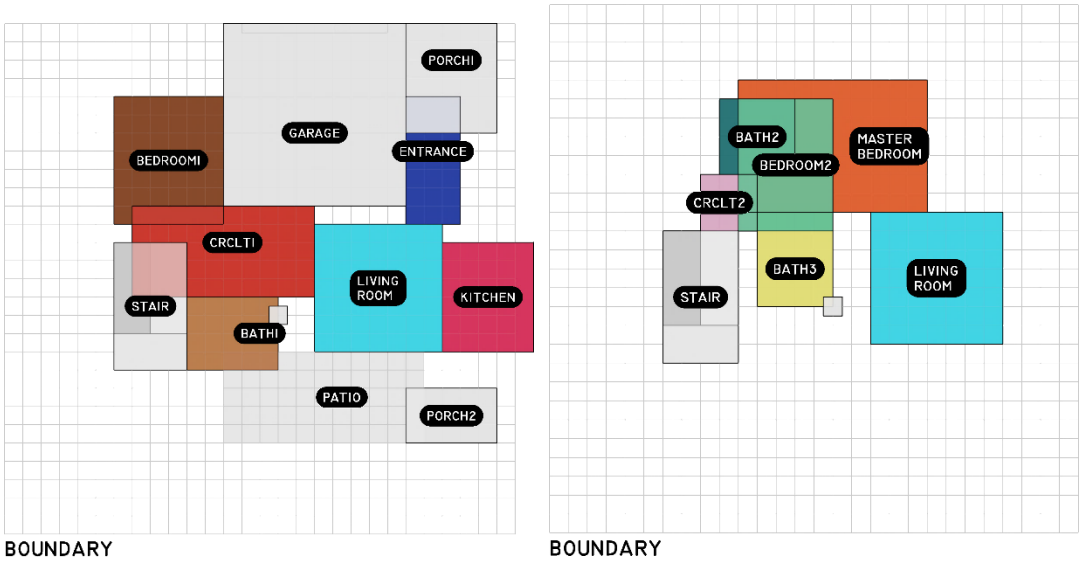


Figure 134. Best layout solution for generation 20 – Top View - Case study 2 – Compactness C. (Drawn by the author)

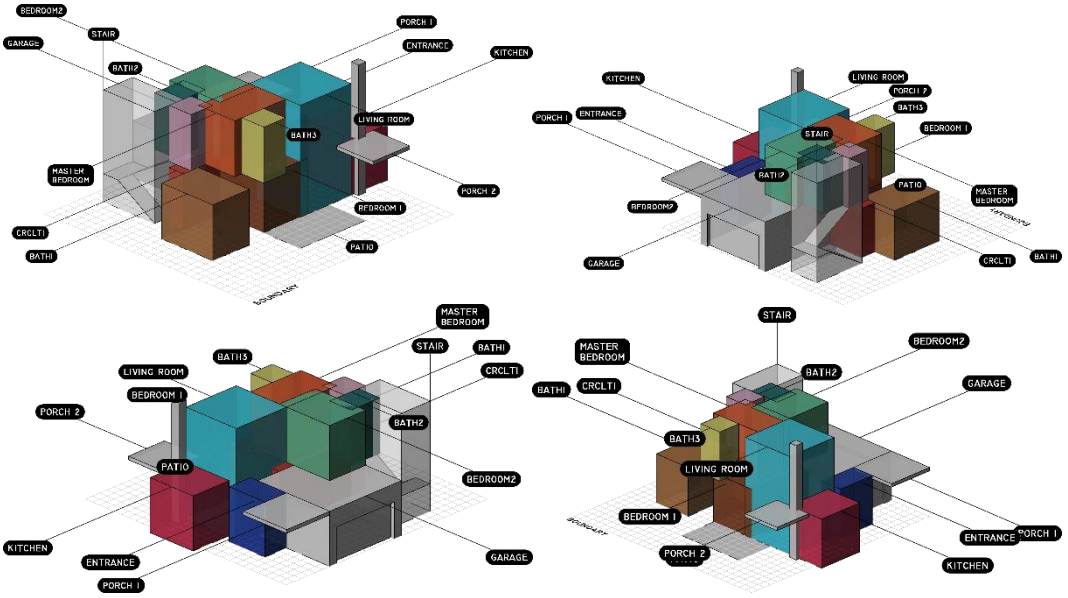


Figure 135. Best layout solution for generation 30 – fitness:0.12, C_{ovf}:0.09, C_{int}:0.19, C_{dim}:0.39, C_{rel}:0.01, C_{comp}:0.35, C_{cant}:0.00, C_{circ}:0.28, C_{view}:0.00 – Parallel projection from four sides - Case study 2 – Compactness C. (Drawn by the author)

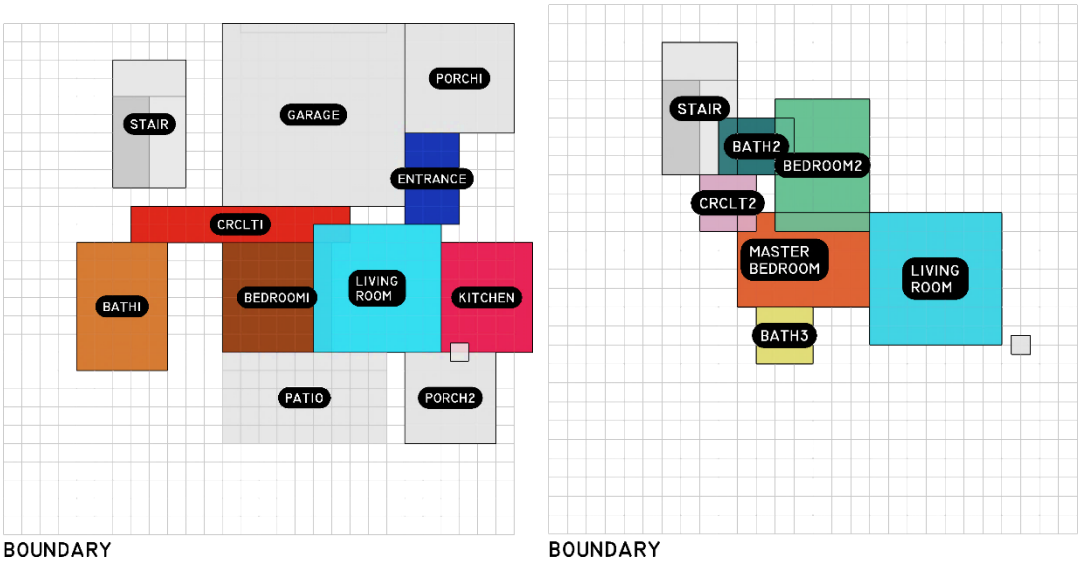


Figure 136. Best layout solution for generation 30 – Top View - Case study 2 – Compactness C. (Drawn by the author)

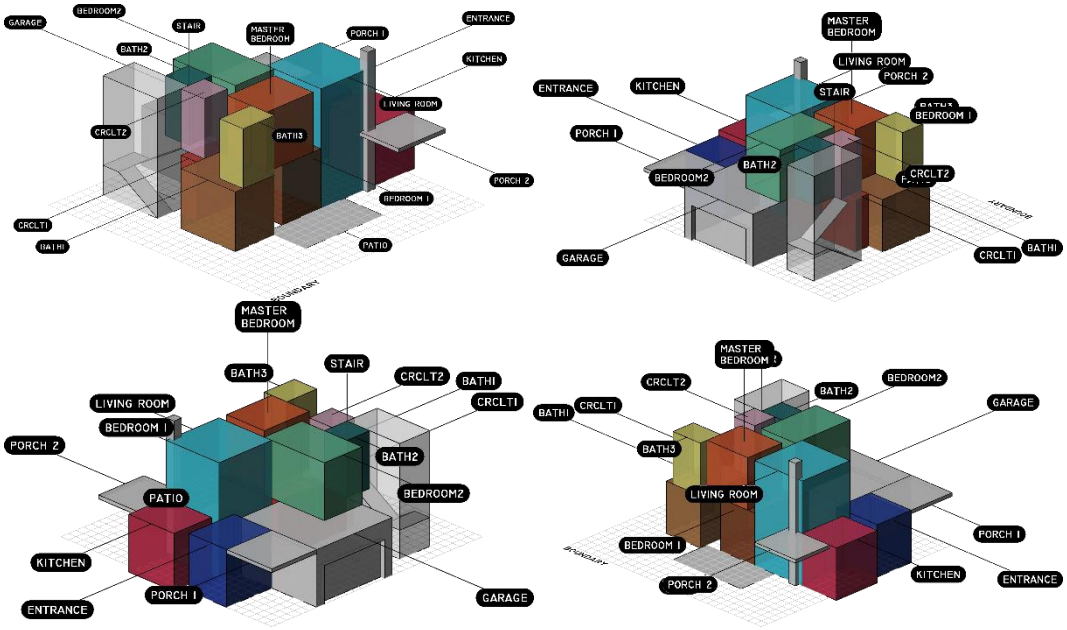


Figure 137. Best layout solution for generation 40 – fitness:0.08, C_{ovf}:0.09, C_{int}:0.00, C_{dim}:0.60, C_{rel}:0.03, C_{comp}:0.33, C_{cant}:0.00, C_{circ}:0.26, C_{view}:0.00 – Parallel projection from four sides - Case study 2 – Compactness C. (Drawn by the author)

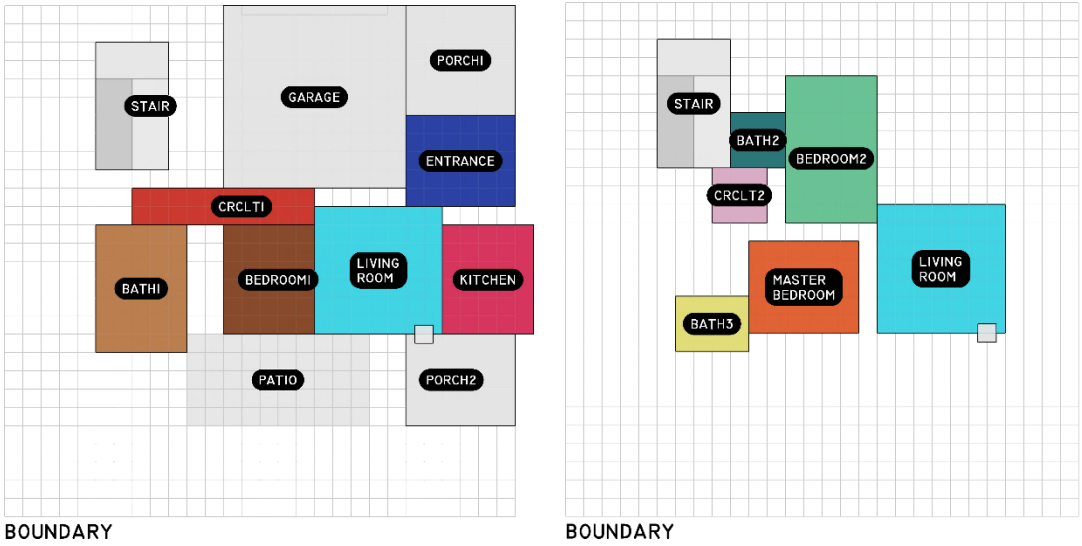


Figure 138. Best layout solution for generation 40 – Top View - Case study 2 – Compactness C. (Drawn by the author)

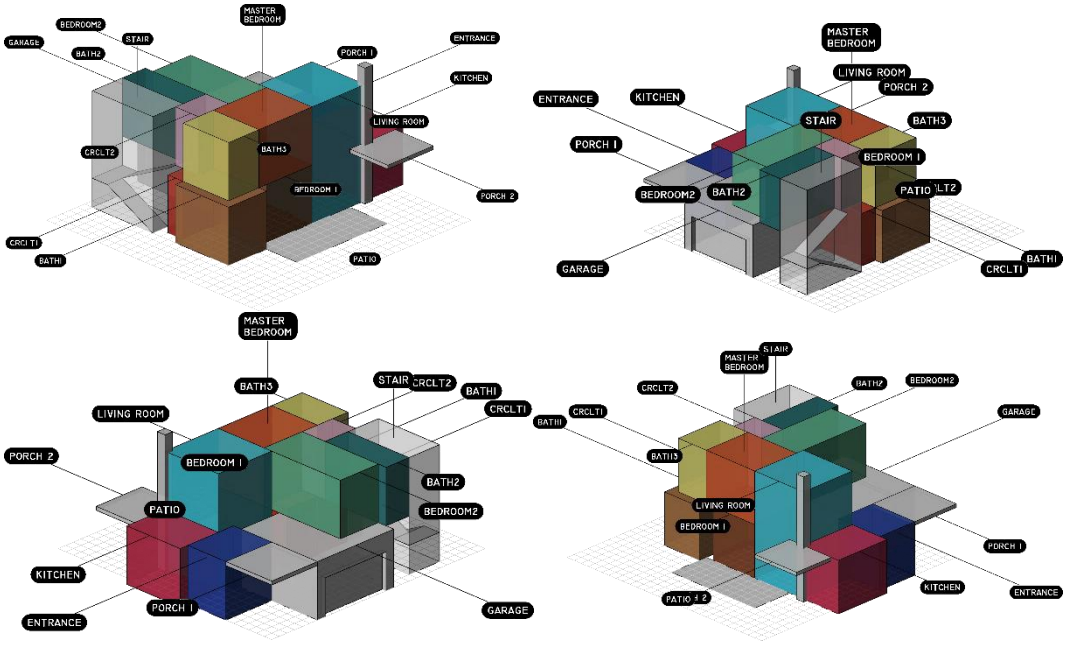


Figure 139. Best layout solution for generation 80 – fitness: 0.07, C_{ovf}:0.09, C_{int}:0.00, C_{dim}:0.53, C_{rel}:0.01, C_{comp}:0.27, C_{cant}:0.00, C_{circ}:0.27, C_{view}:0.00 – Parallel projection from four sides - Case study 2 – Compactness C. (Drawn by the author)

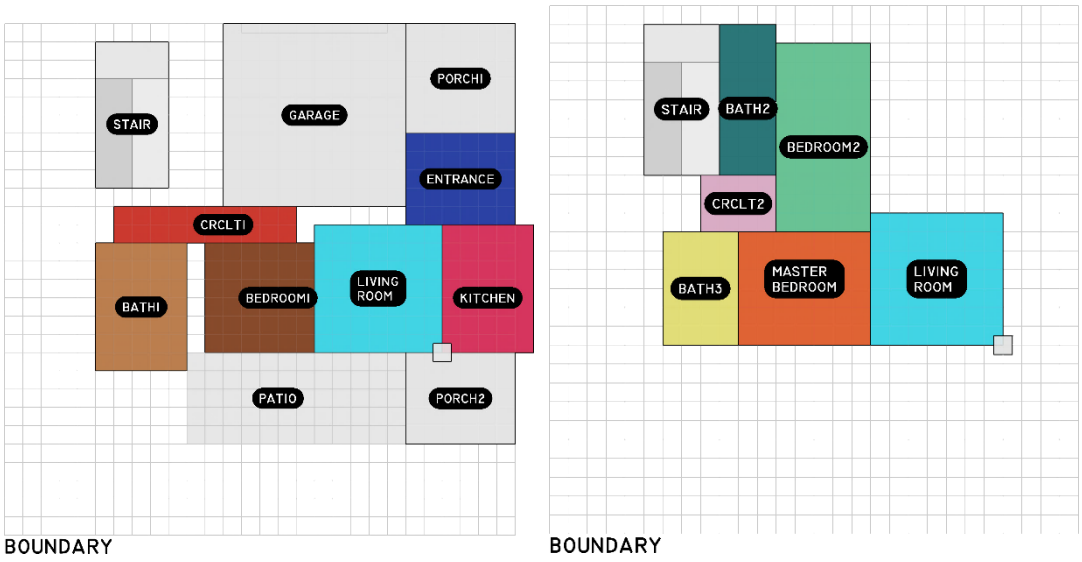


Figure 140. Best layout solution for generation 80 – Top View - Case study 2 – Compactness C (Drawn by the author).

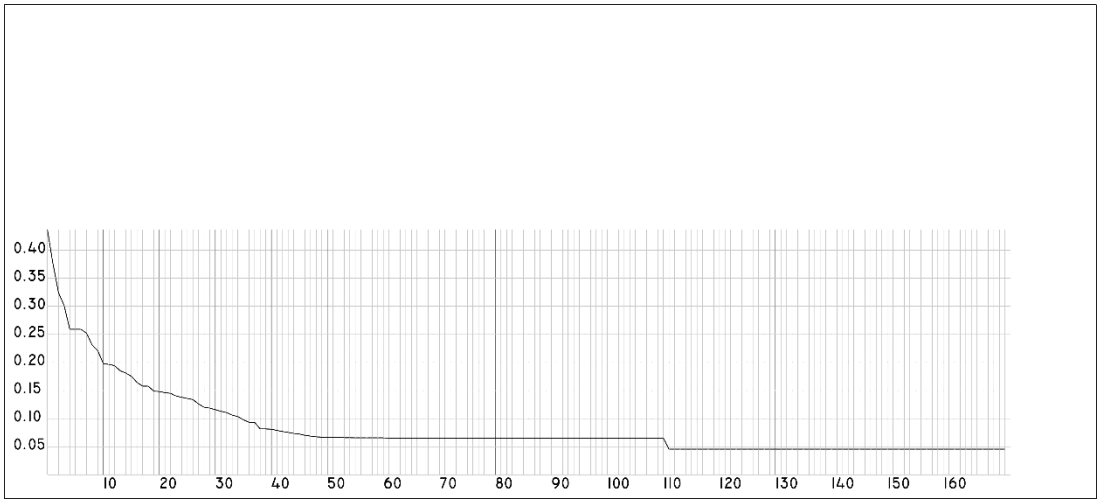


Figure 141. Best total fitness score - Case study 2 – Compactness C(Drawn by the author).

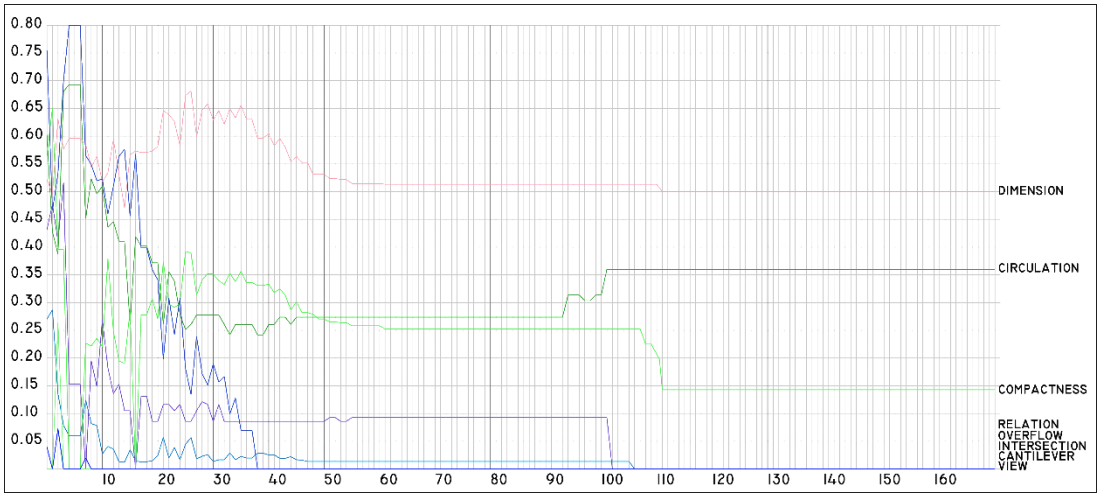


Figure 142. Best fitness score of evaluators - Case study 2 – Compactness C (Drawn by the author).

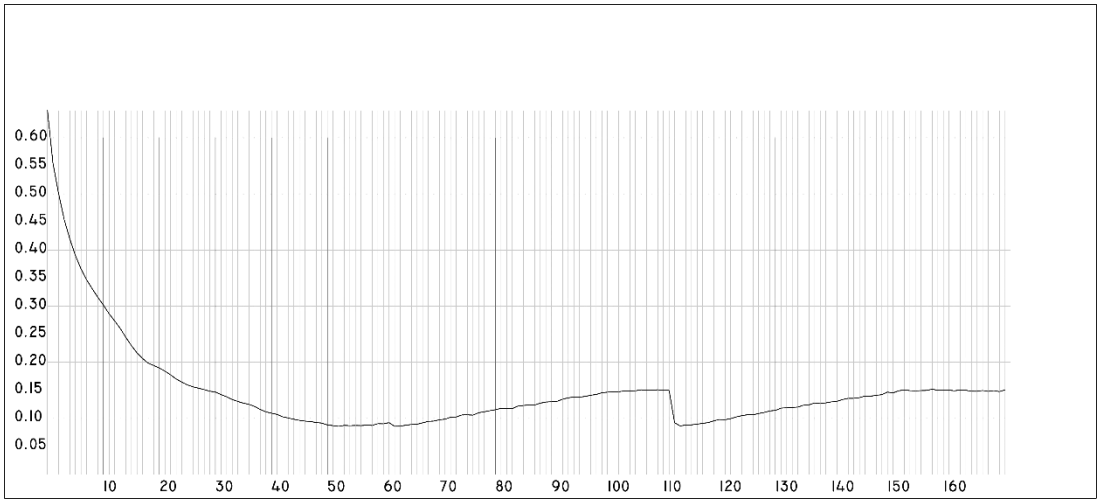


Figure 143. Average total fitness score - Case study 2 – Compactness C (Drawn by the author).

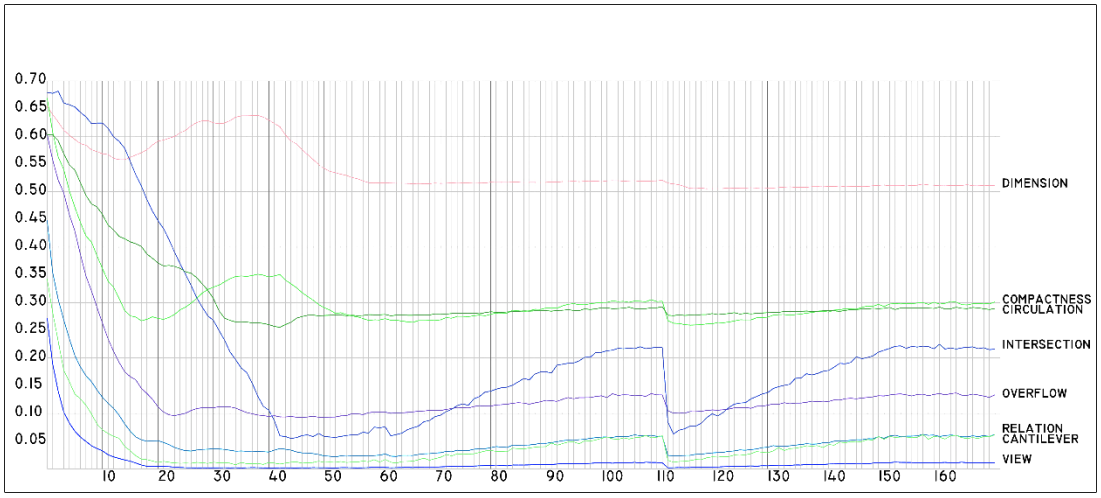


Figure 144. Average fitness score of evaluators - Case study 2 – Compactness C (Drawn by the author).

APPENDIX D

FIGURES FOR CASE STUDY 2 – COMPACTNESS D

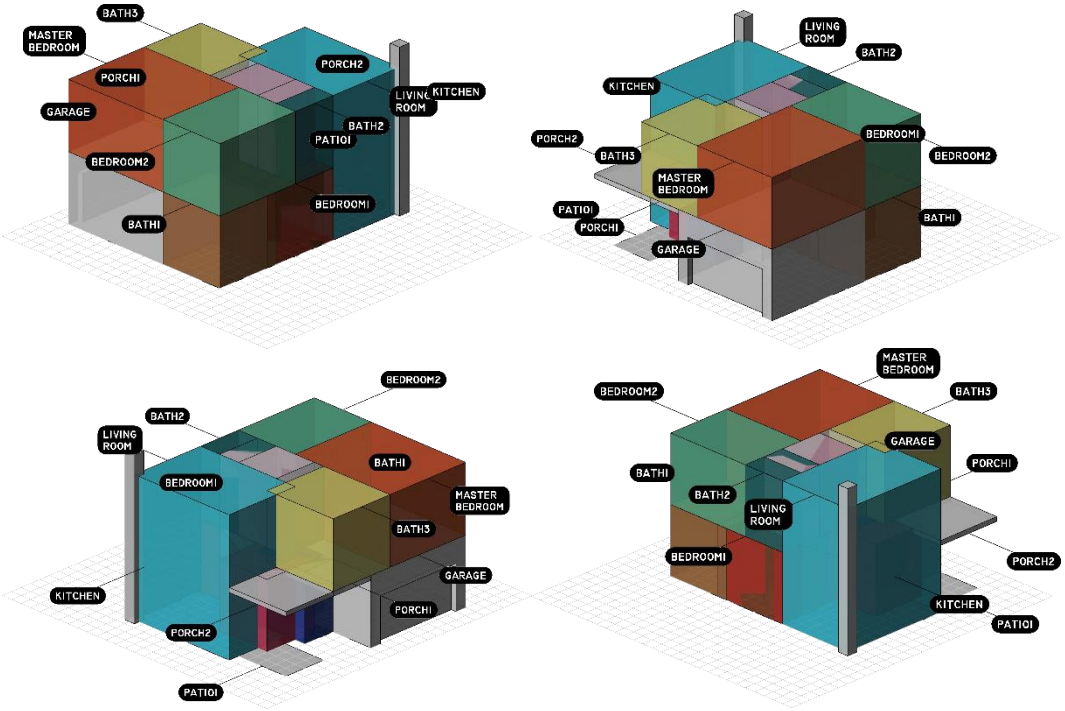


Figure 145. Best layout solution for generation 0 – fitness:0.390, C_{ovf}:0.542, C_{int}:0.418, C_{dim}:0.518, C_{rel}:0.289, C_{comp}:0.663, C_{cant}:0.000, C_{circ}:0.368, C_{view}:0.067 – Parallel projection from four sides - Case study 2 – Compactness D. (Drawn by the author)

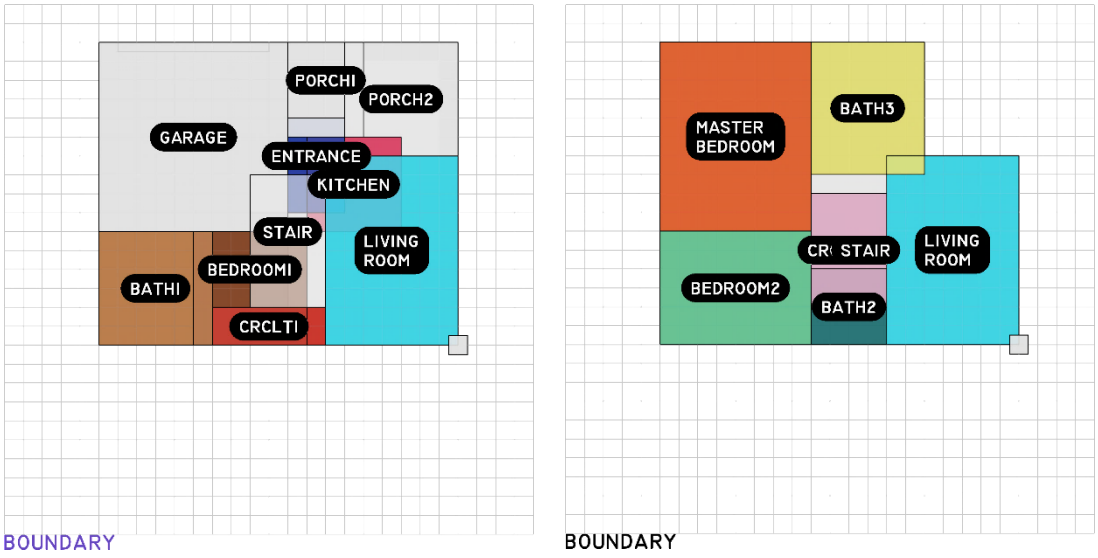


Figure 146. Best layout solution for generation 0 – Top View - Case study 2 – Compactness D. (Drawn by the author)

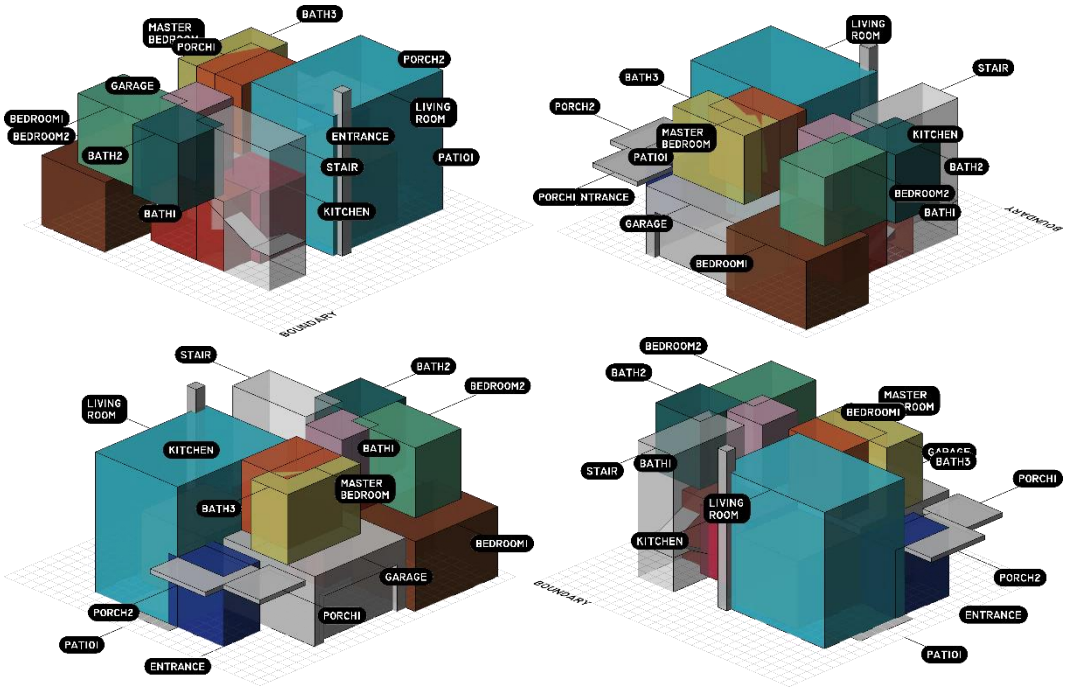


Figure 147. Best layout solution for generation 10 – fitness:0.258, C_{ovf}:0.000, C_{int}:0.525, C_{dim}:0.518, C_{rel}:0.069, C_{comp}:0.585, C_{cant}:0.000, C_{circ}:0.375, C_{view}:0.000 – Parallel projection from four sides - Case study 2 – Compactness D. (Drawn by the author)

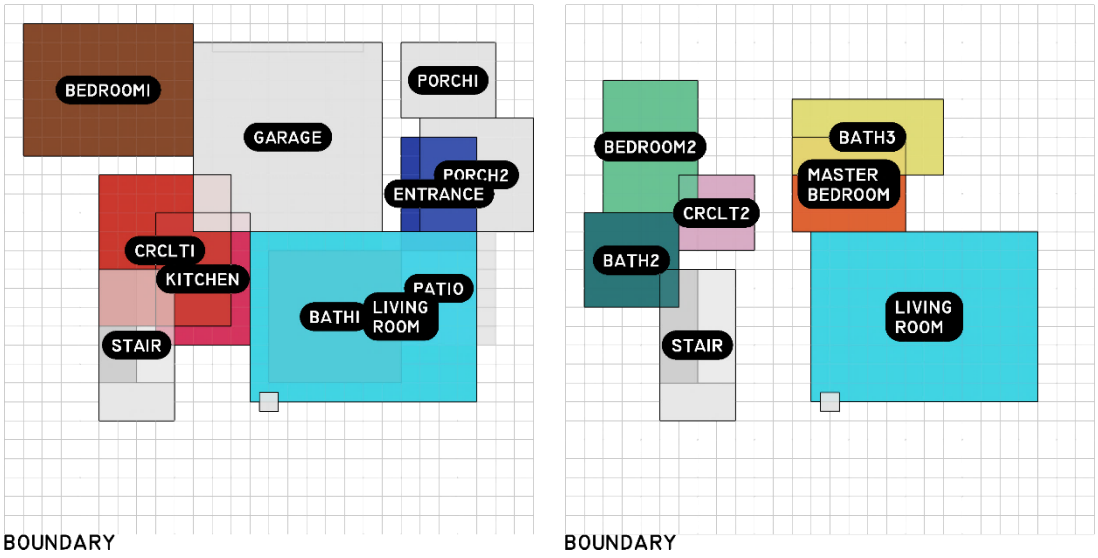


Figure 148. Best layout solution for generation 10 – Top View - Case study 2 – Compactness D. (Drawn by the author)

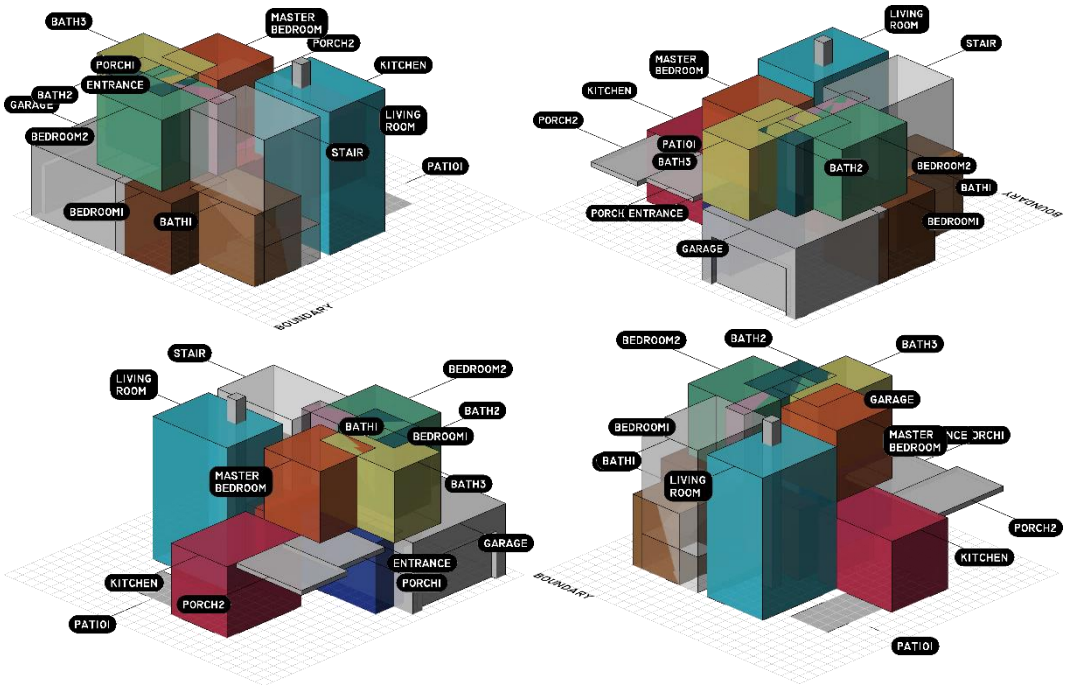


Figure 149. Best layout solution for generation 20 – fitness:0.200, C_{ovf}:0.000, C_{int}:0.381, C_{dim}:0.518, C_{rel}:0.036, C_{comp}:0.611, C_{cant}:0.000, C_{circ}:0.243, C_{view}:0.007 – Parallel projection from four sides - Case study 2 – Compactness D. (Drawn by the author)

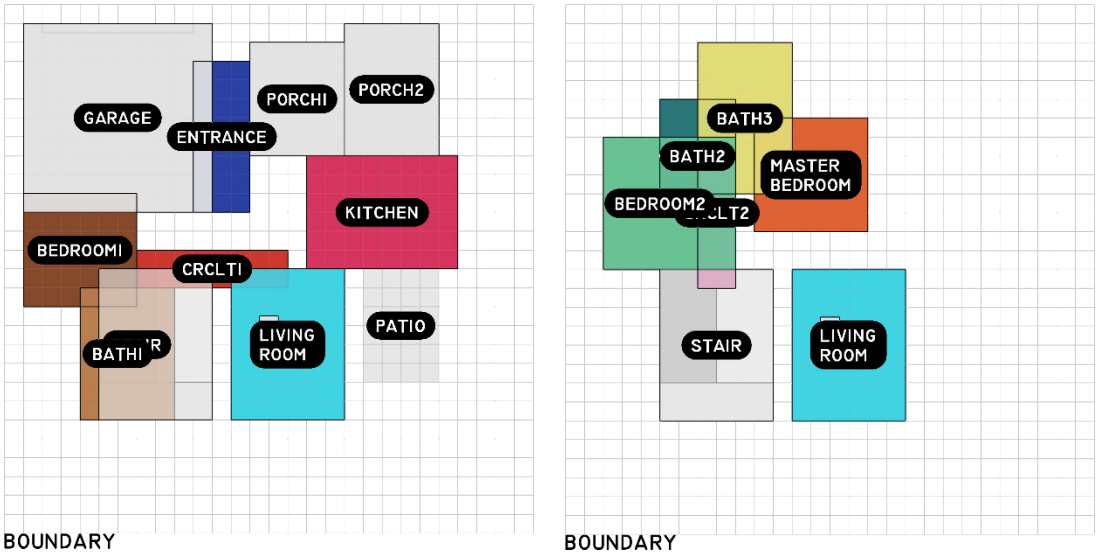


Figure 150. Best layout solution for generation 20 – Top View - Case study 2 – Compactness D. (Drawn by the author)

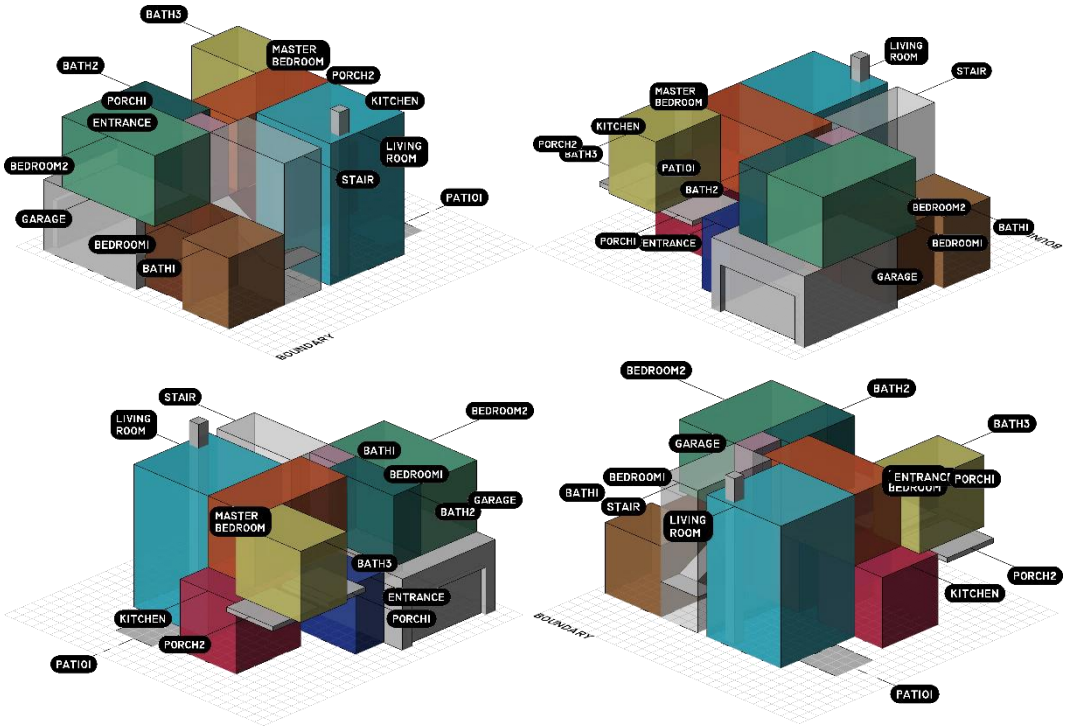


Figure 151. Best layout solution for generation 40 – fitness:0.103, C_{ovf}:0.000, C_{int}:0.000, C_{dim}:0.650, C_{rel}:0.030, C_{comp}:0.376, C_{cant}:0.000, C_{circ}:0.200, C_{view}:0.007 – Parallel projection from four sides - Case study 2 – Compactness D. (Drawn by the author)

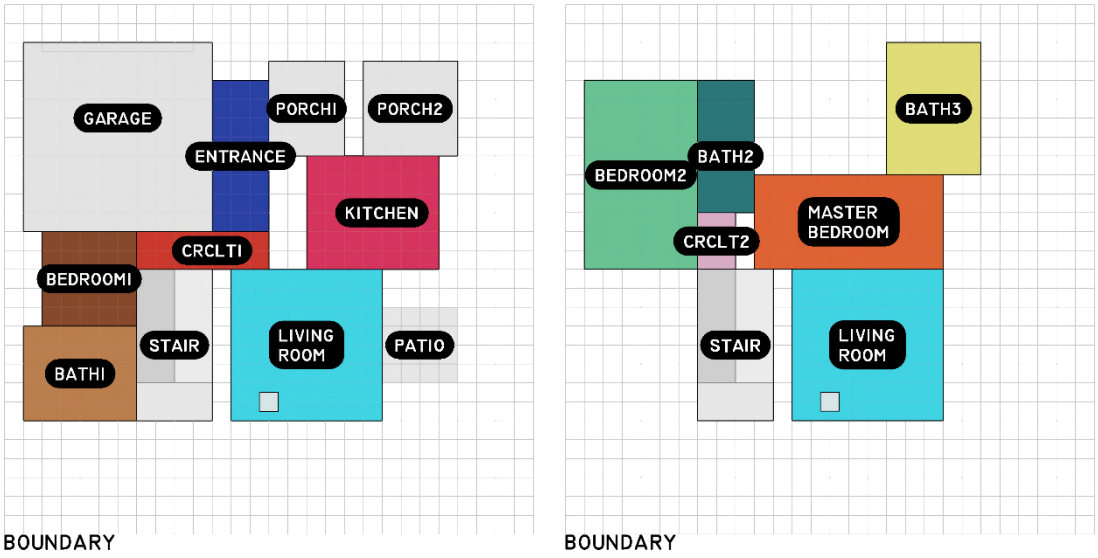


Figure 152. Best layout solution for generation 40 – Top View - Case study 2 – Compactness D. (Drawn by the author)

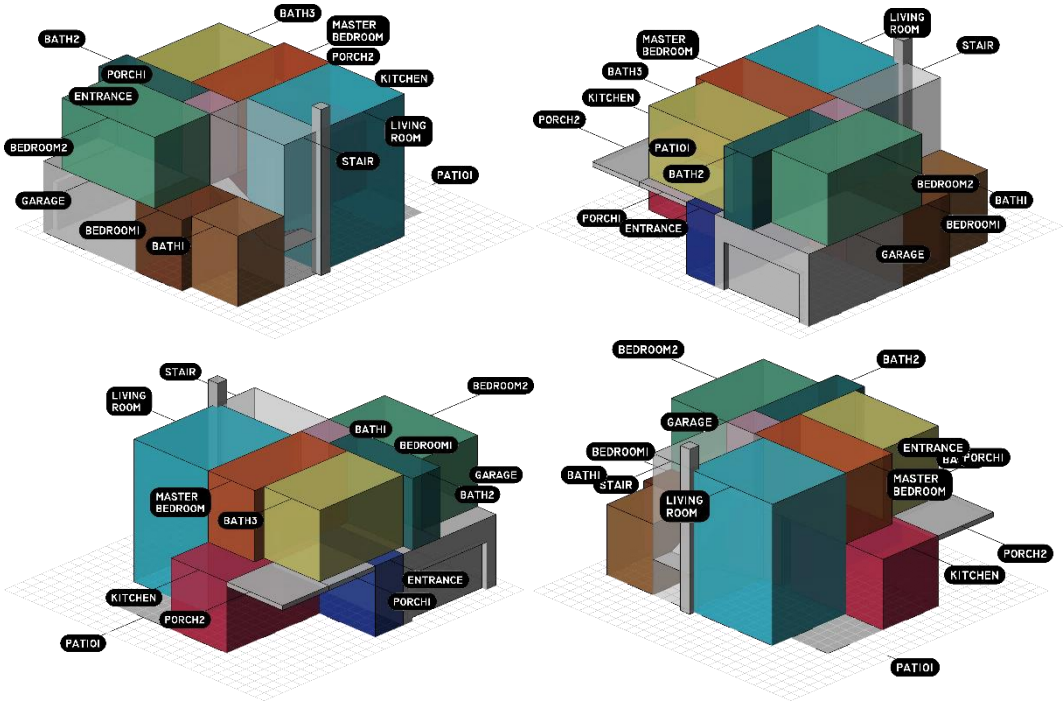


Figure 153. Best layout solution for generation 60 – fitness:0.089, C_{ovf}:0.000, C_{int}:0.000, C_{dim}:0.592, C_{rel}:0.019, C_{comp}:0.315, C_{cant}:0.000, C_{circ}:0.200, C_{view}:0.007 – Parallel projection from four sides - Case study 2 – Compactness D. (Drawn by the author)

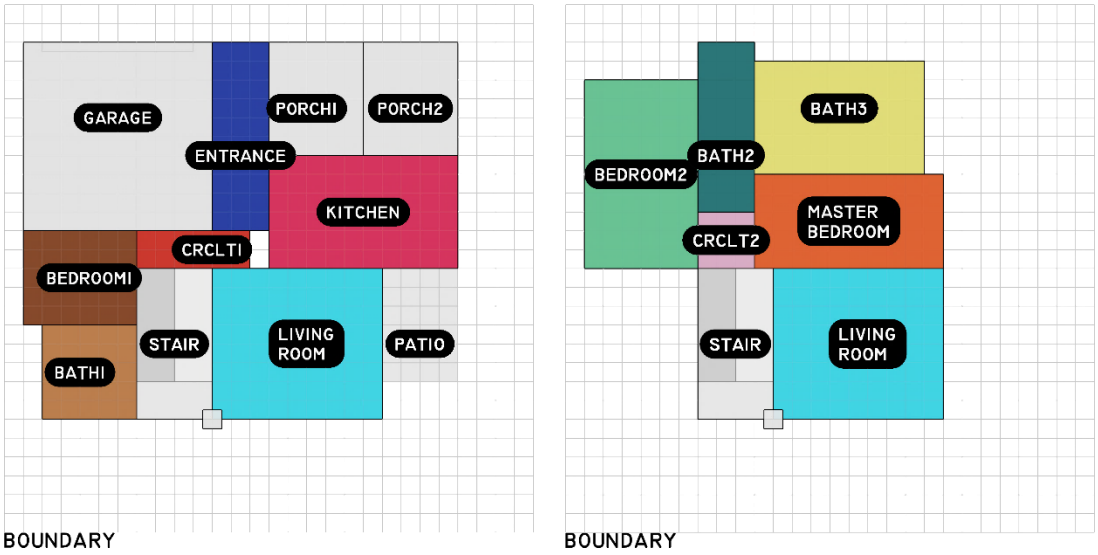


Figure 154. Best layout solution for generation 60 – Top View - Case study 2 – Compactness D. (Drawn by the author)

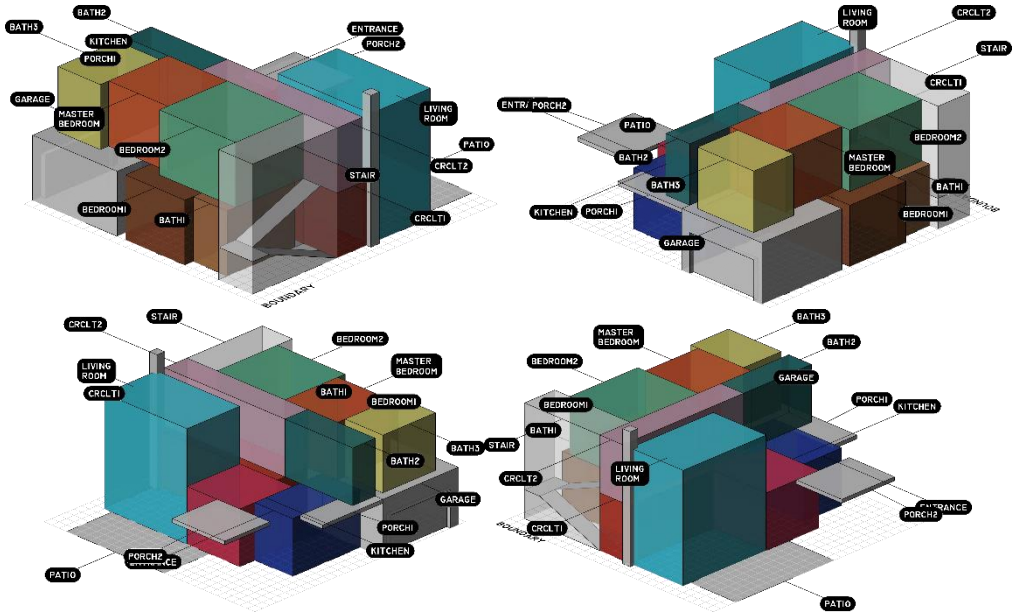


Figure 155. Best layout solution for generation 20 – fitness: 0.103, C_{ovf}:0.000, C_{int}:0.000, C_{dim}:0.523, C_{rel}:0.037, C_{comp}:0.263, C_{cant}:0.000, C_{circ}:0.455, C_{view}:0.000 – Parallel projection from four sides - Case study 2d – Interactive run. (Drawn by the author)

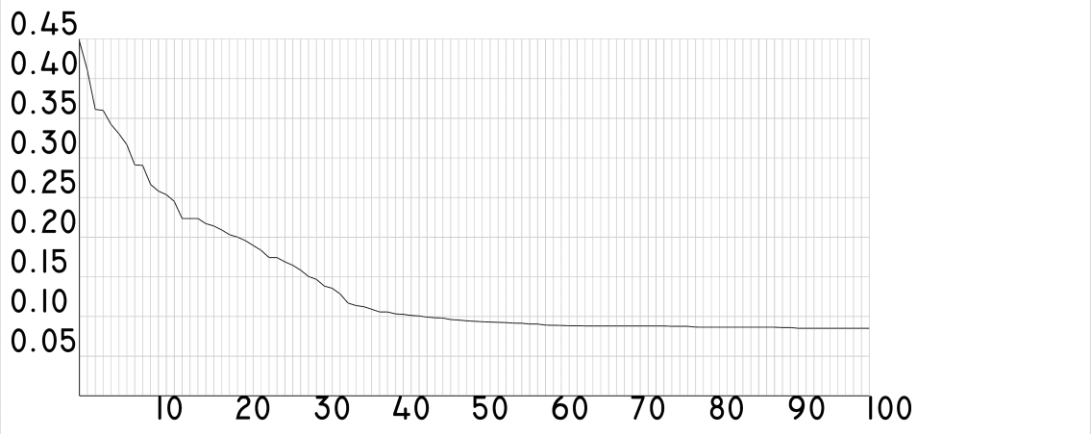


Figure 156. Best total fitness score - Case study 2 – Compactness D (Drawn by the author).

Figure 157. Best evaluator score - Case study 2 – Compactness D (Drawn by the author).

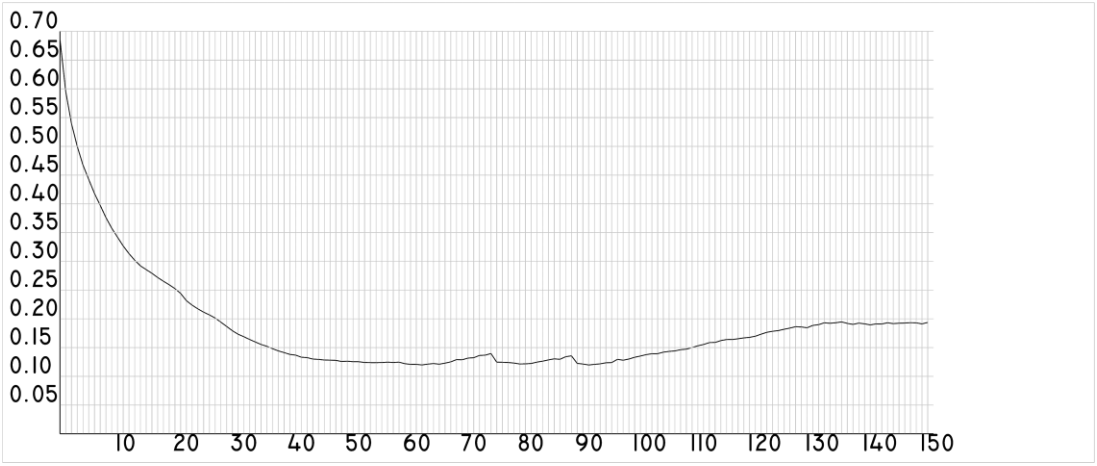


Figure 158. Average total fitness score - Case study 2 – Compactness D (Drawn by the author).

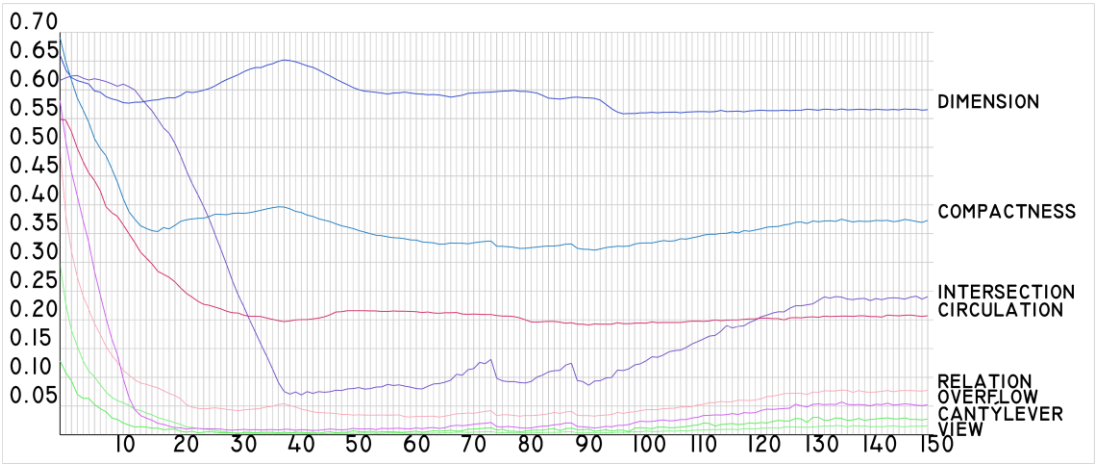


Figure 159. Average evaluator score - Case study 2 – Compactness D (Drawn by the author).

APPENDIX E

FIGURES FOR INTERACTIVE CASE STUDY 2

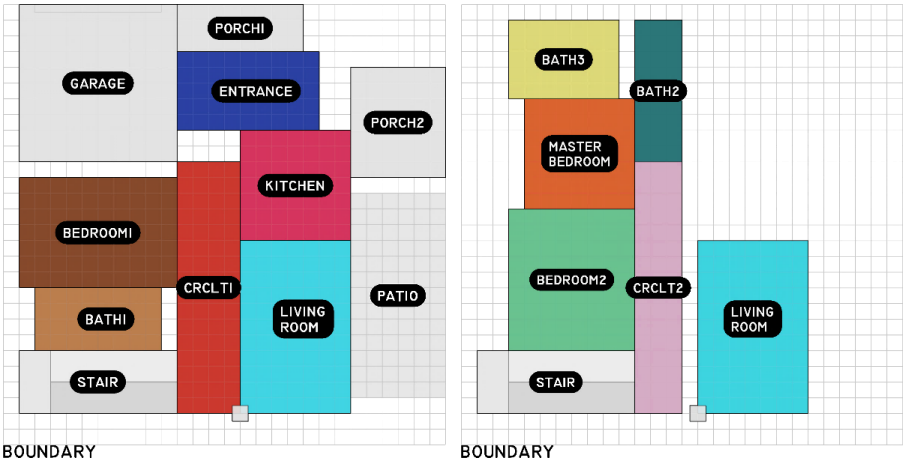


Figure 160. Best layout solution for generation 20 – Top View - Case study 2d – Interactive run. (Drawn by the author)

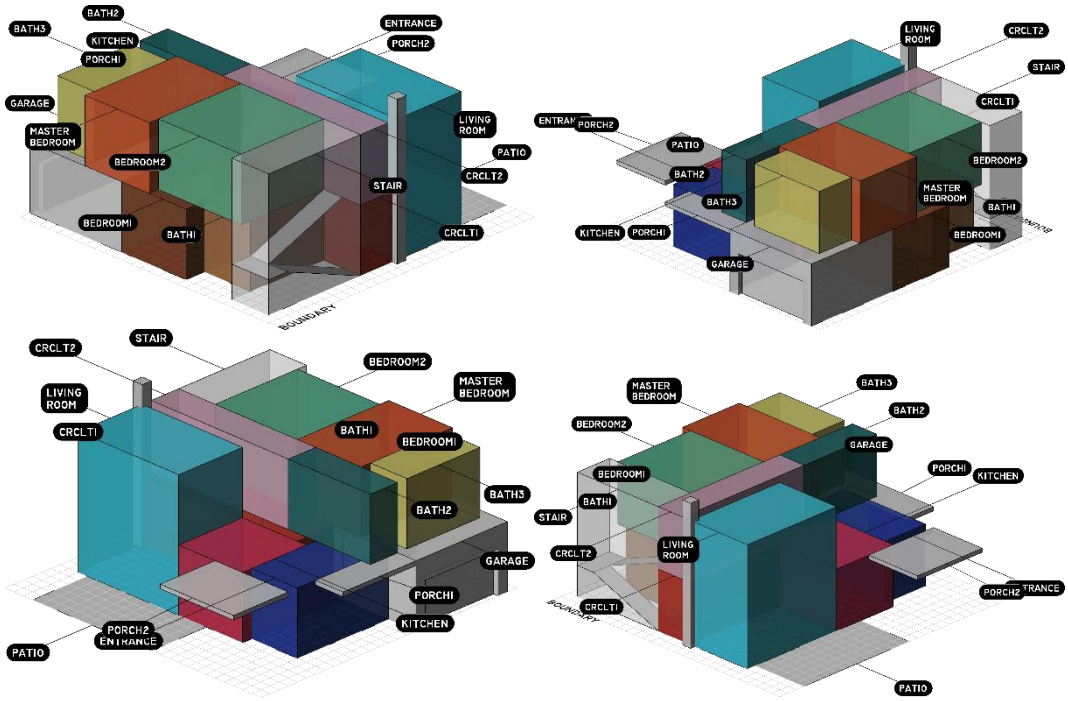


Figure 161. Best layout solution for generation 40 – fitness: 0.089, C_{ovf}:0.000, C_{int}:0.000, C_{dim}:0.502, C_{rel}:0.031, C_{comp}:0.196, C_{cant}:0.000, C_{circ}:0.455, C_{view}:0.000 – Parallel projection from four sides - Case study 2d – Interactive run. (Drawn by the author)

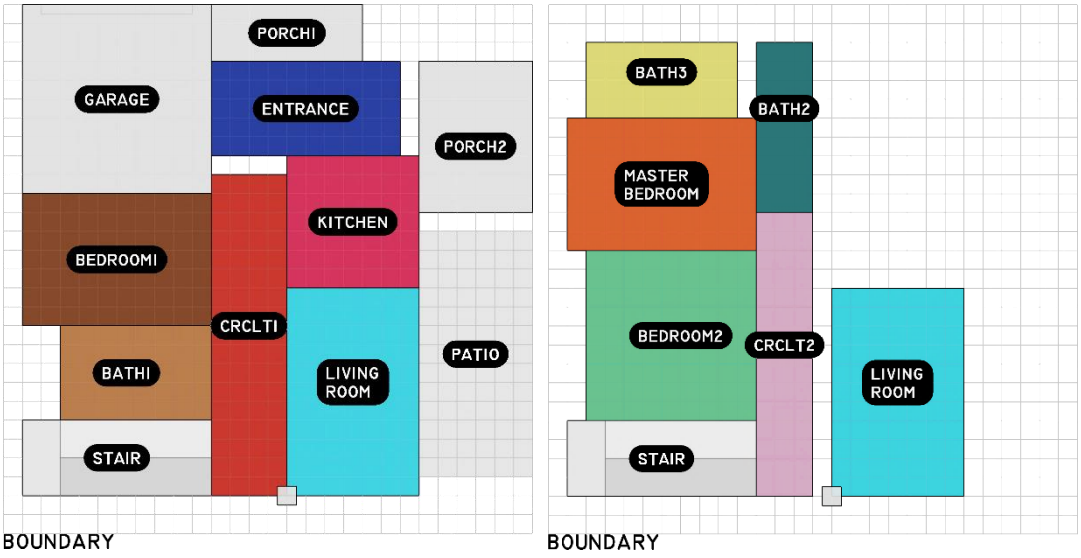


Figure 162. Best layout solution for generation 40 – Top View - Case study 2d – Interactive run. (Drawn by the author)

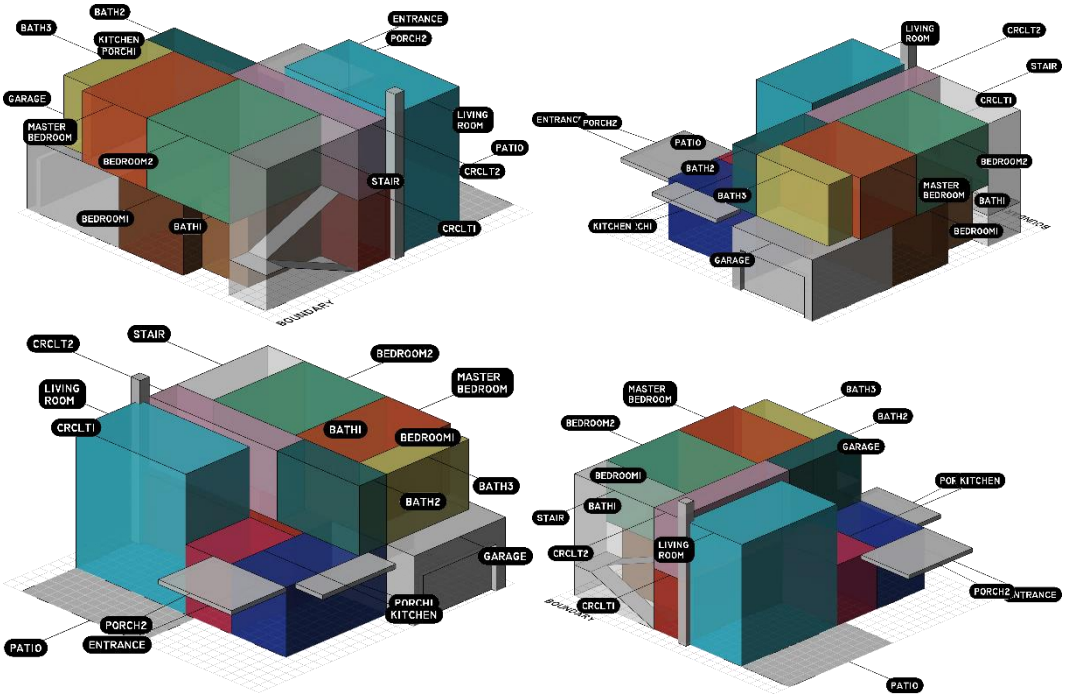


Figure 163. Best layout solution for generation 80 – fitness: 0.083, C_{ovf}:0.000, C_{int}:0.000, C_{dim}:0.552, C_{rel}:0.024, C_{comp}:0.163, C_{cant}:0.000, C_{circ}:0.450, C_{view}:0.000 – Parallel projection from four sides - Case study 2d – Interactive run. (Drawn by the author)

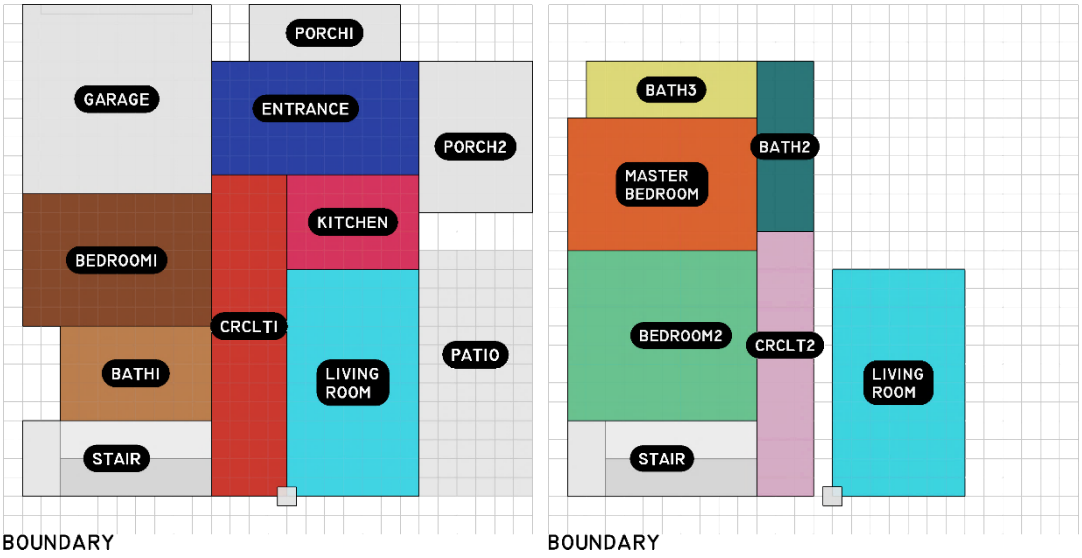


Figure 164. Best layout solution for generation 80 – Top View - Case study 2d – Interactive run. (Drawn by the author)