INTERCONNECTING ACROSS TOOLS AND DESIGN PROFESSIONALS THROUGH A SPATIAL LAYOUT OPTIMIZATION PROCESS

A THESIS SUBMITTED TO
THE FACULTY OF ARCHITECTURE AND ENGINEERING
OF
EPOKA UNIVERSITY

BY

ANISA CENAJ

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

JUNE, 2023

Approval sheet of the Thesis

This is to certify that we have read this thesis entitled **"Interconnecting across tools and design professionals through a spatial layout optimization process"** and that in our opinion it is fully adequate, in scope and quality, as a thesis forthe degree of Master of Science.

_____

Assoc. Prof. Dr. Edmond Manahasa
Head of Department
Date: June, 26, 2023

Examining Committee Members:

Dr. Fabio Naselli	(Architecture)	_____
Dr. Ina Dervishi	(Architecture)	_____
Dr. Anna Yunitsyna	(Architecture)	_____
MSc. Manjola Logli	(Architecture)	_____
MSc. Nerina Baçi	(Architecture)	_____

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 Surname: Anisa Cenaj

Signature: _____

ABSTRACT

INTERCONNECTING ACROSS TOOLS AND DESIGN PROFESSIONALS THROUGH A SPATIAL LAYOUT OPTIMIZATION PROCESS

Cenaj, Anisa

M.Sc., Department of Architecture

Supervisor: Dr. Anna Yunitsyna

Architectural design is in particular interesting for the fact that it involves not only quality of layout-use, esthetics and overall performance and cost but also massively depends on usage of computer capabilities. Computation based approaches in design have been increased in the last decades and rapidly became popular among architects and designers. The programs and their implementation can be beneficial for the design problems which are complex. Computer automation is efficient in terms of both productivity and time consumption. Consequently, it should be taken in consideration as one of the possible and powerful architectural tools of the future.

This study presents an automated computational design process for achieving satisficing spatial layouts for detached houses across pre-defined parameters. The proposed method is based on computation algorithms integrating human-scale inputs to configure adequate spatial configurations using Python programming language. To achieve generative design automation, this research demonstrates a unique algorithm (centrum) prepared from scratch based on the centroid of spaces. The centrum algorithm proposed in this study, is capable of generating several layouts in a short duration of time based on a set of local or user-defined constraints. Additionally, there are integrated a set of criteria to depict the efficient layouts based on goodness value.

As a last step of the workflow, the proposed method incorporates the AutoCAD script modelling to prepare an individual project file which is then imported into the

CAD package. The proposed generative design constantly enables the user to interact with it from an early design stage. Moreover, it illustrates the interconnectivity between different computational tools and techniques for a participatory feedback loop across interacting actors like designers and non-designers. Finally, the entire automation procedure is provided to the user in the form of web-application. This Graphical User Interface (GUI), not only allow the user to interact with each of the automation phases, provide inputs, modify constrains but also gives highest flexibilities in updates as well as usage. Consequently, the user can use any device and operating system to run the application locally or server based.

Keywords: *Architectural Design Optimization, AutoCAD script modelling, Computer Graphics, Detached Houses, Generative Design, Python programming, Spatial allocation*

ABSTRAKT

NDËRLIDHJA PËRMES MJETEVE DHE PROFESIONALËVE TË DIZAJNIT NËPËRMJET PROCESIT TË OPTIMIZIMIT TË PLANIFIKIMIT HAPËSINOR

Cenaj, Anisa

Master Shkencor, Departamenti i Arkitekturës

Udhëheqësi: Dr. Anna Yunitsyna

Dizajni arkitektonik është veçanërisht interesant për faktin se ai përfshin jo vetëm cilësinë e përdorimit, estetikën dhe performancën dhe koston e përgjithshme, por gjithashtu varet masivisht nga përdorimi i aftësive të kompjuterit. Qasjet e bazuara në llogaritje kompjuterike në dizajn janë rritur në dekadat e fundit dhe janë bërë me shpejtësi të njohura në mesin e arkitektëve dhe projektuesve. Programet dhe zbatimi i tyre mund të jenë të dobishëm për problemet e projektimit të cilat janë komplekse. Automatizimi kompjuterik është efikas si për sa i përket produktivitetit ashtu edhe konsumit të kohës. Rrjedhimisht, duhet të merret në konsideratë si një nga mjetet e mundshme dhe të fuqishme arkitekturore të së ardhmes.

Ky studim paraqet një proces të automatizuar të projektimit për arritjen e paraqitjeve të planeve hapësinore për objekt banimi individual përmes parametrave të paracaktuar. Metoda e propozuar bazohet në algoritme llogaritëse që integrojnë inputet në shkallë njerëzore për të konfiguruar planet e duhura hapësinore duke përdorur gjuhën e programimit Python. Për të arritur automatizimin gjenerativ të dizajnit, ky hulumtim demonstron një algoritëm unik “centrum” të përgatitur nga e para bazuar në qendrën e hapësirave. Algoritmi centrum i propozuar në këtë studim, është i aftë të gjenerojë disa planimetri në një kohëzgjatje të shkurtër kohore bazuar në një grup kufizimesh lokale ose të përcaktuara nga përdoruesi. Për më tepër, ka një sërë kriteresh të integruara për të përzgjedhur paraqitjet efikase bazuar në vlerën e “Goodness”.

Si hap i fundit i rrjedhës së punës, metoda e propozuar përfshin modelimin e

skriptit AutoCAD për të përgatitur një skedar projekti individual i cili më pas importohet në paketën CAD. Dizajni gjenerues i propozuar vazhdimisht i mundëson përdoruesit të ndërveprojë me të që në fazën e hershme të projektimit. Për më tepër, ai ilustron ndërlidhjen midis mjeteve dhe teknikave të ndryshme llogaritëse ndërmjet aktorëve ndërveprues si projektuesit dhe jo-dizenjuesit. Së fundi, e gjithë procedura e automatizimit i ofrohet përdoruesit në formën e aplikacionit në internet. Kjo ndërfaqe grafike e përdoruesit, jo vetëm që lejon përdoruesin të ndërveprojë me secilën nga fazat e automatizimit, të sigurojë hyrje, të modifikojë kufizimet, por gjithashtu jep fleksibilitet më të lartë në përditësime si dhe në përdorim. Rrjedhimisht, përdoruesi mund të përdorë çdo pajisje dhe sistem operativ për të ekzekutuar aplikacionin në nivel lokal ose të bazuar në server.

Fjalët kyçe: *Optimizimi i dizajnit arkitektonik, Modelimi i skripteve AutoCAD, Grafika kompjuterike, Objekte banimi indvidual, Dizajni Gjenerativ, Programimi Python, Shpërndarja hapësinore*

To my family

ACKNOWLEDGEMENTS

First and foremost, I would like to begin by expressing my deepest gratitude to Almighty **God** for providing me with health, strength, guidance, and inspiration throughout my academic pursuits.

Next, I am immensely grateful to my **family** and **fiancé** for their unwavering support, encouragement, and understanding. Their love and belief in me have been instrumental in reaching this milestone in my academic journey.

I would like to extend my appreciation to my supervisor, **Dr. Anna Yunitsyna**, for her guidance and continuous support throughout the entire process of conducting my master thesis.

I would also like to acknowledge the esteemed faculty members who have played a significant role in my academic development. I express my gratitude to **Dr. Desantila Hysa** for her valuable insights and contributions, to **Prof. Dr. Hüseyin Bilgin** and to the rector of Epoka University, **Prof. Dr. Ahmet Öztaş**, for their support, and for providing me with the opportunity to conduct further publications based on the findings of my master's thesis.

Lastly, I would like to thank all the participants who generously contributed their time to this research.

TABLE OF CONTENTS

ABSTRACT iii

ABSTRAKT v

ACKNOWLEDGEMENTS viii

TABLE OF CONTENTS ix

LIST OF TABLES xiii

LIST OF FIGURES xv

CHAPTER 1 1

INTRODUTION 1

1.1 Motivation 1

1.2 Problem Statement 2

1.3 Scope and Objectives 3

1.4 Thesis Outline 4

CHAPTER 2 6

LITERATURE REVIEW 6

2.1 Computational Design definition 6

2.2 Generative Design Systems 7

2.2.1 Cellular Automata 7

2.2.2 L-systems 8

2.2.3 Shape Grammar 10

2.2.4 Evolutionary Method 11

2.2.5 Swarm Systems 13

2.2.6 Voronoi Diagrams 14

2.2.7 Agent-based Model 15

2.3 Generative Design Methods 16

2.3.1 Matrix Method 17

2.3.2 Polygon Method 17

2.3.3 Grid Method 18

2.4 Constraints 18
2.4.1 Design Constraints 20
2.4.2 Algorithmic Constraints 24
2.4.3 User-Defined Constraints 25
2.4.4 Local Constraints 26
CHAPTER 3 29
METHODOLOGY 29
3.1 Detached House Definition 29
3.1.1 Grouping of spaces in functional areas 30
3.1.2 Typological categories of floor plans 31
3.1.3 Elements of a residential building 33
3.2 Software Programming Description 34
3.3 Programming structure used for automation procedure 36
3.4 First Phase 39
3.4.1 Space layout assignment 39
3.4.2 Space Orientation 40
3.4.3 Path coordinates calculation 42
3.4.4 Corner prevention 43
3.4.5 Generation of possible room combinations 45
3.4.6 Fixing clashes 46
3.4.6.1 Overlap prevention of two or more spaces 46
3.4.6.2 Filtering Entrance 48
3.4.7 Raw plan layout generation 48
3.5 Second Phase 50
3.5.1 Validation of a good architectural plan layout 50
3.5.2 Perimeter over Area (Compactness) 51
3.5.3 Room Adjacency (Functionality) 52
3.5.4 Parcel Occupancy 54
3.5.5 User interaction 55
3.6 Third Phase 58
3.6.1 Definition of AutoCAD Scripting 58
3.6.2 Preparation of the AutoCAD environment 58

3.6.3 Drawing outer walls 59
3.6.4 Drawing inner walls 61
3.6.5 Drawing windows 63
3.6.6 Naming the rooms 64
3.6.7 Allocating doors and furniture 65
3.6.8 Drawing dimensions 67
3.6.9 Preparation of the template 68
3.6.10 Turning on the layers 69
CHAPTER 4 71
ANALYSIS AND RESULTS 71
4.1 Overview 71
4.2 Analyses for algorithm checks 71
4.2.1 Verifications regarding first phase 71
4.2.1.1 Demonstration of space layout assignment 72
4.2.1.2 Verification of space orientation 72
4.2.1.3 Verification of path coordinate calculation and corner preventions 73
4.2.1.4 Permutation combinations, overlap prevention and raw plan generation 74
4.2.2 Verifications done for second phase 75
4.2.2.1 Verification of general information 75
4.2.2.2 Checks for compactness 76
4.2.2.3 Checks for functionality 77
4.2.2.4 Checks for parcel occupancy 78
4.2.3 Verifications done for third phase 78
4.3 Analyses for architectural layout checks 79
4.2.4 Normalization of fitness parameters 80
4.2.5 Customization of adjacency matrix parameters 84
4.2.6 Calibration of algorithm 86
4.4 Analyses for computational checks (CPU) 87
CHAPTER 5 89
DISCUSSION AND CONCLUSION 89
5.1 Overview 89

5.2 The study innovation 90
5.3 Limitations 92
5.4 Future work 92
REFERENCES 94
APPENDIX 102

LIST OF TABLES

Table 1: Detailed constraints collected from a wide list of publications regarding Geometric design constraints 20

Table 2: Architectural design constraints 22

Table 3: Functional and Physical design constraints 22

Table 4: Efficiency design constraints 23

Table 5: Topology design constraints 24

Table 6: Algorithmic constraints 25

Table 7: User Defined Constraints based on their preferences 25

Table 8: Local constraints of Living Room 26

Table 9: Local constraints of Kitchen 27

Table 10: Local constraints of Bedroom 27

Table 11: Local constraints of Bathroom 28

Table 12: Necessary spaces of a house 30

Table 13: Typologies of floorplans based on the way of the organization referred to Albanian housing design standards [80] 31

Table 14: Standards of the surfaces of the different plan typologies 34

Table 15: Definitions of the Python libraries implemented in the code 35

Table 16: Space orientation Matrix Input 41

Table 17: Adjacency Matrix Input 53

Table 18: Weight / coefficient of fitness values for different scenarios 80

LIST OF FIGURES

Figure 1: The workflow of the Algorithmic Design prepared for this study 4

Figure 2: The schematic view of the computational design method branches involved in architecture design..... 6

Figure 3: Design evolution using an Interactive Cellular Automata [9] 8

Figure 4: Modular L-System [19] 9

Figure 5: Basic Shape Grammars models generated [34] 11

Figure 6: Example floor plan proceeded with Evolutionary Method [40] 12

Figure 7: Tensegrities construction generated based on swarm system as a mimic of nature [47] 13

Figure 8: Generation process of the Voronoi diagram [53] 15

Figure 9: Visualized room occupancy in automated optimization process with Agent-based Model [56]..... 16

Figure 10: The schematic view of constraints and sub-divisions..... 19

Figure 11: Main spaces of a Detached House arranged in relation to the north direction 29

Figure 12: Algorithmic flow chart of the general programming structure used for the automation procedure 38

Figure 13: A conceptual idealization of spatial layout inputs 40

Figure 14: Example of traveling path for one space unit 42

Figure 15: Path coordinate calculations. The concept of coordinates for one space example, and the path coordinates for a space which is forbidden to be oriented in one of compass direction 43

Figure 16: Demonstration of minimum distance ensured for space connectivity on corners 44

Figure 17: The graphical description of the combinations using path coordinates for different spaces, and the respective result of space arrangements 46

Figure 18: Illustration of different cases of possible combinations and overlap evaluation 47

Figure 19: An example for the correct location of rooms in a plan layout referred to entrance (right), and incorrect location of space unit through the entrance (left) 48

Figure 20: Algorithmic flow chart of first phase used in plan layout design automation method 49

Figure 21: Validation of parcel occupancy, not fully occupied parcel (on the left), fully occupied parcel (on the right) 54

Figure 22: The graphical user interface to update fitness values and Adjacency Matric 56

Figure 23: Web Application prepared for the outcome presentations and user interaction 56

Figure 24: Algorithmic flow chart of the second phase used in plan layout design automation method 57

Figure 25: First step of outer wall drawings - reflecting exterior coordinates 60

Figure 26: First step of outer wall drawings - reflecting offset coordinates 60

Figure 27: First step of outer wall drawings - reflecting hatching with a specifying pattern 61

Figure 28: (On the left) example of rejected coordinates provided by python shapely library, (on the right) example of accepted coordinates provided by python shapely library 62

Figure 29: Demonstration of steps followed to draw inner walls: a) drawing the polylines, b) offset of the polylines, c) applying hatch 63

Figure 30: (In the left) representation of calculated coordinates for the arrangement of the window block, (in the right) the insertion of window blocks in the previously window calculations 64

Figure 31: Calculation of center points and allocation of text while naming each of the rooms 65

Figure 32: Sample of a part of the matrix used for the definition of rules and constraints for furniture and doors. 66

Figure 33: Possible orientations of the door for the first corner in one room 67

Figure 34: Illustration of the dimensions for the left side for the plan layout. 68

Figure 35: Demonstration of the arrangement of the template related to the plan drawing 69

Figure 36: Algorithmic flow chart of the third phase used in plan layout design automation method 70

Figure 37: The GUI prepared as web application for user interaction 72

Figure 38: Analyses performed for space orientation based on compass direction 73

Figure 39: Several location of entrance confirming path coordinates 74

Figure 40: Analysis conducted by the algorithm for the overlap prevention 75

Figure 41: (On the left) parameters calculated by the algorithm, (on the right) the parameters calculated in AutoCAD 76

Figure 42: (on the left) a maximum compactness value reached, (in the middle) an average value of compactness reached, (on the right) a low compactness value reached. 77

Figure 43: (on the left) a maximum functionality value reached, (in the middle) an average value of functionality reached, (on the right) a low functionality value reached. 77

Figure 44: (on the left) a maximum parcel occupancy value reached, (in the middle) an average value of parcel occupancy reached, (on the right) a low parcel occupancy value reached. 78

Figure 45: Finalized version of the AutoCAD drawing 79

Figure 46: Comparative assessment for the top six plans based on compactness value 81

Figure 47: Comparative assessment for the top six plans based on functionality value 82

Figure 48: Comparative assessment for the top six plans of parcel occupancy value ... 82

Figure 49: The influence of each fitness value category in compactness, functionality and parcel occupancy 83

Figure 50: The adjacency matrix modification parameters and their influence in the functionality value 85

Figure 51: Analysis for calibration purposes. The original plan (on the left), the twin generated plan (on the right) 86

Figure 52: The improvement in execution time for scenarios with 1 CPU core (on the left) and 6 CPU cores (on the right) from the same device 87

CHAPTER 1

INTRODUTION

1.1 Motivation

Prior to the rise of technology, architects utilized paper and pencils as guides to manually draw plans and develop concept ideas. Over the past twenty years, computing technology has made enormous strides. With new methodologies, software development tools and programming languages, the software industry has also developed significantly, where architecture is not an exclusion.

The broader incentive of the study is making the optimum use of computational tools in architectural design, not just by speeding up the processing of design data, but also by enhancing the designer's intellectual abilities. Since its first inception, the architectural design process has aspired to produce efficient buildings. In addition to the process itself, the design evaluation methods, education, and criteria are subject to change and dependent massively on current research and technology.

The development of technology especially for architectural purposes has not only save the architect's time but also provided various innovative and new design solutions. In the discipline of architecture, there is now a tendency toward the use of automated procedures. Plan layout automation for residential building design is a significant subcategory of computational design. Furthermore, it is known to be as one of many topics that has attracted the attention of researchers in various layout configuration problems as building arrangement in urban design, interior design, furniture arrangement and texture atlases [1].

The common tools for drafting floor plan layouts are mostly known by applying various software such as Computer Aided Design (CAD). The process of planning becomes more open to errors when the designer is faced with varying levels of ambiguity in multidimensional complications. Despite the development of the commercial design software and tools, the professional designers are in need of more sophisticated computational design methods for simulating and generating floor plan

layouts. Therefore this study aims on providing additional computational tool which would ease designers work and lead to more accurate solutions.

1.2 Problem Statement

Designing a floorplan is an imperative aspect of architectural design. Among the primary focuses of automation in floorplan layout is how computers can be incorporated into the process. There has been considerable interest in this field of research for approximately fifty years [2]. Even though it provides very good solutions on layout automations, the Generative Design (GD) methods have been considered with some skepticism from researchers and architects. For instance, the continuous usage of GD by an architect may bring more robotic solutions and somehow limits the inspirations. A culture of passiveness has been attributed to algorithms by architect Peter Eisenman, which is characterized by an immediate response from the computer that numbs critical thinking [3]. In addition to this problem the algorithm prepared for GD method is based on user inputs. As a consequence of this, the outcomes are said to have its own limitations depending on the number of inputs defined. The main advantage of GD method is to automate the generation of spatial layout design which can bring solutions to not only architects but also stockholders. Nevertheless, knowing the complex nature of this procedure which is based in algorithms, it can be used only by trained individuals or architects.

The algorithms prepared in this scientific thesis will mainly target the automation detached housing plans. The importance of GD methods in architectural layouts in general as well as in housing are mainly space and time optimization. According to INSTAT the overall increase of Albanian building sector from year 2021 to 2022 is about 58.2% more [4]. The increased demand in this sector has shown multiple cases of layout replications in different housing blocks. It is strongly supported by researchers that GD is a very good and suitable procedure on avoiding the re-usage of the same plan layout. In addition, it provides a rapid cost estimation as well as real time and large set of layouts which may enrich and satisfy the relationship between the architect and the owner.

The construction sector is always facing several challenges from the initial design idea to the final implementation. This is a phenomena which is often seen in Albania from the early period of communism until recent years. During the communism period, in order to save architectural fees and maintain quality control, they used to design a few layouts and then implement throughout different regions of Albania. The same tradition is followed by some construction sectors nowadays. By considering appropriate GD methods this issue can be fixed for not only new constructions but also retrofitting old architectural design. Moreover, the location of Albania is part of moderate to high seismic region [5]. The Durrës earthquake which happened on November 26, 2019 led to many causalities in both human and materials. It was reported that about 2500 houses were destroyed by the earthquake, leaving about 15000 people homeless [6]. The pilot plan initiated by the government to help the homeless people resulted on a building block of the same building plan. In such circumstances, the implementation of the algorithm can provide various design layouts to adopt a heterogeneous building block. Finally, such procedures can be a bright future and contribution of our society by properly training the upcoming generations involved in architectural education.

1.3 Scope and Objectives

The scope of this research is the compilation of an algorithm that generates readable space layout plans, and distinguishes different plans from each other by applying certain filtering rules defined by user. It focuses on analyzing spatial configuration of residential buildings, specifically on detached houses by implying mathematical equations. The plan layout generation procedure demonstrated in this thesis is proposed uniquely from the other methods observed in literature. The algorithms are written in python programming language. The inputs of this application are based on local design regulation but also flexible to be adopted for other versions. The final output generates several filtered house plan layouts based on user restrictions. These layouts are then analyzed by the architect or the user and can be 2D modeled in CAD software automatically by the application. Hence, additional target of this thesis is to provide a facility of this application in plan conversion to the

commercial software.

The objectives to achieve the abovementioned targets are part of a well prepared flowchart. Initially user has to define the main inputs which will be used also as constraints for the plan generations by the GD method. Using the method implemented in the algorithm which is based on the rule of centroids and controlled by a reference point for each space, first layouts are saved in the application memory. In this way the procedure is of iterative one and aims to combine all possible plan configurations. Once all the combinations are achieved, another objective of this system is to filter the efficient layouts based on previously defined constraints. The selected layouts finally show different design versions and can be used for further decisions.

The architect can select one of the generated layouts based also in the owner requirements. Furthermore, a good objective of this study is to adopt the application layouts to commercial software. The conversion is based on AutoCAD scripting and integrated in the same application package. Finally the application renders the selected graphical layouts and saves all previous solutions. Figure 1 shows the workflow of the generative design algorithm prepared for this study.

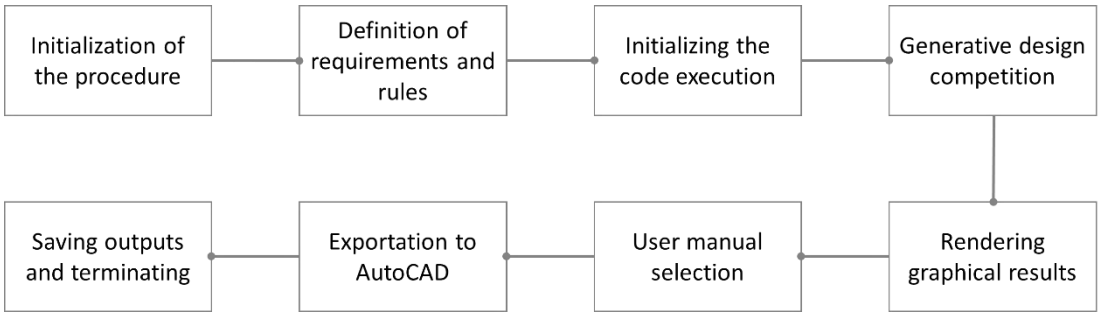


Figure 1: The workflow of the Algorithmic Design prepared for this study

1.4 Thesis Outline

In the first chapter of this study it is introducing the main scope of this thesis on the automation procedure developed in Python programming language for detached houses. The second chapter (literature review), illustrates a wide literature on previous

studies on the same matter. Furthermore, the third chapter gives a detailed method on the implementation of constraints as well as developing each part of the algorithm to achieve the targets of this study. The analysis and results are then demonstrated in the fourth chapter from which the main outcomes are validated based on the previous studies. Finally, the fifth chapter concludes this research study with discussions and final remarks about the findings of the automation process developed for the detached houses.

CHAPTER 2

LITERATURE REVIEW

2.1 Computational Design definition

Computational design deals with automated methods involved in architectural tasks. In literature, there are known three main computational design branches such as Parametric Design, Generative Design and Artificial Intelligence-based methods. In this study, the sub-branches of the Generative Design (GD) procedures are studied. Figure 2 demonstrates the general tree of computational design methods involved especially in architecture. GD approach can be defined as the generation process of several design possibilities based on a set of rules. This approach is based on a system of algorithms which execute multiple procedures which provide multiple design variations [7]. Such systems are dependent on set of rules and parameters pre-defined. The list of rules and parameters will be used to limit the number of outputs to match specific design requirements in a logical and meaningful way.

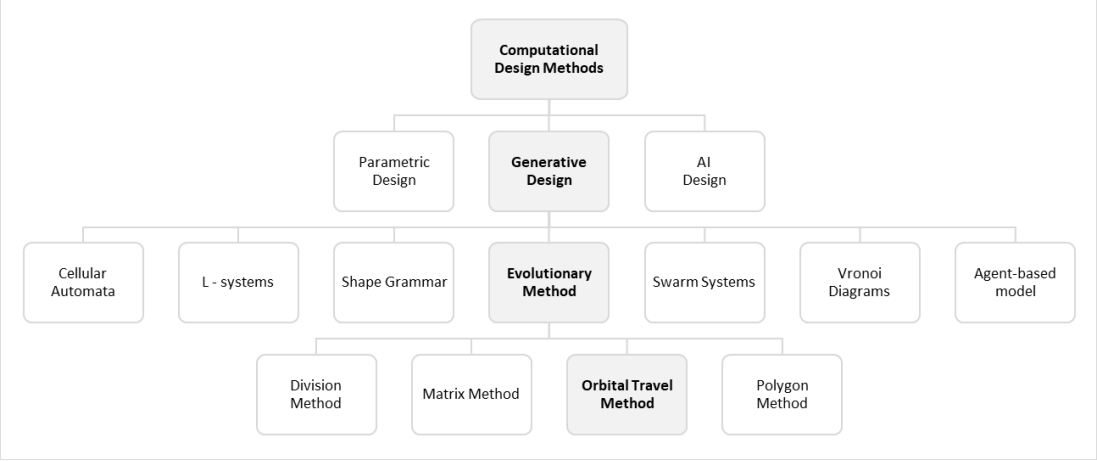


Figure 2: The schematic view of the computational design method branches involved in architecture design

2.2 Generative Design Systems

Among the most well-known GD systems are Cellular Automata, L-systems, Shape Grammars, Evolutionary Method, Swarm Systems, Vronoi Diagrams, Agent-based Model [7]. The target procedure used in this study will be evolutionary method using the unique procedure of Orbital Travel Method which will be called as Centrum method.

2.2.1 Cellular Automata

Cellular Automata (CA) are mathematical models designed to display their knowledge of complex systems. They are all structures of a series of cells, where each cell has a certain state which is determined by some rules [8]. CA consist of three main elements which are Cells, State and Rules. "Unit" cells are the basis of cellular automaton structure. Each cell has a true state that can be varied. States can be binary (such as "alive" or "dead") or multi-valued, such as integers, while rules determine how cells change from one state to another based on their current state and the state of other cells around. These rules can be simple or complex and can affect the development and behavior of the system in different ways. Cellular Automata often build their rules based on natural phenomena, taking inspiration from the development of biological systems. In the context of generative design, cellular automata are used to model and evolve different forms and structures. A cell automaton can represent a design space, where each cell represents a piece of material or finished shape. The use of CA rules can influence the changing state of cells through transformations, interactions, growth and differentiation. Figure 3 illustrates examples from interactive Cellular automata approach. This automation procedure deals not only with the design evolution but also with the tridimensional plan of the design.

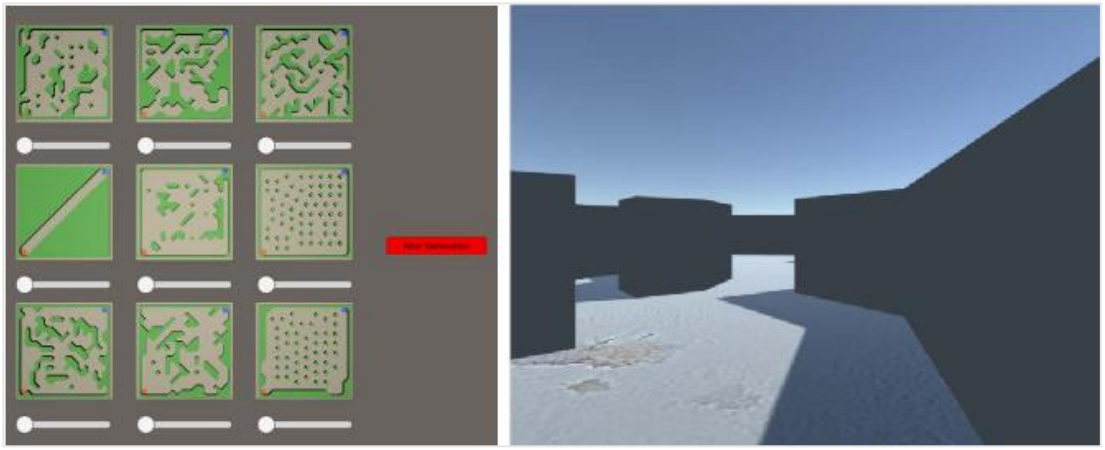


Figure 3: Design evolution using an Interactive Cellular Automata [9]

Urban planning to solve traffic problems is a literature that used Cellular Automata to generate new urban design [10]. In the present research, a cellular automaton was implemented for modeling an urban area's road network to determine potential modifications and enhancements in urban planning. The automaton principles specify how road network cells should change state in response to various conditions such as traffic interactions, and road space utilization. Using the CA, the researchers were able explore with many variations in urban planning, such as adding additional roads, changing the road plan, or increasing the space for public transportation. Simulations using cellular automatons displayed the impact of changes in these components on road traffic and circulation spread in the region being investigated. The study's findings are applied to make suggestions and provide guidance to urban planners to improve road network structure and design. Many other works use the Cellular Automata method to investigate and improve elements of urban design, inspired from natural events and their development processes [11, 12, 13, 14, 15, 16].

2.2.2 L-systems

L-systems, are a mathematical structure similar to cellular automata that is used to model and analyze the relationship between components in a system of design inspired by organisms in nature. This system is used in a variety of design fields,

including architecture [17]. L-systems are made up of three major components: components, links, and dynamic changes. The L-system components can represent design elements such as material, parameters, or other design aspects [18]. Interaction and influence between design elements are represented by links. These links could include rules, constraints, preferences, or any other information that influences the design evolution process. Meanwhile, Dynamics describes the evolution and change of the design system. In Figure 4 is shown an example of modular L-System with minimal inputs. This generates a complex output which potentially can be adaptive.

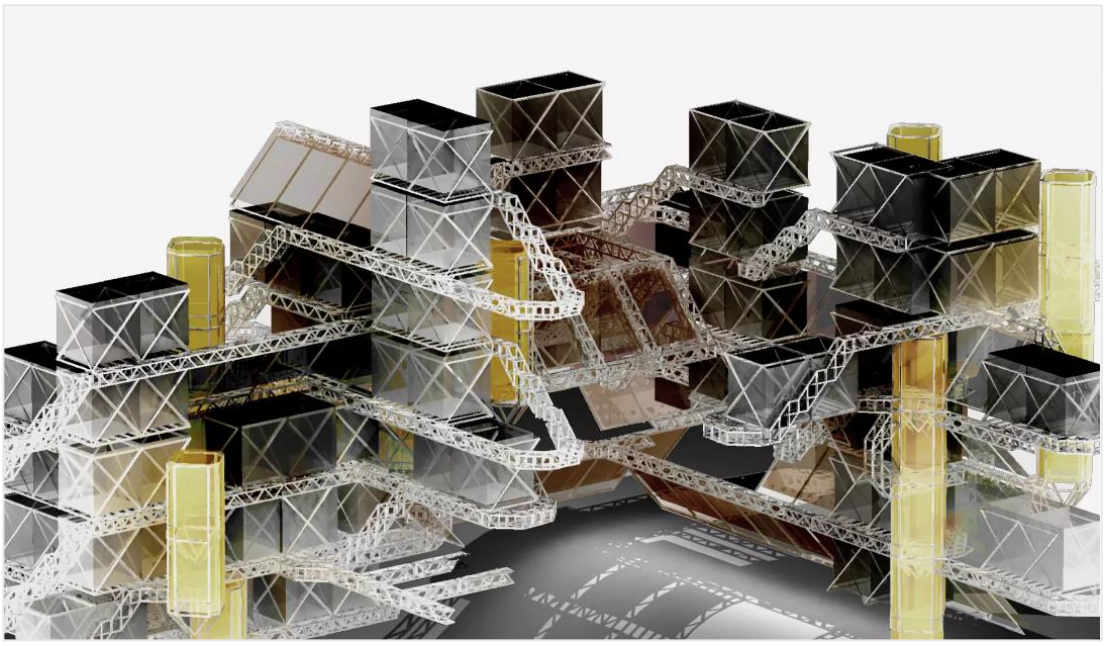


Figure 4: Modular L-System [19]

An L-system was used in the literature to model and develop 3D objects with various organic shapes [20]. The interacting parts of the 3D object were represented by components in the L-system, while links referred to the connections and interactions between them. Applying the rules of this approach in relation to design parameters and criteria enabled the production and exploration of various variations of organic-shaped objects. This consisted of transforming and changing shapes, increasing and decreasing the size of object elements, and changing the texture. The findings of this study will be used to develop guidelines and recommendations for the design and production of 3D objects with organic shapes. The usage of the L-system

in this context opens the way for innovations in the design and development of 3D objects and not only [21, 22, 23], in addition to relying on the complex interaction of the system's elements.

2.2.3 Shape Grammar

Shape grammars in generative design refer to the use of grammatical rules and structures to generate design patterns automatically and freely. This approach utilizes a rule-based and parameter-based system that allows for the iterative and automated production of various shapes [24]. Segments, linking elements, fragments, and units are all basic design elements in a shape grammar. These elements combine to form complex patterns based on grammatical rules and constraints. Grammar rules and parameters like rotation, reflection, and scale increase design flexibility, allowing for quick changes and deep exploration [25]. In practice, generative design can use grammatical forms to develop architectural structures, urban landscapes, industrial products, and much more [26, 27, 28, 29, 30, 31, 32, 33]. Figure 5 represents an example of models generated by shape grammars method by having an initial shape and rules to proceed the automation. In the end this procedure leads to several unit combinations.

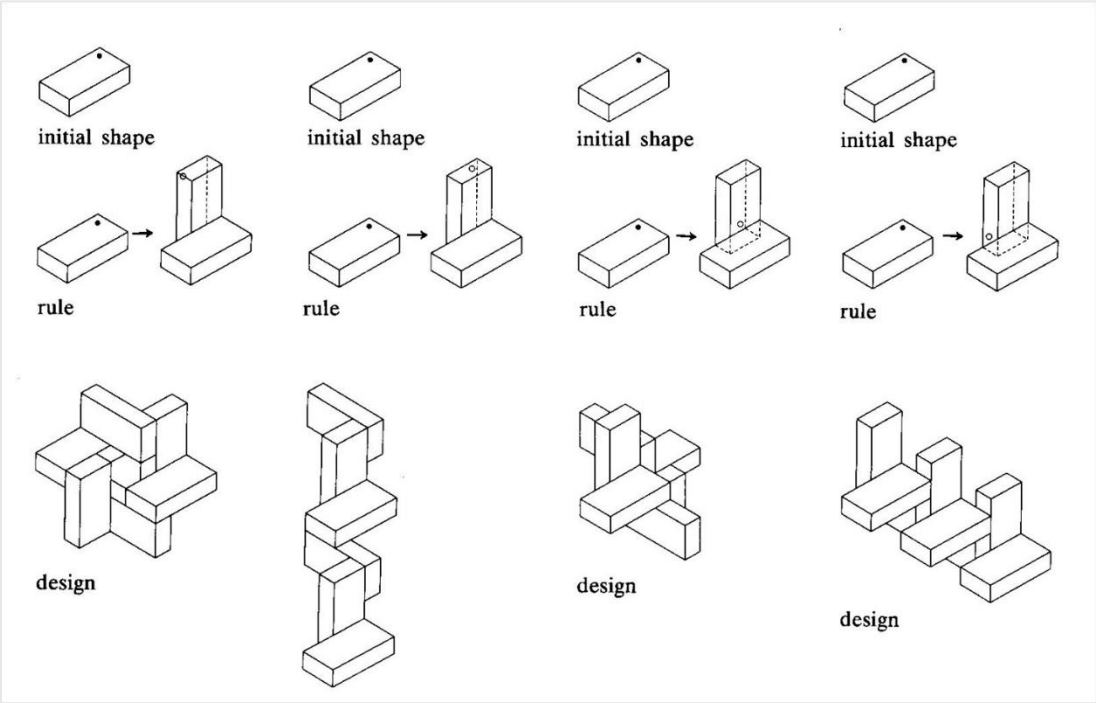


Figure 5: Basic Shape Grammars models generated [34]

Researchers used shape grammar to generate rules applicable at the urban level [30]. This approach was used to specify and describe how different elements of urban design can be combined and organized to form a functional and aesthetically pleasing master plan. In addition, they determined variables such as building size, open space, roads, parks, and shopping areas. Following the identification of these elements, they developed a design grammar which contained rules and parameters for the arrangement and interaction of these components. In addition, these were adapted it to a particular urban area to generate various designs.

2.2.4 Evolutionary Method

The evolutionary method is an approach for developing and producing new shapes and patterns that uses algorithms grounded in concepts from biological evolution theory [35]. While implemented in the framework of generative design, the evolutionary method generates and improves patterns using methods similar to natural selection. This works through integrating design components with design population

evolution processes to generate and select the most effective possible designs and patterns automatically. This approach starts with the development of an initial population of diverse possible patterns or components. The use of evolutionary algorithms including genetics, develops these models through mutation and crossover processes, integrating, adjusting, and identifying the most appropriate and effective models based on specific criteria. The simulated population evolves and adapts to the design requirements gradually. Models with greater quality and more appropriate features are developed and improved, while those with less capable features are removed or transformed. The evolutionary method has been used and researched in a variety of architectural fields [36, 37, 38, 39].

Figure 6: Example floor plan proceeded with Evolutionary Method [40]

A researcher demonstrates the usage of the evolutionary method in the design of a floor plan design as shown also in the Figure 6 [40]. Evolutionary algorithms were used in this study to develop and enhance housing plans based on criteria such as spatial efficiency, functionality, space interaction, and living conditions. This procedure starts with the generation of initial floor plan design using random or rule-based algorithms. The evolutionary algorithm followed up and chose the most suitable layouts based on the defined criteria. The chosen designs serve as "parents" for operations such as genetic crossover and mutation, which produce new and improved designs. This process is repeated and optimized to achieve more effective and appropriate results. The findings of this research are utilized for providing suggestions as well as guidelines for the development of housing plans. The application of evolutionary methods enables the identification of new approaches and the

optimization of living spaces based on predefined criteria. This may include spatial organization, orientation, room and common spaces, and other architectural interactions.

2.2.5 Swarm Systems

Swarm systems are computational models that simulate the behavior of insects or animals in groups by utilizing the concepts of multiple intelligences [41]. Swarm systems (that can represent architectural elements, design elements, or other objects) proceed independently and interact with one another and the surrounding environment in these systems. They adhere to predetermined rules for orientation, departure, and interaction with other members of the group as well as with objects around them. Swarm systems use complex algorithms to change their behavior and position in order to achieve specific targets. In addition they are characterized by self-organization [42]. By applying this system may result in the development of a wide range of designs and models that promote innovation and creativity. One example is shown in the Figure 7 by illustrating how the swarm system produces a tensegrities construction by implementing natural mimics. This approach employs collective intelligence in order to find unexpected and new solutions also to architectural projects and designs. Swarm systems have been used in architecture to produce multiple drawings of building facades, spatial organization, and urban planning [43, 44, 45, 46].

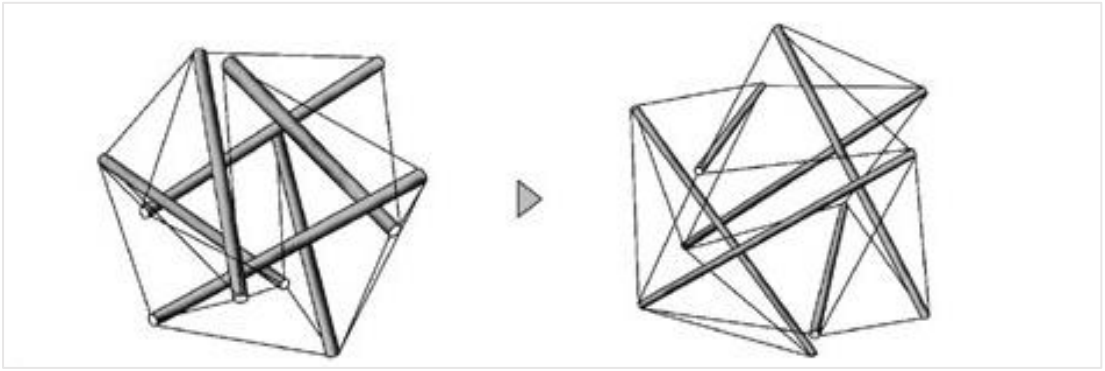


Figure 7: Tensegrities construction generated based on swarm system as a mimic of nature [47]

In literature, Swarm systems have been implemented to improve street planning in a large city [48, 49]. The objective proved to determine the best and most efficient routes to reduce traffic, travel time, and impact on the environment. The procedure starts with a simulation of a group of components representing possible pathways. These components moved through the city space independently, adhering to established rules for orientation, interaction with buildings and street conditions. Components' routes changed and adjusted as the Swarm system iterated and improved in achieving an optimal urban plan. This included changing the trajectory of the roads, interconnection between them, and determining essential locations for the construction of the pathways. This study's findings present an optimized urban circulation map that reduces congestion and travel time for city residents. Furthermore, by identifying preferred routes for fast and efficient transportation, this process was capable of influencing environmental pollution.

2.2.6 Voronoi Diagrams

Voronoi diagrams are tools that are used to generate new structures and patterns according to the division of space into specific areas. Based on the interaction of given points, these diagrams are utilized to arrange and shape space [24]. Additionally they serve to define the interdependence and convenience of the topology of architectural elements or 3D objects. These diagrams are made by dividing space into areas that are closer to and further away from specific points. This results in a round structure with distinct divisions among various parts of the space. The diagram illustrated in Figure 8, is made up of a series of polygonal cells, each of which surrounds a single point and incorporates all points in space that are closer to that seed point than to any other. In addition, Voronoi diagrams can be used to generate complex building facades in the field of architecture. They can help determining the shapes and structures of different parts of the facade based on the distribution of space around them by placing reference points on the facade surface [50, 51, 52]. This can produce creative visual effects and give the impression of geometric processing.

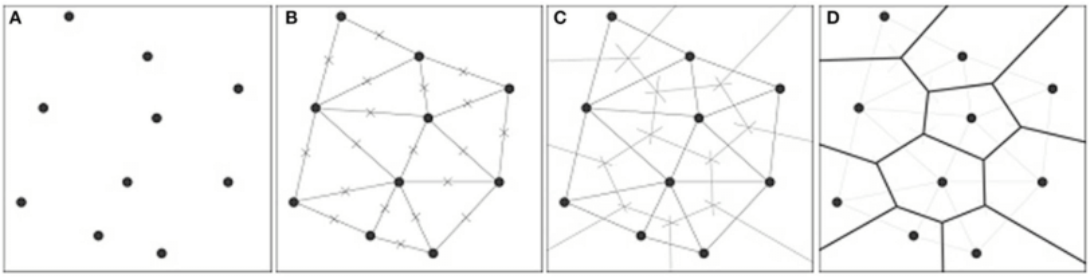


Figure 8: Generation process of the Voronoi diagram [53]

Voronoi diagrams were used by researchers to help in forming the facade of a new building [50]. The aim was to achieve a practical and aesthetically pleasing facade by separation of space into areas which affect the form and arrangement of the facade components. The procedure started with the positioning of reference points on the facade's surface. These points become the basis for Voronoi diagrams. These were generated by dividing the facade space into different areas based on the proximity of the reference points using the necessary algorithms. Following the development of Voronoi diagrams, space division can be used to produce details and various elements on the facade such as exterior panels, windows, and other design components. This can give the building facade a structured and geometrically elaborate design. The findings of this study ensured that the space was organized in order to create a unique and special form for the facade, giving it a distinct identity.

2.2.7 Agent-based Model

In generative design, agent-based models are utilized to generate new models based on the activity and interaction of defined agents [54, 55]. These models depict complex systems by employing individualized agents that act and interact in a specific space. Agent-based models are used to describe and simulate the interaction processes of buildings, equipment, or other elements in an architectural environment. Agents are self-contained, intelligent entities with their own set of properties and behaviors. Agent-based models can be applied to the design of new cities to simulate resident behavior and influence space organization. Different agents may have different preferences, such as residential location, service accessibility, or cultural preferences.

Agent models may assist in the identification of potential locations for buildings, city infrastructure, and other services in order to optimize them for the needs and preferences of residents. In Figure 9 is illustrated an example of automated optimization by using agent rules for the generation of building form and room occupancy.

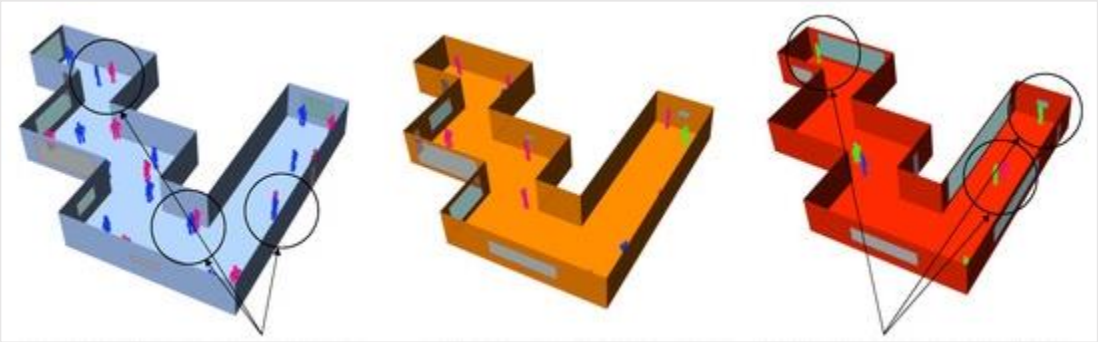


Figure 9: Visualized room occupancy in automated optimization process with Agent-based Model [56]

The Agent-based model approach has been implemented in several designs, including those of [57, 58]. The researchers use their method to design a virtual city, through a set of rules to control the behavior of virtual agents. These rules define how the agents should interact with one another and with the environment, including how they move and interact with objects in the environment. To guide the design process, the researchers also incorporate feedback from human designers. The resulting virtual city is a complex and dynamic system that responds to the behavior of virtual agents as well as changing environmental conditions. The authors show how their method can generate a variety of design solutions, each with a unique set of constraints.

2.3 Generative Design Methods

Generative design is a paradigm that automates the production and assessment of design alternatives based on predefined goals and criteria by combining the constructive qualities of mathematical and visual optimization approaches. It is

capable of performing a variety of tasks, including conceptual design, parametric design, topology and shape optimization. Generative design may be used to solve layout challenges utilizing various approaches such as matrix, polygon, and grid. Depending on the nature and complexity of the problem, each approach has advantages and drawbacks.

2.3.1 Matrix Method

The matrix method is a widely used approach for generative design of plan layouts. It involves representing the design space as a matrix, where each cell corresponds to a specific area or element. Various algorithms, such as genetic algorithms and simulated annealing, can be applied to iteratively optimize the layout based on defined constraints and objectives [59]. The matrix method offers flexibility in defining design rules and allows for easy integration of spatial relationships between different elements. It has been successfully applied in architectural design, facility layout planning, and urban design.

2.3.2 Polygon Method

The polygon method, also known as the Voronoi diagram method, is another popular technique for generative design of plan layouts. It is based on the concept of dividing the design space into polygons that represent different zones or regions. The layout optimization process involves adjusting the boundaries of these polygons to achieve desired objectives, such as maximizing connectivity or minimizing distances between elements [60]. The polygon method offers advantages in generating organic and irregular layouts, making it suitable for landscape design, urban planning, and terrain modeling. However, it can be computationally expensive and sensitive to the initial configuration of polygons.

2.3.3 Grid Method

The grid method involves dividing the design space into a grid of cells and assigning different attributes or functions to each cell. The optimization process aims to determine the most suitable allocation of functions within the grid, considering constraints and objectives. The grid method provides a structured and modular approach to plan layout generation, making it suitable for applications such as building floor plans, transportation networks, and utility distribution. It allows for efficient data management and simplifies the implementation of automation and parametric design techniques [61, 62, 63]. However, the grid method may lack the flexibility to capture complex spatial relationships and can result in regular and repetitive layouts if not carefully designed.

2.4 Constraints

The set of constraints defined for the Centrum method in this study are collected from a wide list of publications, books and guidelines. These constraints will be integrated into the algorithm based on the Graphical User Interface. In the field of plan layout generation, there are several constraints that must be considered to generate an efficient and functional design. One of the major struggle in plan layout generation is that some of these constraints may be contradictory to each other, and some constraints may appear to be repetitive, even though they are classified differently [64, 65, 62, 66, 67, 68, 69]. Balancing these conflicting and repetitive constraints is a key challenge in plan layout generation, and requires a systematic approach to ensure that the final design meets the desired criteria. To achieve the target of Generative Design for Detached Houses, constraints are subcategorized into Design, Algorithmic, Local and User-defined parameters as shown in Figure 10.

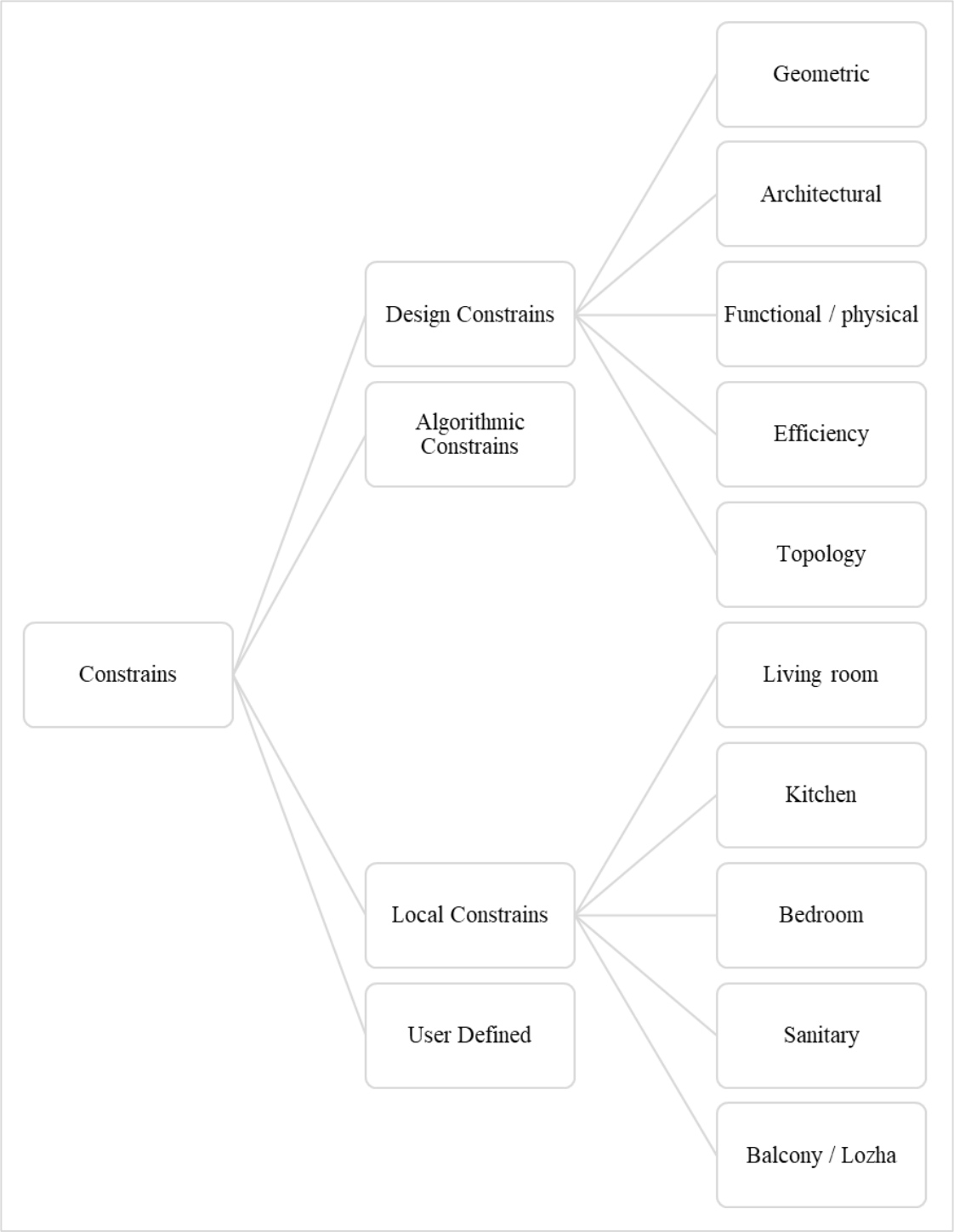


Figure 10: The schematic view of constraints and sub-divisions

Based on the schematic view shown in the Figure 10, a detailed list of these constraints for each of the sub-branch are listed and descripted below.

2.4.1 Design Constraints

Design constraints play a crucial role in the generation of plan layouts in architecture and interior design. These constraints include factors such as geometric, architectural, efficiency, topology and functional requirements of the space. Adhering to these constraints is essential in order to ensure that the final design is safe, feasible, and functional. However, balancing these constraints with the aesthetic goals of the design can be a challenge. In literature, [63, 70, 62, 71, 64, 72, 73, 67, 68, 69] researchers have proposed various techniques and algorithms to optimize the plan layout generation process, taking into consideration several design constraints. By incorporating these techniques, designers can generate a significant number of plan layouts that can be accepted as functional proposals. In Table 1 is defined a wide list of constraints collected from a wide list of publications and categorized as geometric constraints [63, 70, 62, 71].

Table 1: Detailed constraints collected from a wide list of publications regarding Geometric design constraints

No.	Design Constraints - Geometric
1	Requiring Units to be placed within the main building boundary or other designated group boundaries.
2	Prohibition of the intersection of two Units from occupying the same space.
3	Requiring a minimum amount of overlap between two or more units to prevent overlap and establish a clear relationship between them.
4	Forcing units to align with or be placed near the edge to ensure that are properly positioned relative to other elements in the design.
5	Bounding the area of a Unit with minimum area, maximum length/width and minimum length/width.
6	Preserving a specific aesthetic appearance or avoiding elongated, narrow rooms that may not be functional.
7	Limiting construction costs to a specific budget, i.e. in terms of material cost.

8	Ensuring a minimum level of natural lighting for designated rooms.
9	Ensuring that rooms are positioned in a way that promotes easy access and circulation.
10	Positioning of the main rooms in order to meet the requirement for natural lighting.
11	Positioning auxiliary rooms in order to meet the design requirement.
12	Gaps are projected between the first and last rooms to ensure that the main entrance may be realized.
13	The building's size is restricted and determined by the user or designer.
14	For simplicity in programming, rooms are limited to rectangular shapes.
15	The dimensions of the Rooms must be provided as input by the client or designer.
16	The dimensions of all apartments are predetermined and provided as input.
17	As an input, any distance limitation between two rooms may be provided.
18	Corridors are not prescribed but should be as short as feasible.

Among design constraints there are many categorization listed from the different researchers. Architectural constraints are one of them used to guide in regards to essential architectural principles as the room size, aesthetical style and structural elements. These constraints are collected from different literatures in Table 2 [71, 64, 72, 73, 67, 68, 69]

Table 2: Architectural design constraints

No.	Design Constraints - Architectural
1	The Room's size should be approximately equal to the specified area.
2	Ensuring that the room is not too large or too small, or maintaining a specific aesthetic style.
3	The Rooms should be positioned in order to have common structural elements.

Functional constraints are utilized to contribute to the plan layout regarding functional and physical aspect of the building plan as it includes the room adjacency [71] which is one of the crucial principles that a professional designer has to take into consideration while designing a residential layout. Some other factors that are revised from the researchers are cost budget and natural light. All these constraints are summarized in Table 3 from different studies [64, 74, 63, 73, 67, 69, 75] .

Table 3: Functional and Physical design constraints

No.	Design Constraints - Functional / physical
1	Ensuring that a minimum amount of glass surface area is included in the design to provide natural lighting.
2	Communal spaces within a building design compared to the total number of rooms.
3	The living area, kitchen, and dining room should all work as a single functional unit.
4	Private Rooms should be positioned in order to not be isolated from natural light.
5	Two toilets should not be adjacent to each other in layout composition.
6	It is recommended that the living room have access to exterior spaces such as a garden, patio, or huge balcony.
7	Keeping the design costs within a minimum budget limit for the user.

8	The units must be located within the main building boundary as initially defined.
9	Preventing two units from intersecting and occupying the same space.
10	The doorways must be included in the design layout at to prevent design errors.
11	The windows must be ensured to ensure natural light in almost every Unit.
12	The area of Units must be minimal as an initial input from the user.
13	The user must provide an initial input specifying a minimum ratio of units.
14	The window width of a Unit cannot be larger than the wall width.
15	Certain rooms must have a minimum amount of natural lighting.

Design constraints are massively used by various researchers in order to increase the efficiency of the generated plan layouts. For instance, in Table 4 it is shown an example of maintaining sufficient distance between two or more spaces which is directly influenced by the area of these two areas involved [71].

Table 4: Efficiency design constraints

No.	**Design Constraints - Efficiency**
1	Two of the solutions at least would be identical if the specifications of the Units are the same type and size.
2	Depending on the surface area of the room, a maximum distance between any two cells in the same room is defined.

Additionally, topology constraints summarized in Table 5 plays a crucial role to the entire stage of the design due to foundation rules and logical instructions that have to be considered during the process. These requirements consist of overlapping between two or more spaces and room connectivity [64, 63, 70].

Table 5: Topology design constraints

No.	Design Constraints - Topology
1	Overlap Constraint guarantees that two units do not occupy the same area.
2	Connectivity constraints, for instance how a certain room must be assigned.
3	Path constraints may be necessary among all room combinations.
4	Constraints on planarity ensure that the geometry can be achieved using a two-dimensional floorplan.
5	Assuring that units are compelled to be linked to an exterior wall.

2.4.2 Algorithmic Constraints

Algorithmic constraints define the rules and conditions that guide the layout generation process, ensuring that the ultimate design is near an optimal spatial plan composition. The selection of appropriate algorithmic constraints is crucial to the success of the layout generation process, as it helps to ensure that the final design meets the necessary requirements and constraints. Different researchers have proposed various algorithms for plan layout generation as also shown in Table 6, taking into consideration various algorithmic constraints [66]. These algorithms range from rule-based systems to machine learning approaches, each with its own set of strengths and limitations. Designers can generate plan layouts that are not only functional and feasible, but also visually appealing and efficient by incorporating appropriate algorithmic constraints. The formulation of new and more improved algorithms for plan layout generation remains an active area of study in the field.

Table 6: Algorithmic constraints

No.	Algorithmic Constraints
1	Constraint graph format to model layout restrictions. It defines the interactions between elements in a layout, such as relative sizes, positions, and alignments.
2	Use of efficient algorithms to identify practical solutions by formulating layout generation as a constraint satisfaction issue and
3	Shape grammar rules specify the decomposition and the transformation of layout elements depending on the constraints stated.

2.4.3 User-Defined Constraints

These constraints are defined by the user and are customized to their specific design needs. They may include, among other things, total built area, room number, size, and location. By incorporating user-defined constraints into the layout generation process, designers can ensure that the final design meets the specific needs and preferences of the client. Various strategies have been presented for integrating user-defined constraints into the plan layout generating process, as shown in Table 7 [68, 60]. These techniques span from interactive design tools to machine learning algorithms that can learn from user input. The incorporation of user-defined constraints in plan layout generation helps enhance the design process by allowing it to be more personalized and adapted to the client's unique needs. As a result, user-defined constraints plays a vital role in ensuring that the final design is not only practical and feasible, but also fulfills the client's specific needs and preferences.

Table 7: User Defined Constraints based on their preferences

No.	User Defined Constraints
1	The user-defined constraints are customized to match their individual design goals and preferences.

2.4.4 Local Constraints

Local constraints are crucial elements that must be considered while automating house plan layouts. The Albanian housing design standards defines a number of limits and criteria that must be followed while constructing a housing plan [76]. These local constraints must be respected and used as a guide when automating residential plans to generate designs that are appropriate and well integrated with the local environment. Furthermore, the standard book defines limitations concerning the minimum area of the rooms, the height of the ceilings, the requirements for natural lighting, and the systemization of the apartment's hygiene. These local constraints are based on residents' needs, as well as their well-being and safety. In Table 8 are shown the constraints regarding the living room that shall be considered while drafting a residential plan layout.

Table 8: Local constraints of Living Room

No.	Local Constraints – Living Room
1	The minimum dimensions for the living room have to be specified and must be followed in the design.
2	Having the desired orientation and adequate lighting to meet the specific requirements of the design.

Some important rules for kitchen layout and design have been suggested by the Albanian housing design standards. These rules are intended to create a functional and ergonomic environment in the kitchen, allowing the user to easily use and access cooking equipment and materials. The Table 9 summarizes some of the main rules that have to be applied.

Table 9: Local constraints of Kitchen

No.	Local Constraints – Kitchen
1	The specific design or layout of the kitchen area is defined initially.
2	The minimum area, width, and length dimensions for the Cooking Space are specified.
3	Defining the specific requirements or guidelines for ensuring proper ventilation in the kitchen area.
4	The minimum area, width, and length dimensions for the Dinning Space are specified.

Different organizational rules have been recommended for the bedroom design. These rules are intended to provide residents with a comfortable, relaxing, and conducive sleeping environment. The Table 10 shows some of the rules regarding the minimum room dimensions and size, orientation and ventilation.

Table 10: Local constraints of Bedroom

No.	Local Constraints – Bedroom
1	The specific design or layout of the Bedroom area is defined initially.
2	Defining the minimum size requirement for a bedroom of a chosen layout.
3	Setting a requirement for the minimum width of a bedroom in a specific layout.
4	The minimum dimensions for the bedroom have to be specified and must be followed in the design.
5	Defining the specific requirements or guidelines for ensuring proper orientation and lighting
6	Defining the specific requirements or guidelines for ensuring proper ventilation and hygienic needs.

Moreover, there have been established certain principles for bathroom

arrangement and design. These guidelines are intended to provide a functioning and acceptable environment for the apartment's occupants. In the Table 11 are summarized some of the crucial instructions.

Table 11: Local constraints of Bathroom

No.	Local Constraints – Bathroom
1	Specifying the proper orientation and lighting requirements for a toilet or a bathroom.
2	Ensuring proper ventilation by defining the specific requirements or guidelines for a toilet or a bathroom.
3	Defining the specific positioning by the guidelines to ensure a proper plan composition.

These suggested rules in Albanian design standards of the premises of a residential plan are prepared based on the experience and knowledge of architects and engineers, with the goal of providing suitable and well-organized premises where users can utilize an efficient environment.

CHAPTER 3

METHODOLOGY

3.1 Detached House Definition

The detached house, known as a single-family house in the literature, as a type of private residence is independent and functions separately and individually from other buildings. As part of free-standing structure, the detached house can be categorized as one, two or three stories. They are usually characterized by living systems composed of the main structure built and the garden which surrounds all it sides [77]. Typically, a single family houses are designed to provide residents with all the essential living spaces as Living room, kitchen, dining room, bedrooms and hygienic-sanitary units. They offer a high degree of privacy and independence, as the residents have complete control over their own living spaces and can customize them to suit their needs and preferences [78].

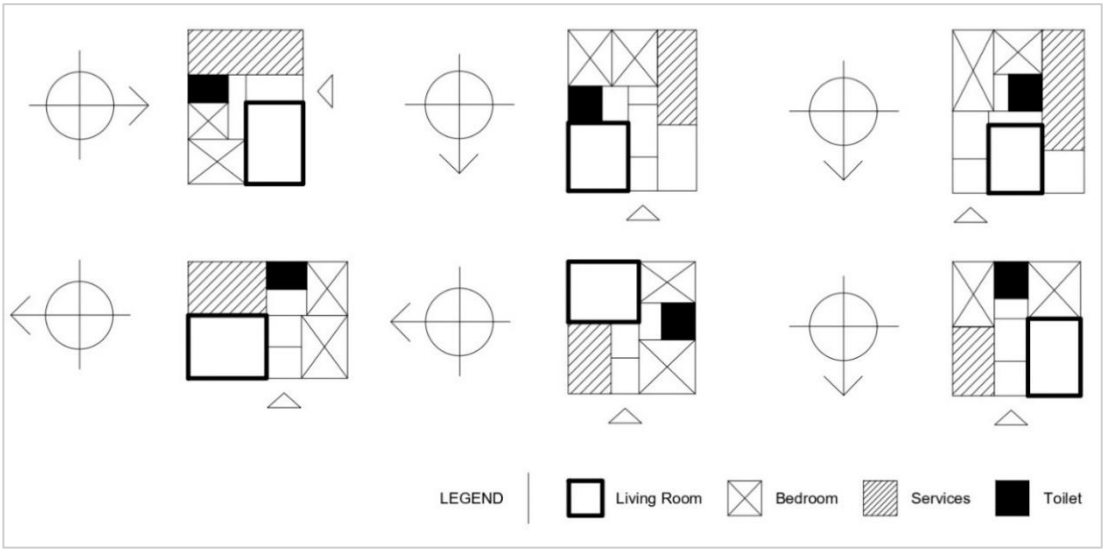


Figure 11: Main spaces of a Detached House arranged in relation to the north direction

Detached houses offer greater flexibility in terms of orientation compared to other housing types, as they can be situated in any orientation, independent of the north direction [79]. This design characteristic allows for maximum use of natural light and optimal solar access throughout the day [80, 81]. The positioning of the house can also be optimized to take advantage of prevailing winds and promote natural ventilation [82]. In Figure 11 are shown possible arrangements of the main functional zones and entrance in relation to the north direction. The layout of a detached house is designed to provide maximum privacy and separation from noisy roads. Typically, the objects are isolated from the street by a garden, and the entrances to the house can be positioned from different sides of the building to increase privacy. Furthermore, detached houses often include the potential for future extensions to accommodate growing families or changing needs [83].

3.1.1 Grouping of spaces in functional areas

The grouping of living spaces is divided into two functional areas: the day-time and night-time zone. The living area consists of the living room and the kitchen. Finding the noisiest parts of the house, they are placed near the entrance. The night area consists of the bedroom and the sanitary units, arranged next to them. The bedrooms require quietness, therefore they are located away from the entrance area.

Taking into account the vital functions, as well as the possibilities of combining them, the necessary premises of a house are shown in Table 12 referred to Albanian housing design standards [82].

Table 12: Necessary spaces of a house

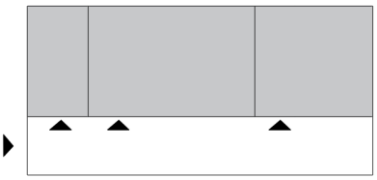
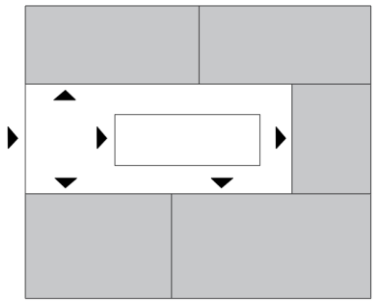
No.	Living Spaces	Auxiliary Spaces
1	Living room / Day room	Entrance space
2	Bedroom	Kitchen
3	-	Hygienic-Sanitary unit
4	-	Storages

In addition to the essential units, open spaces such as loggias, balconies, terraces, perform many useful functions.

3.1.2 Typological categories of floor plans

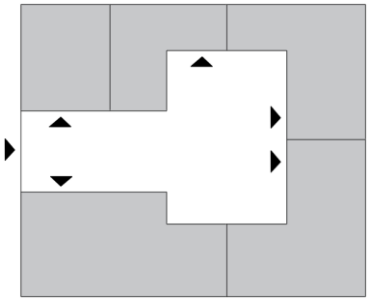
Organizing the spaces in a floor plan layout plays a critical role in determining the overall functional and aesthetic appeal of the building or structure. They should be designed considering carefully the needs and preferences of the occupants merged with the architectural design principles. In a well-designed floor plan, socializing and privacy are crucial stage, therefore, open and closed spaces can be arranged accordingly. This plays an important role in producing an efficient, comfortable and practical living space. In Table 13 are presented the categories of typologies of floor plans, based on the way of organization of the object [82].

Table 13: Typologies of floorplans based on the way of the organization referred to Albanian housing design standards [82]

No.	**Typology**	**Illustration**
1	**Floor plan with corridor** The apartment is organized according to a long axis, in which the rooms are lined up on one side or on both sides.	
2	**Floor plan with inserted boxes** The apartment is considered as a wide empty space, in which a cube (a box) is inserted, or walls, which can be a kitchen, a bathroom or a sanitary unit.	

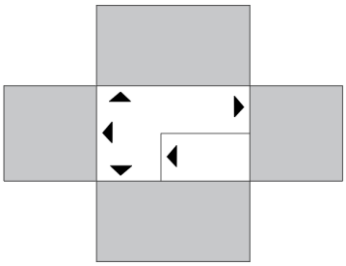
3 **Floor plan with living room, as distribution center**

The floor plan of the apartment is developed around the living room, which becomes the center of distribution and passages.



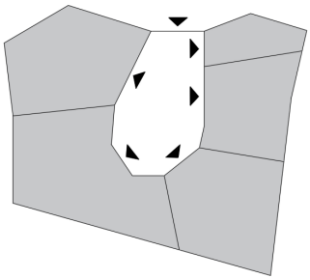
4 **Floor plan with separate functional areas**

In the plan of the apartment, the functional areas are clearly separated, with the aim of preserving privacy. Each area contains a corridor that connects to the entrance.



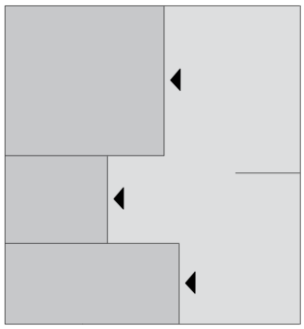
5 **Organic layout**

The layout is based on the study of the resident's movement during various activities. Walls are placed around areas of concentrated movement.



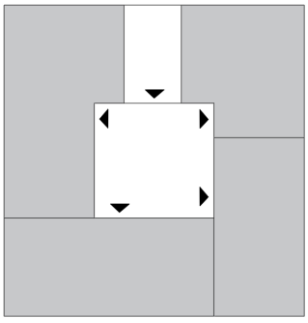
6 **Permeable layout**

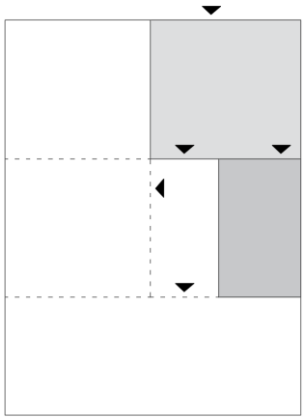
It is a variant of organic planning. Characteristic is the absence of a corridor, which enables the division into day and night areas.



7 **Circular layout**

It creates numerous possibilities of connecting functional areas between them.



8	**Variable layout** Creates the possibility of changing the internal organization, in function of the needs of the residents. The shapes and sizes of the rooms can be changed by means of movable walls.	

The typology that we have taken into consideration for this study is the circular plan, which generates different possibilities of connecting the functional spaces between them. In a circular layout, all spaces have a common center. In addition to facilitating the processing of the algorithm and coding, this typology produces a sense of harmony in organization of the space. The connection of many functional areas in a single area can stand in reducing the time to move from one unit to another, making the livability more comfortable [78].

3.1.3 Elements of a residential building

The spatial organization which is considered in this study includes these main spaces of a house layout that are defined as below referred to Albanian housing design standards [82]:

The entrance has the function of connecting the different spaces of the home, making them independent from each other and providing everyone with the necessary privacy.

The living room is the main space in the apartment, because it is where all the members of the family gather together. Therefore this space helps in. the best possible organization of family life.

The kitchen is the resident's main place for household work, which includes preparing and cooking food. The size of the kitchen depends not only on the size of

the furniture, but also on the free surface needed for their use, as well as the creation of normal conditions for the resident.

The Bedrooms is of great importance, since the most important life activity, which is sleeping, is performed. Their surface area depends on the number of people sleeping in them. The typologies of bedrooms are Master bedroom, Bedroom for two people and Bedroom for one person.

Hygienic-sanitary **spaces** are used to maintain cleanliness and personal hygiene of the residents. Its surface area depends on the number of sizes or devices, furthermore on the surface needed for their use.

Table 14 shows the norms of the surfaces of different types of housing referred to Albanian housing design standards [82].

Table 14: Standards of the surfaces of the different plan typologies

No.	**Spaces**	**1+1 typology** (m^2)	**2+1 typology** (m^2)	**3+1 typology** (m^2)
1	Master Bedroom	12	12	12
2	Bedroom	-	8	12
3	Bedroom	-	-	8
4	Living room + Open kitchen area	12 2.52	13 2.52	14 2.52
5	Kitchen Kitchen + dining area	4.2 6.16	4.2 6.8	4.2 8
6	Bathroom	3.6 – 4.2	3.6 – 4.2	3.6 – 4.2
7	Storage	1 – 1.5	1 – 1.5	1 – 1.5
8	Corridor	4 - 5	6 - 7	6 - 7

3.2 Software Programming Description

The generation of plan layouts is a crucial aspect of software programming, as it allows for the creation of effective designs for various applications. In recent years, the use of Python programming language has become increasingly popular for the

implementation of such algorithms [84]. Python is a free, widely used programming language that is easy to learn and quick to implement [85]. Its syntax is short and clear, which can be used even by non-professionals. This makes Python a good choice for plan layout automation, especially for architects. The increasing demand for python over years, is characterized by huge user community as well as massive open resources. Therefore, programming with Python language is practical and supported by large number of examples throughout forums and guidelines [85].

This language provides numerous and powerful libraries that facilitate and simplify the process of programming and developing various applications. Some of the libraries implemented in this study are listed and explained in the Table 15.

Table 15: Definitions of the Python libraries implemented in the code

No.	Library	Definition
1	time	Used to apply delays in seconds for different stages of algorithms provided in this study. Usually implemented to detect various functions by slowing the processing time for debugging purposes [86].
2	shapely	Utilized while dealing with constraints stage of the overlapping spaces. It is a very wide library which provides multiple functions, but in this study it is used for determine either several polygons are crashing with each other. This would label the plan as incorrect [87]
3	import.lib	Implemented to export the data of the algorithms into to .txt files which will later be recalled by other functions in different phases [88].
4	tkinter	A graphical user interface (GUI) is demonstrated in this study for a better visualizations of the plan layout automaton using tkinter library [89].
5	datetime	Additionally to the main thesis scope, it is presented an advanced stage for faster processing procedures using parallel programming in several CPUs. To accurately measure the duration, datetime is implemented as a suitable library of python [90].
6	intertools	Powerful library which generates several permutation combinations [91].

3.3 Programming structure used for automation procedure

Plan layout automation is a complex process that needs a lot of coordination and detailed organization. The automation of housing plan has many advantages both in terms of speed of work and in terms of increasing employee productivity. The division of the work presented in this study is developed into three phases due to a better management of the work organization as well as facilitating the modification or updating of the code at any moment during the work. This will be done by dividing the code into small parts, known as functions, for each of the phases. Also, this strategy will allow the modification of the code in an easy way, minimizing the risk of negative impact on other parts of the code. By sorting out the work areas into phases, it becomes possible to give the code the necessary attention in each stage and in this approach an efficient product can be generated while it suitable for debugging at any time.

The first phase presented in this study deals with the development of the algorithms prepared to ensure the basic, unfiltered, invalidated plan layouts. Therefore, in the end of phase one, it is expected to have a set of plan layouts in a raw format. The entire process is ensured by considering multiple calculations and filters. The code itself is structured into functions for better management and execution. This stage will contain all the necessary information to organize the living spaces in a plan layout. Moreover, some algorithmic restrictions and rules for the spatial organization will be established, in order to ensure that possible patterns are generated correctly and acceptable as a functional plan layout. These limitations are the basic elements in the organization of a plans such as, overlap prevention between two or more units and permitting the connection of each component with the central access. This will guarantee error-free plans for the upcoming stages and hence increase the efficiency of the algorithm. The entire process, from the user inputs in terms of space functionalities, orientation-direction, until the generation of the raw layouts is developed using a graphical user interface for better visualizations.

The second phase consists in the process of filtering the plans generated before based on validation rules and criteria. The development of plans in the first phase offers a wide range of possibilities, but not all of these combinations are acceptable or reasonable for the user. In phase two, several restrictions are configured in the code based on regulations, guidelines and suggestions taken into account from a wide

literature review as well as Albanian housing design standards [82]. All these criteria and filters provide an overall validation of the plan which in evolutionary procedures is known as Goodness value parameters. The goodness value will be used to identify the best and suitable plans to use. The criteria set to establish the limitations and evaluations of the plans depend on the purpose of the residential plans and the needs of the user. At the end of this phase, it is stored a set of housing plans that are evaluated by the Fitness functions. This set of acceptable plans are efficient and acceptable to the user and professional designers.

The importance of the third phase lies in the graphic visualization, since the algorithm must be suitable for understanding by the user and for the stockholders. To reach this goal, the plan layouts validated in the second phase, are exported to drawing software. As one of the well-known commercial software used widely by architects is AutoCAD, phase three ensures the development of a script to re-draw the plan details in this software. Consequently, all validated and selected plans can be transferred to be re-generated more easily at this stage. The architect has the opportunity to make the necessary changes to complete the architectural project. In this way, the automation of the housing planning process can greatly facilitate the work of architects and encourage more effective and efficient work in the architectural industry. In Figure12 it is presented an algorithmic flow of the general programming structure used for the automation procedure of this study.

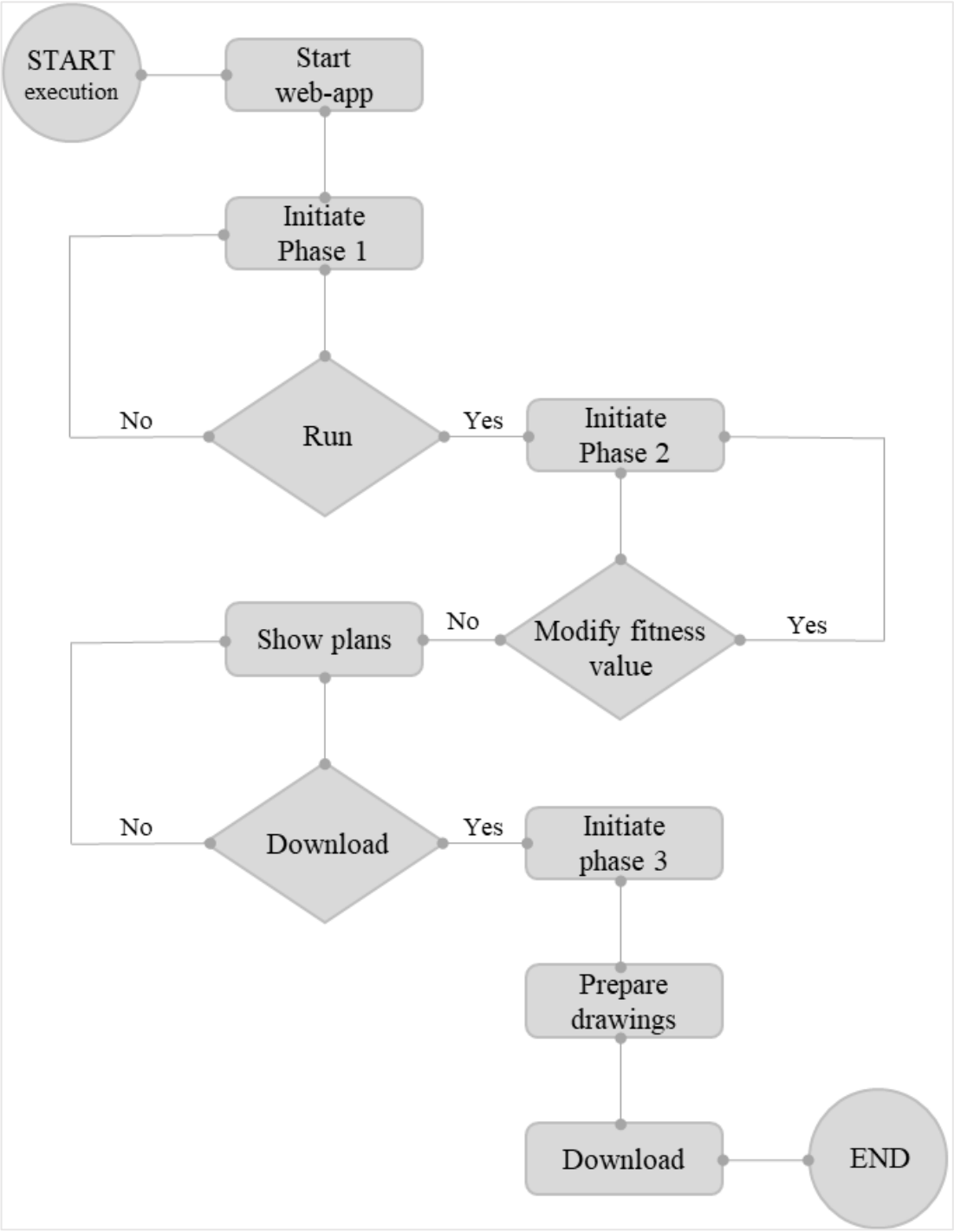


Figure 12: Algorithmic flow chart of the general programming structure used for the automation procedure

3.4 First Phase

The initial phase of the methodology deals with the organization and operation of housing plan automation. This stage starts with determining the number of spaces desired by the client or the architect. Following this decision, the path is calculated and the possible combinations of the spaces are generated considering the orientation and corner prevention constraints. It is important to mention that all these spaces will circulate around the corridor, which will be referred to as a stationary point. In order to reduce the number of possible combinations, generated layouts passes some conditions that are the primary rules in an appropriate architectural design plans. These rules include overlapping between two or more spaces and filtering the entrance. All the relevant steps that will be developed in this phase will be explained in detail below.

3.4.1 Space layout assignment

One of the main factors considered during the automation of housing plans is the determination of the number of spaces. This is an important parameter that affects the organization and operation of a plan layout. Accordingly, they can be defined by the client or the professional designer and serves as a starting point for the algorithm to generate a floor plan.

Since this study focuses on the typology of circular layout floor plans, the first space that is taken into consideration is the distribution space, otherwise known as the corridor or hall, which functions as an immovable stationary point in the organization of the spaces during this algorithmic processing. The hall provides access to other spaces and can be changed based on the standards of housing design or the needs of the user. These standards may include specific requirements for room size, orientation, and adjacency. The designer must consider the needs and preferences of the customer to decide the number of spaces to ensure the minimal inputs for the automation procedure. However, by default configurations in the current automation algorithms for the hall parameters are defined based on the guidelines [82]. In the plan layout automation, the number of spaces is a crucial element, as it has an impact on the overall size and form of the structure or building. Moreover, the location of hall as well as

other rooms, plays an important role for the procedure developed in this study as it belongs to the centroid based method.

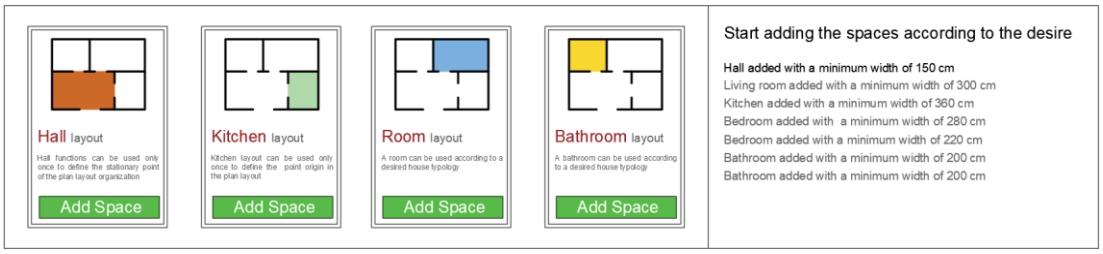


Figure 13: A conceptual idealization of spatial layout inputs

A conceptual example of the parameters that the user can choose to start the algorithm's work in relation organization of the spaces is presented in Figure 13. All spaces are available to the user and has the opportunity to choose them according to the desired or required typology of the residential plan. These typologies can range from 1+1, 2+1 or 3+1.

3.4.2 Space Orientation

The orientation of spaces is crucial in building design since it has an impact on many aspects of the daily life of residents. This is an important parameter to understand how to plan the housings so that they have a thermal stability, a natural ventilation system and a good natural lighting. Specifically, the spread of natural light is a key factor in building a brighter and comfortable environment for occupants. In general, the orientation of the spaces according to the directions North, South, East and West should be adapted to the conditions of the climate and environment where the residence is located. A good optimization of the orientation plan can help reduce energy costs and improve the quality of life at home.

The regulation of the Albanian standards gives some specific suggestions for the orientation of the spaces according to the North, South, East and West directions [82]. Table 16 defines the parameter criteria for arranging these spaces.

Table 16: Space orientation Matrix Input

	LR	**D**	**K**	**MB**	**B**	**MBA**	**BA**	**S**
North	-1	+1	-1	-1	-1	+3	+3	+3
South	+3	+3	+2	+2	+2	-1	-1	-1
East	+1	+2	+3	+3	+3	+1	-1	-1
West	+2	+2	+1	-1	-1	+2	+2	+1

The space orientation matrix shown above helps to determine the distribution of different spaces in a house based on the four cardinal points of the compass. Using this table, the values of each space indicate their orientation relative to the directions. Negative values (-1) in the matrix indicate that a space in a house should not be oriented in that direction, while positive values (+1, +2, +3) indicate the suggestion to orient the space in that direction. The higher the value of a space in the matrix, the stronger the suggestion of orientation in the given direction.

The table shows that the corresponding values for the orientation of the living room have a strong suggestion for the orientation towards the South and a lower preference for the orientation towards the North. Likewise, the dining room and kitchen have a preference towards East and South, with values ranging from (+2) to (+3). In this way, the optimal use of natural light will affect the reduction of electricity for lighting as well as the required ventilation of these spaces.

On the other hand, for bedrooms Table 16 shows a strong suggestion for orientation towards the East reaching the value (+3). If it is oriented in this direction, it will be exposed to natural sunlight in the morning. This can affect the human's circadian rhythm (body's natural rhythm), helping in waking up easier in the morning and feel more refreshed [92].

The orientation of the utility spaces in the North direction with the suggested value of (+3), allows natural light to enter optimally and can ensure ventilation. This orientation can allow fresh air to enter the toilet naturally and lead to a cleaner environment. However, to place these spaces in the north direction, it is important to consider other factors of the project. Architects must ensure that it is placed in a position that will not interfere with other spaces in the house and will provide an

appropriate level of privacy. In addition, the individual needs of residents and personal preferences regarding the location of such units in the home should also be taken into account.

Therefore, it remains crucial to emphasize that the space orientation matrix generated for plan layout automation does not guarantee a finalized version for the organization of the spaces in a detached house. Indeed, the utilization of the matrix must be taken into account in ratio with additional factors such as building location, climatic conditions, terrain configuration and more. Nevertheless, the orientation guidelines must be taken into account as they provide important tools for engineers and architects in the house planning process.

3.4.3 Path coordinates calculation

The calculation of path coordinates will be introduced after the user has specified the number of spaces and their direction. The concept is based on the principle of "space-traveling" around the main unit of the house, the hall. In this way, the hall remains static while each of the spaces defined by user performs a 360-degree rotation around it as shown in Figure 14.

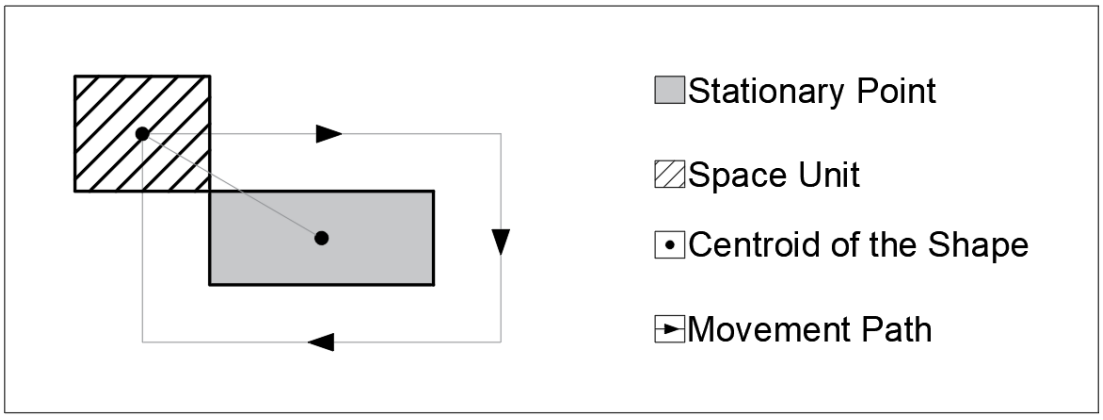


Figure 14: Example of traveling path for one space unit

For coding purposes, the rotation of each space is defined by the "path-coordinates". This involves the calculation of several coordinates to ensure the 360-degree travel-path around the main unit. As demonstrated in Figure 14, the calculation of path coordinates is required independently for each of the spaces as they have different sizes. Moreover, based on the orientation inputs provided by the user, the path coordinates are calculated accordingly only for the suitable directions, North, South, West and/or East as shown in Figure 15.

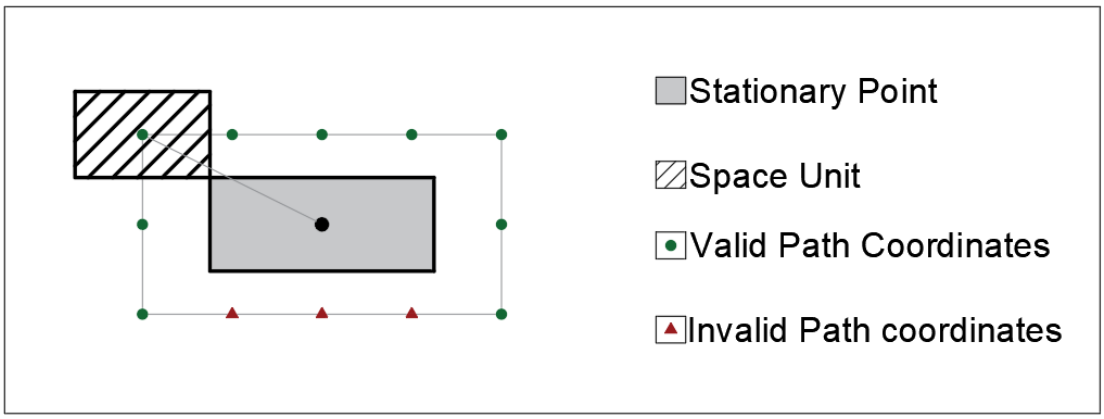


Figure 15: Path coordinate calculations. The concept of coordinates for one space example, and the path coordinates for a space which is forbidden to be oriented in one of compass direction

The coordinates are calculated based on several parameters. Initially the dimensions of the hall play an important role because it is taken as the reference point for each coordinate. In addition, the size and orientation of each space may influence the path coordinates and shall be considered separately. Furthermore, the coordinates must only be adopted for the directions suggested by guidelines.

3.4.4 Corner prevention

In order to fix and prevent connectivity errors between the stationary unit and other spaces in the automation of housing plans, it is important to apply certain logical constraints. One such criterion is the filtering of each space coordinate to ensure that

the hall maintains a minimum distance in the most critical area, defined as corners. To achieve this task, each coordinate of the spaces travelling around the hall is checked through a filtering process.

The filtering process requires that each coordinate of the spaces be checked and changed in order to maintain a minimum distance in the corner. This means that if two spaces are located at a very close distance in the corner, then they will be arranged in a pattern that they have at least a minimum distance from each other. This will allow communication between rooms and prevent non-functionality of the circulation plan. Therefore, on the right of the figure shown below, the constraint parameter is represented by the label "a"

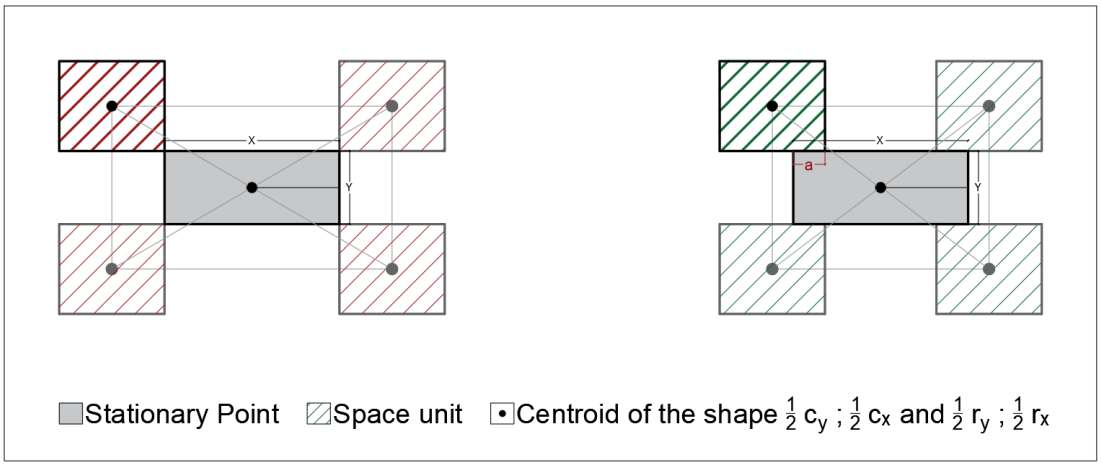


Figure 16: Demonstration of minimum distance ensured for space connectivity on corners

The equation which would adequately determine the location of the room regarding the centroid of the Hall would be as shown below:

$$X = C_X + \frac{1}{2}R_X + \frac{1}{2}R_X \quad (1)$$

$$Y = C_Y + \frac{1}{2}R_Y + \frac{1}{2}R_Y \quad (2)$$

Where: C_x is the width of the Hall, C_y is the length of the Hall, R_x is the width of the room $_i$ and R_y is the length of the room $_i$. These two equations can generate all possible coordinates for the position of the room.

The preliminary set of coordinates on the left demonstrates a 360-degree

movement around the stationary point in any direction. However, the algorithm should be adopted according to logical cases. Therefore, the equation of the motion coordinates would change by integrating the corner constraint, as shown below:

$$X = C_X + \frac{1}{2}R_X + \frac{1}{2}R_X - 2a \quad (3)$$

$$Y = C_Y + \frac{1}{2}R_Y + \frac{1}{2}R_Y - 2a \quad (4)$$

The modified equations must be included into several calculations to define all the coordinates needed. The methodology developed to calculate the path coordinates is based on two nested loops in python programming language, for both orthogonal directions of the spaces. Some criteria are implemented to achieve the valid coordinates and ignore the ones which are not needed using "if" functions. Finally, each of the path coordinates are saved in different lists written in the code.

3.4.5 Generation of possible room combinations

As the path coordinates are calculated and saved for each space individually, then they should be simulated simultaneously around the hall, so it can generate different layouts. This step can be achieved in several ways by using different procedures and libraries in python. However, it requires a lot of computational power and time as the simulations have to check for all possible combinations. Therefore, in this study it is presented an alternative way of generating room combinations. By using the permutation sequences of the set of coordinates among each space defined it results in all possible combinations. Hence, this solution guarantees that none of the possible layouts that can be generated are missed. A graphical interpretation of how the permutation combination is utilized in the code is shown in Figure 17.

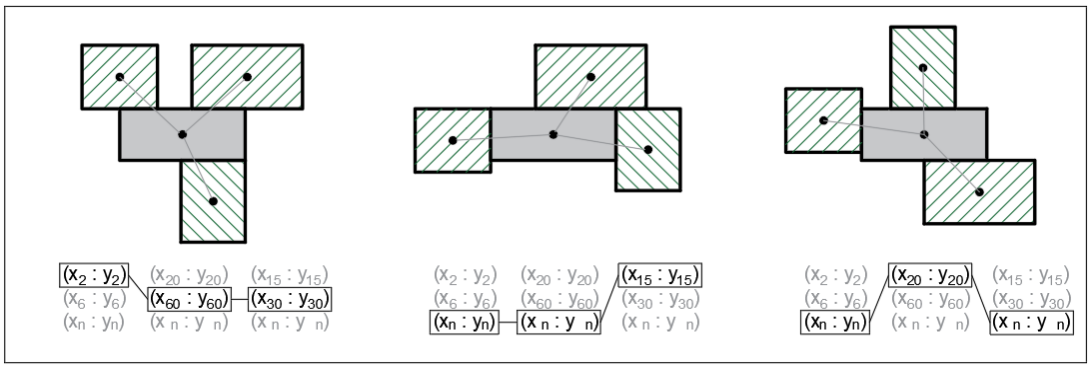


Figure 17: The graphical description of the combinations using path coordinates for different spaces, and the respective result of space arrangements

Nonetheless, it should be emphasized that this results in a huge number of space arrangements which consider non-valid plans. This means that the permutation combinations will provide any regular or irregular (spaces overlapping with each other) therefore, additional operations must be taken into account for filtering illogical layouts.

3.4.6 Fixing clashes

After generating the coordinate combinations, it is crucial to define functions to filter overlapping of two or more spaces. Each of the combinations is checked individually for overlapping between spaces. The overlapping results are returned from a python library known as “shapely” [87]. Combinations that include overlapping spaces are automatically deleted while the others are saved in a list in python. The checking criteria is done for spaces and entrance separately as follows.

3.4.6.1 Overlap prevention of two or more spaces

Working on a plan layout it is important to find a balance between functionality and aesthetics. Overlapping spaces can cause conflicts in both aspects, making the

production process unacceptable. Therefore, it is necessary to develop an algorithmic condition that can help in preventing these situations. Overlap occurs when two or more spaces occupy the same physical space, partially or completely. To prevent spaces from this phenomenon, in this study an algorithmic criterion is used to estimate the distance between spaces. This constraint will be an automatic function that would serve to identify any potential risk of overlapping spaces and prevent any possible conflict between them.

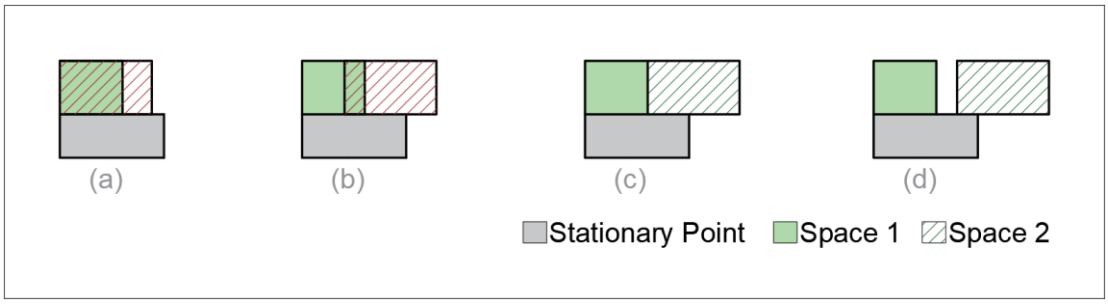


Figure 18: Illustration of different cases of possible combinations and overlap evaluation

As presented in Figure 18, an illustration of several combinations of two spaces of a residential plan where the corridor is the central point is presented. Cases (a) and (b) demonstrate a typical example of overlapping that may occur during the automation process of generative design. This is a phenomenon that can cause functional and aesthetic conflicts, which results in illogical layouts and therefore have to be filtered by the algorithm during the evaluation process. Similarly in cases (c) and (d), the two spaces are combined so that they do not overlap with each other. This makes the evaluation process to continue, as they pass the overlap criterion successfully.

With this algorithmic constraint, automation can increase the efficiency and quality of the generation process of spatial arrangement, enabling a higher speed and precision in the evaluation of combinations and guaranteeing a higher quality of the work done.

3.4.6.2 Filtering Entrance

Unlike the other spaces, the entrance is programmed as a one-dimensional shape in the algorithm. The direction and width of the entrance is defined in the same manner as other facilities. Nevertheless, the clash with circulating spaces must be avoided. Therefore, a function is added particularly to prevent other spaces locate through the entrance as shown in Figure 19.

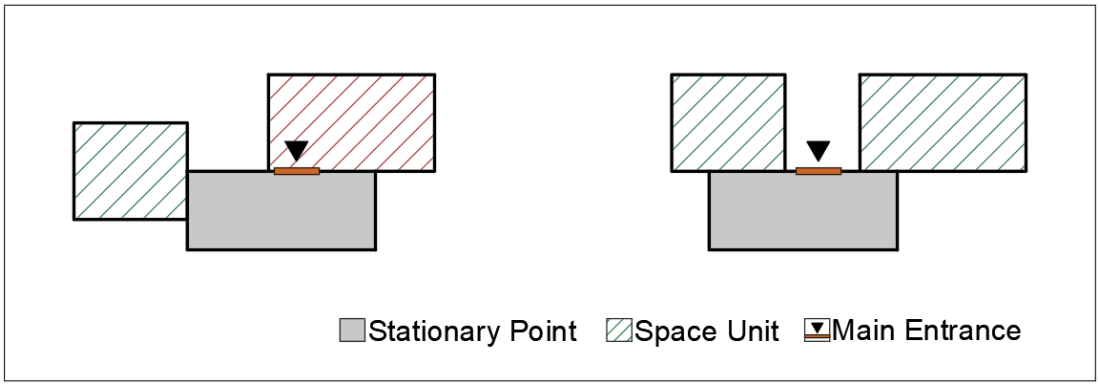


Figure 19: An example for the correct location of rooms in a plan layout referred to entrance (right), and incorrect location of space unit through the entrance (left)

3.4.7 Raw plan layout generation

To finalize the first phase of automation, a process of generating raw plan layouts and storing them in a Python list is needed. These plans are generated using the previously defined parameters and criteria although accurate, are not evaluated in terms of the fitness function. Therefore, to evaluate them, a specific algorithm is needed that will be implemented in the second phase.

In the Figure 20 it is shown a summary of the algorithm developed and used for first phase to achieve the generation of raw plan layouts.

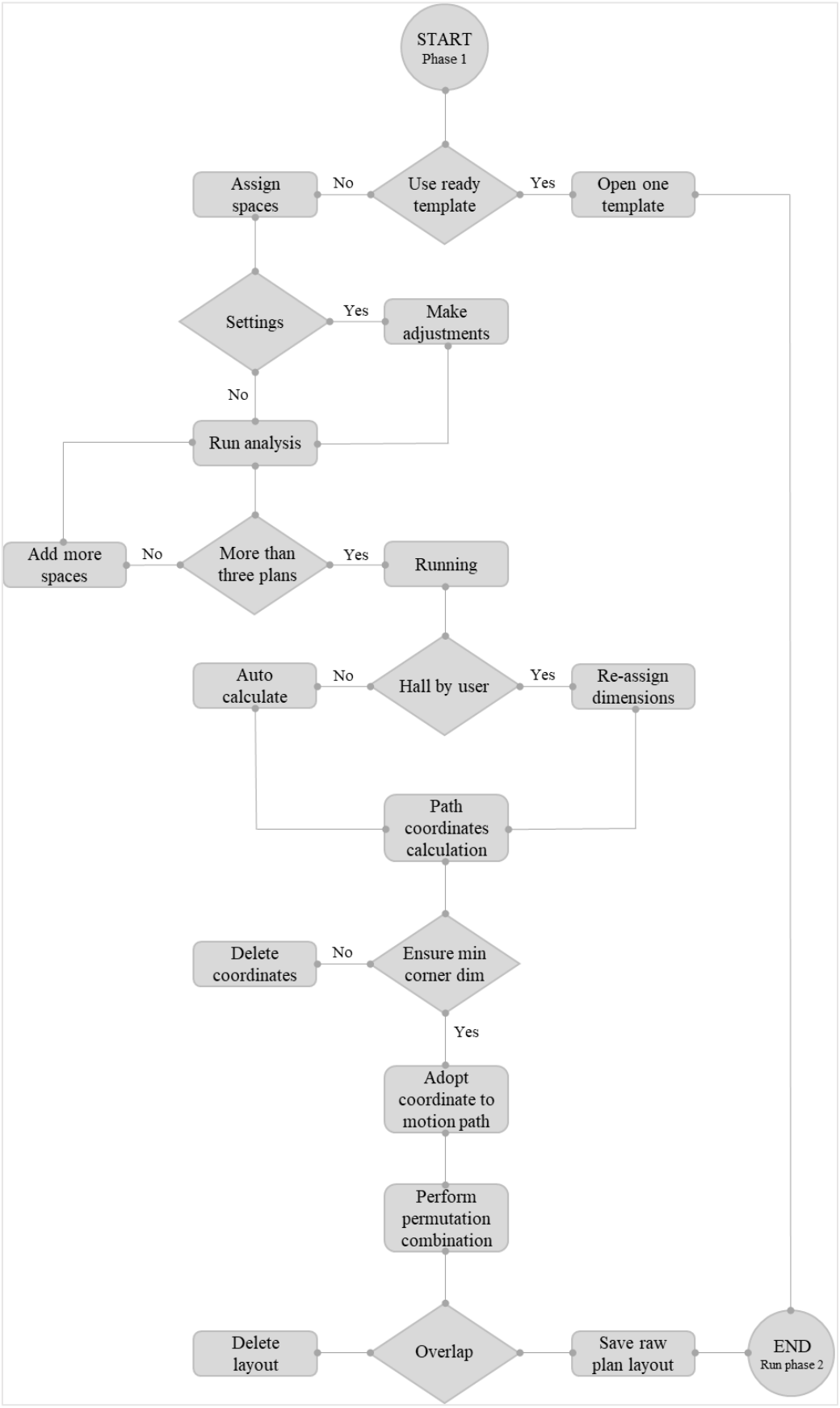


Figure 20: Algorithmic flow chart of first phase used in plan layout design automation method

3.5 Second Phase

The second phase of the methodology involves filtering the combined space units from the first stage based on the validation criteria. All plans will pass certain validation criteria defined in this study known as fitness value. These criteria include three main factors that influence the validation of a plan, which are compactness of the plan layout, functionality of spaces and parcel occupancy. All these values will finally be determined in the goodness value target. All the relevant steps that will be developed in this phase will be explained in detail below.

3.5.1 Validation of a good architectural plan layout

Housing is a fundamental human need, and developing good residential plans is a great challenge for professional designers. At the present time, where technology has influenced many aspects of living, automation has become a significant option for improving construction processes and evaluating the quality of residential plans. A good architectural layout provide suitable environment for living and developing daily activities. This aims to provide the maximum of usability in the minimum of space, responding to the needs of the occupants and incorporating aesthetic and functional elements. Moreover, it affects the well-being and health of the inhabitants [93].

Validation of a good housing plan is a process of overall evaluation of the plan to ensure that it meets all the necessary criteria and standards. This section explores in depth the validation process and relevant factors for a good architectural layout in a generative design context. In its focus is the selection of different values that have the validity of a good arrangement. Moreover, information is gathered to filter the plans and its verification, requiring it to reach the standards essential for a feasible design. In this study, the validation of appropriate layouts is done based on the criteria of Perimeter over Area, Room adjacency and Parcel occupancy.

3.5.2 Perimeter over Area (Compactness)

The ratio of the perimeter to area and its importance in the automation of residential plans has vital role in the filtering and validation criteria [94]. The perimeter of a residential layout is the total length of the exterior boundaries of the plan, while the area is about the interior space of it. The area is calculated by multiplying the width and length of each units. Perimeter and area are two crucial elements in producing and defining residential plans. The perimeter helps identify the exterior dimensions of the plan, while the area defines the space available for use in planning the interior of the building. Thus, adding the importance of the perimeter principle in relation to the surface of the building shows that the smaller this ratio, the better is the output of a residential plan. When the perimeter is reduced in relation to the surface, there are obvious advantages in terms of efficiency and aesthetics of the building. In addition, a small perimeter has clear benefits in terms of cost-effectiveness. The smaller the perimeter, the less materials and resources will be needed to construct the building. This results in a reduction in construction costs and the use of resources needed for construction.

The scholars have been interested in developing an accurate measure of compactness. Ritter developed in 1822 a basic ratio of a shape's perimeter (P) to area (A) to determine compactness of the building plan layout [95]. Several more measures have since been proposed by other researchers [96, 97, 98, 99]. These measurements are roughly classified into four groups: reference shape, geometric pixel properties, dispersion of area elements, and area-perimeter measurement.

Ritter's (P/A) approach is classified as an area-perimeter compactness measure. Nevertheless, this approach has a drawback as the measure varies when the shape's size changes. To overcome this issue, the measure can be made by dividing the shape’s area by the square of its perimeter. Among the several different names and mathematical formulations [96, 97], the IPQ [100] has emerged as one of the most commonly applied compactness measures in this field, which is defined as below:

$$C_{IPQ} = \frac{4\pi A}{p^2} \tag{5}$$

In the formula for the IPQ compactness measure, the perimeter P is squared to eliminate the impact of scaling. The area of the shape is denoted by A, while the

compactness value is represented by C_{IPQ}. A shape with a higher C_{IPQ} value is considered to be more compact than a shape with a lower C_{IPQ} value.

The execution of the compactness filtering criteria is based on two main parameters, the area of the plan layout and the outer perimeter. The calculation of the plan is based on the cumulative area of each space which is found by multiplying the width by its length. On the other hand, the perimeter of the plan is calculated using the shapely library of python. Shapely provides multiple information as regards the lines, rectangles, circles or irregular polygons. It is important to convert each of the room's information to coordinate so that it is adopted for the shapely library. The end result provides the total exterior perimeter of the building. Considering these two parameters, the necessary formula is utilized to find to calculate the compactness value. Furthermore, the value is remapped from 0 to 100 so it can be unified and compared at the same time with other filtering criteria which are described in following chapters.

3.5.3 Room Adjacency (Functionality)

Room functionality has an important role in plan layout design as it directly impacts the usability and efficiency of a space. Rooms that are designed to be near each other or have sharing walls offer several benefits as ease of movement, improved communication, and increased accessibility.

In the following section, a matrix Table 17 will be presented, which has been developed as a result of extensive research including a review of relevant literature, architectural plans, and design strategies [101, 102, 103, 104, 105]. The matrix table serves as a valuable tool for assessing the optimal spatial relationships between different rooms or spaces within a building, based on the preferences and needs of the occupants. The matrix assigns values to different pairs of rooms based on different functions of each room, the frequency of use, and the potential for interaction between spaces. The values range from -1 to +3, with -1 indicating that the two rooms should not be located next to each other, 0 indicating that there is no preference, and +3 indicating that the two rooms should be located right next to each other.

Table 17: Adjacency Matrix Input

SPACE TYPE		**LR**	**D**	**K**	**MB**	**B**	**MBA**	**BA**	**S**
Living Room	**LR**	0	+3	+2	-1	+1	-1	-1	-1
Dining Room	**D**	+3	0	+3	-1	-1	-1	-1	+2
Kitchen	**K**	+2	+3	0	-1	-1	-1	+1	+3
Master Bedroom	**MB**	-1	-1	-1	0	+2	+3	-1	-1
Bedroom	**B**	+1	-1	-1	+2	0	+1	+2	-1
Master Bathroom	**MBA**	-1	-1	-1	+3	+1	0	+2	-1
Bathroom	**BA**	+1	-1	-1	-1	+2	+2	0	+1
Storage	**S**	-1	+2	+3	-1	-1	-1	+1	0

In this table of matrix, the highest adjacency preferences are between the Living room, Dining room, and Kitchen, as they are often used together and it is desirable for them to be located in close proximity. The Master Bedroom has a high adjacency preference with the Master Bathroom. Additionally, all of the bedrooms have a low adjacency preference with the living areas.

A. Living Room is to be shared by the kitchen and dining area.

(LR, D) = +3, (LR, K) = +2

B. Bedrooms are preferred to be grouped

(MB, B) = +2

C. Storage is to be shared by the kitchen and dining area

(S, K) = +3, (S, D) = +2

D. Master bathroom should be shared by the master bedroom and bathroom

(MBA, MB) = +3, (MBA, BA) = +2

This matrix can be a useful tool to guide the layout and design of a house to ensure that the different rooms are located in the most desirable and functional way possible.

The execution in python programming language is done by considering in the

main part the matrix developed before. The values used to validate the sharing walls between different spaces are configured in the algorithm. While evaluating the plan, for each of room pairs it is considered the evaluation value and assigned to the functionality. Once all combinations are executed from the nested loop, then a general value is calculated for this criterion. Afterwards, the remapping is applied for the unification of values and comparison reasons with other criteria used for the validation of the plan layouts. In addition, the matrix used for the sharing walls is provided in the web-app which can be updated by the user with different values according to the demand.

3.5.4 Parcel Occupancy

The parcel occupancy fitness criteria is based on the idea of the percentage of the plan occupying the parcel's surface. The parcel is defined as the space from the external boundaries of the building considering the outmost coordinates of the generated plan. In other words, the occupancy of this parcel defines a good fitness value. Therefore, a maximum parcel occupancy fitness value will be represented by a rectangular plan with no remaining unused spaces as shown in the figure below. By using this value it can be easily determined the efficiency of using the maximum possible parcel surface.

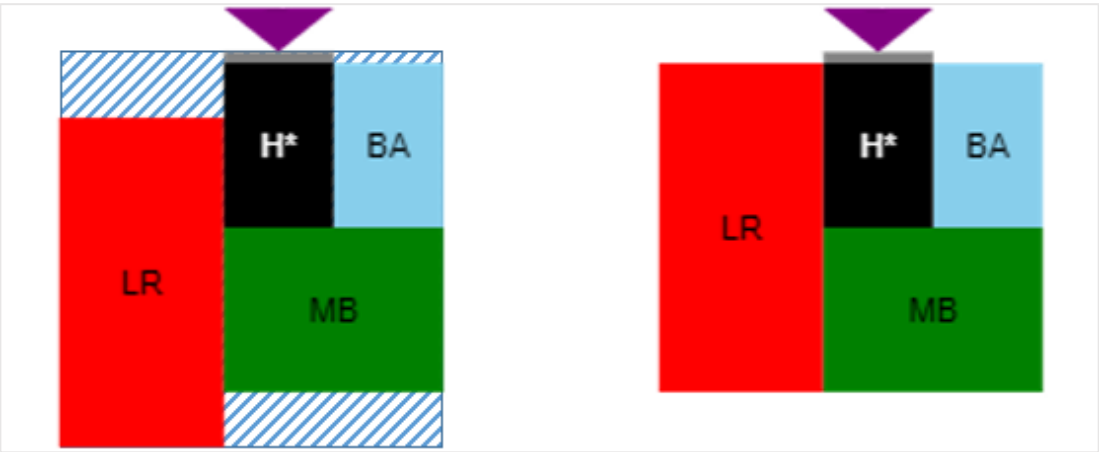


Figure 21: Validation of parcel occupancy, not fully occupied parcel (on the left), fully occupied parcel (on the right)

The implementation of the parcel occupancy in the algorithms of this study is done by considering the ratio of overall plan surface and the parcel area. The plan surface is calculated from the example mentioned in the section above. While the parcel area is calculated considering the four outermost boundaries of the building. The code is written to get the maximum and minimum coordinates of the west-east and north-east directions. Once these coordinates are ensured, then the calculation of the parcel width and length and as a result, the area is easily performed. In addition, the ratio of building vs parcel area for each of the plan layout generated is saved in a python list after remapping their values from 0 to 100. In the end, this filtering criteria shows the percentage of the plan fulfilling the parcel from its outer boundaries.

Every fitness value calculated is then multiplied by different coefficients which are given by default according to literature suggestions but can be modified very easily from the web-app according to user preferences.

3.5.5 User interaction

The second phase is developed in the environment of python programming language. All calculations are structured in various functions which communicate among each other. The final output of the results is visualized in the graphical user interface (GUI). For the scope of this study, the GUI for the second phase is developed as web-app utilizing the flask library. This is done to have a more responsive and dynamic interface which enhances interaction and builds the bridge between the code and user. As shown in figure# the user is prompted in the top of the web-app with the evaluation criteria. The compactness, functionality and parcel occupancy can be set according to the most preferable combination by the user or architect. Furthermore, the values which are used to validate the functionality while considering the sharing walls (Adjacency Matrix) can be edited and updated. Users may update one or more values and click the button “Update Parameters” to reflect the changes. In this way, the analysis of different categories, combinations and goals can be achieved.

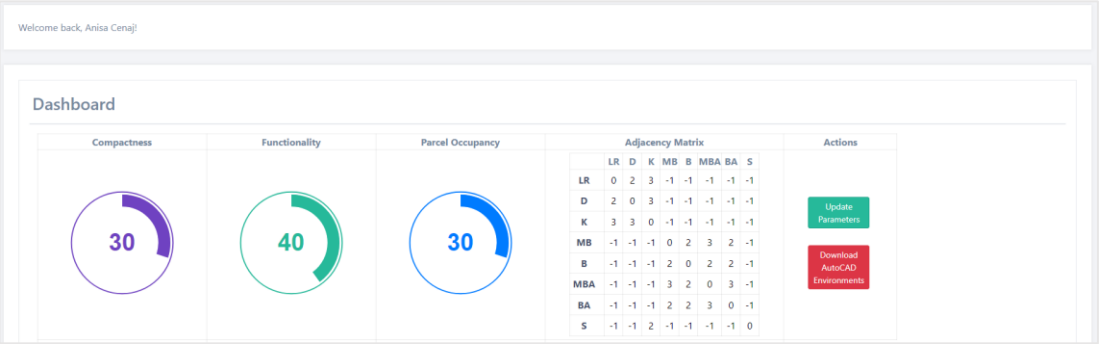


Figure 22: The graphical user interface to update fitness values and Adjacency Matric

Once the parameters are set, then the application demonstrates all the generated plan layouts according to the highest goodness value as shown in Figure 22. The window provides important information regarding the plan number or ID, the compactness, functionality and parcel occupancy values remapped, parcel area in square meters, building area and building perimeter. Furthermore, it gives the goodness score in association with the percentage and rating stars. In addition, the option to export the plan in AutoCAD drawings using the automation CAD script is provided. Lastly, a fast preview of the plan configuration is shown below the information mentioned above.

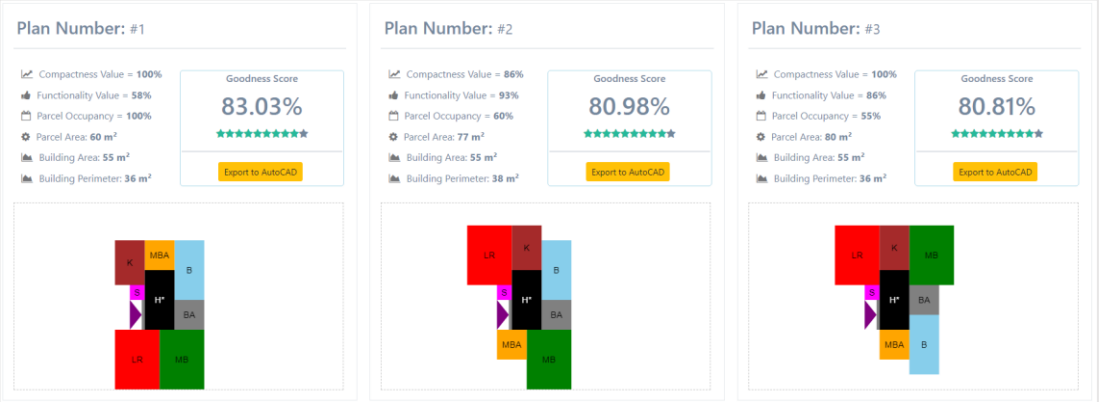


Figure 23: Web Application prepared for the outcome presentations and user interaction

In the Figure 24 it is shown a summary of the algorithm developed and used for the second phase to achieve the validation process of plan layouts.

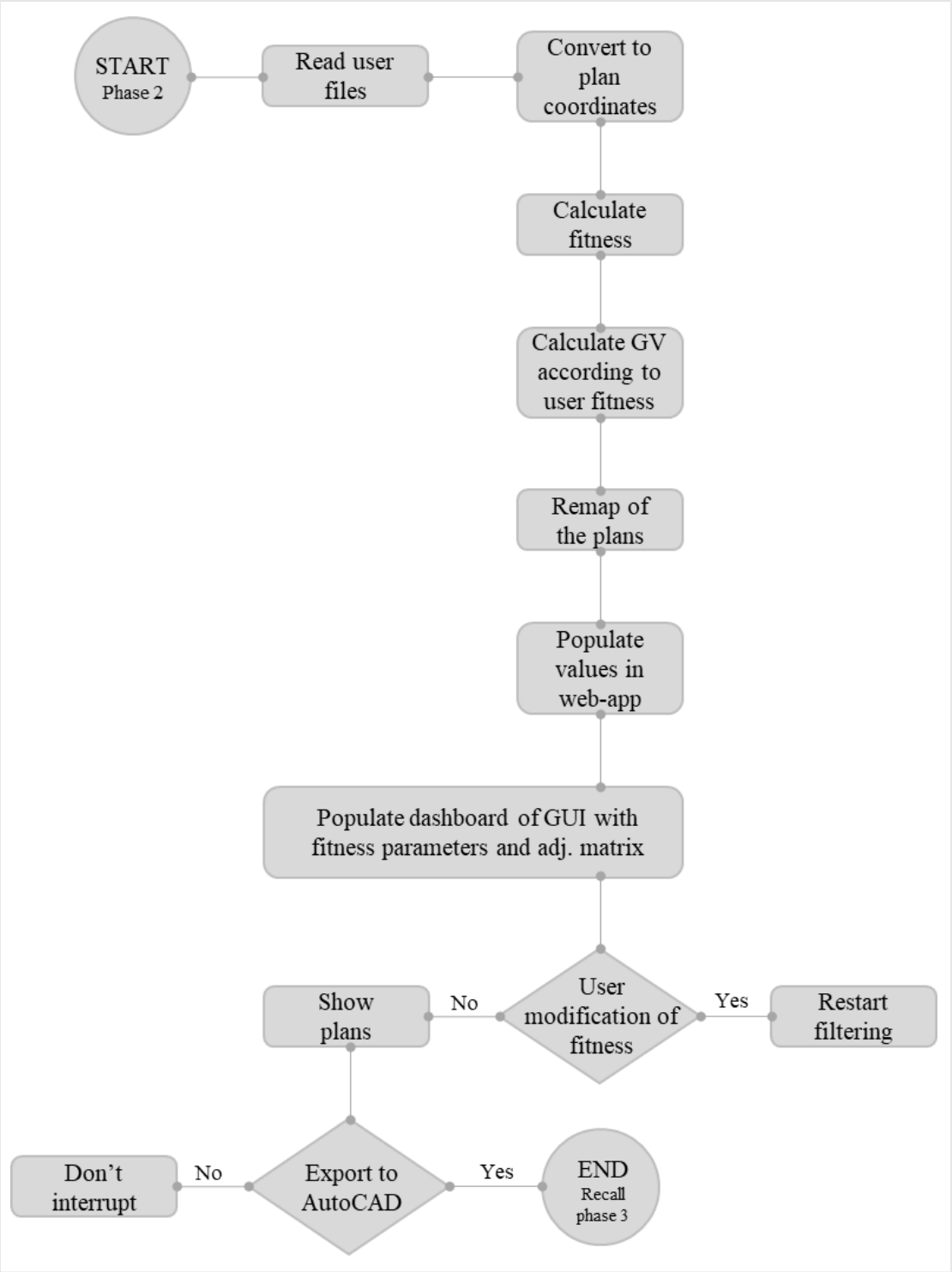


Figure 24: Algorithmic flow chart of the second phase used in plan layout design automation method

3.6 Third Phase

The third phase of the methodology deals with extracting generated plans in AutoCAD software. This is an important step in visualizing all the project data to architectural drawing layouts. At this stage, each plan selected for extraction into AutoCAD will be considered as an individual project package, containing all the data required to be assigned as an architectural plan. Complex plan information is simply accessed with AutoCAD scripts and includes visualizing the dimensions of the plan, identification of the arrangement of important elements, and other graphical representations. Using scripts to extract final plans is also an efficient tool for architects and designers. This frees them up to focus on the other parts of design, since they are no longer loaded by the technical effort of collecting and arranging data in AutoCAD. Furthermore, because the automated extraction of plans uses the data generated in the previous second phase, the possibility of probable errors are lessened.

3.6.1 Definition of AutoCAD Scripting

AutoCAD scripts are a set of written commands that are run to conduct different tasks in the AutoCAD application. Scripts allow users to automate repetitive actions including drawing objects, editing them, altering parameters, importing data, and many more. These scripts offer a high level of flexibility and speed when designing and visualizing. To develop scripts in AutoCAD, a simple text editor like Notepad or an advanced AutoLISP editor like Visual LISP can be used. After generating the script, the file is launched within AutoCAD by using the "SCRIPT" command and providing the script's location, or simply by dragging and dropping the file onto the workspace.

3.6.2 Preparation of the AutoCAD environment

All AutoCAD scripting commands in this study are processed in Notepad++

file. Preparing the workspace in AutoCAD begins by deleting any existing objects in the workspace. This phase is crucial in ensuring that the new design starts with a blank drawing page. Following this, it is necessary to clear all groups or layers, to ensure that the script runs without errors when updating the new drawing. The purge command will be used to clean out groups or layers. This will remove any unnecessary element from the drawing file. After emptying the workspace, it is ready to start the new design in AutoCAD software. This work environment preparation generates a clean basis for the next design layout.

3.6.3 Drawing outer walls

The respective area included in that category of drawing. The methodology followed to draw the outer walls is based on the outer coordinates of each of the spaces involved in the plan layout after the automation of the second phase. These coordinates are filtered for possible errors and then drawn using polyline command in AutoCAD considering the script file. To represent the outer wall thickness the offset command is utilized. The selection of the previously drawn polyline is done by using the group name selection option from scripting. Furthermore, the hatch is specified by giving adequate properties of hatch pattern, scale, color, transparency if applicable. Figure 25 below demonstrates the steps followed to achieve the drawing stage of outer walls.

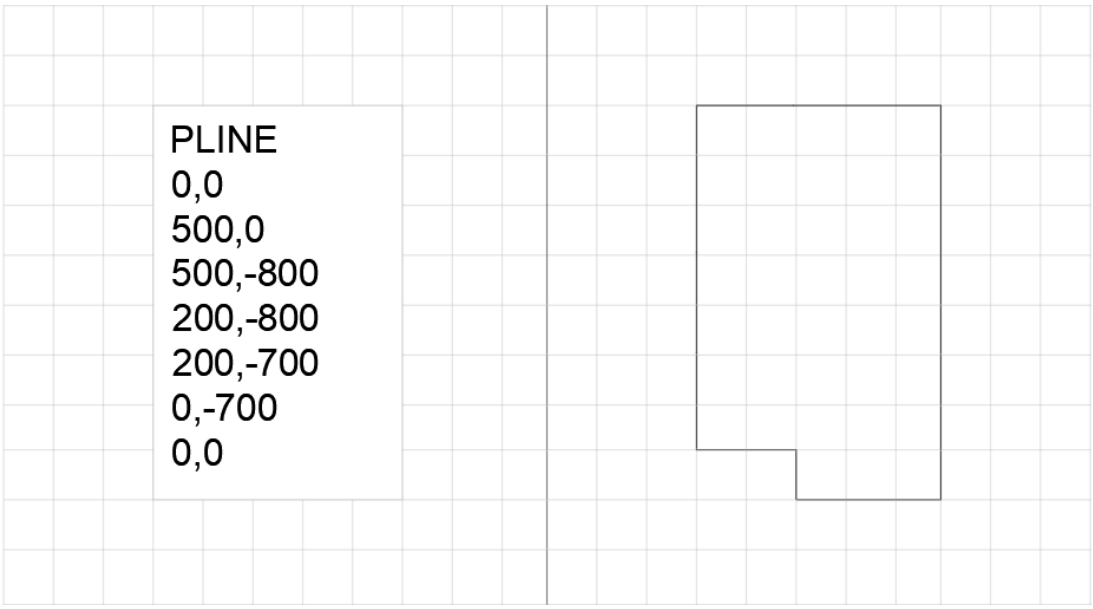


Figure 25: First step of outer wall drawings - reflecting exterior coordinates

In Figure 26 it is shown the drawing polylines using the exterior coordinates for the outer wall. On the left it is demonstrated a few code lines from the scripting file. The second step involves the offsetting of the previously drawn polyline considering the thickness of the outer wall which is given as input by the user. Figure 26 illustrates second step by associating sample of code for the offsetting procedure.

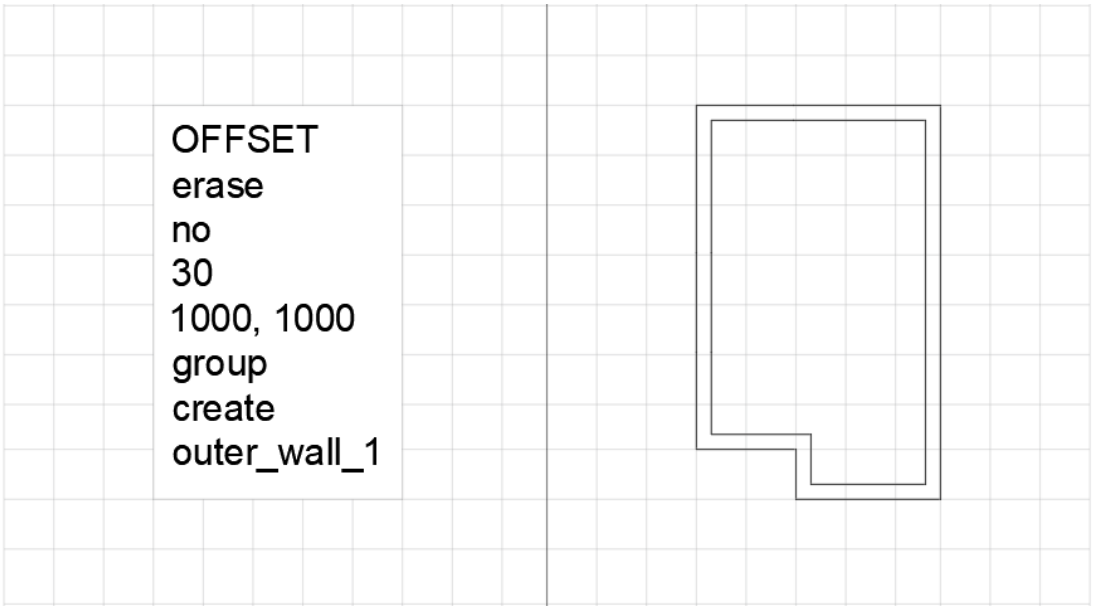


Figure 26: *First step of outer wall drawings - reflecting offset coordinates*

As the offsetting stage is finalized, a coordinate between exterior and interior polyline is specified and used for hatching. A set of parameters as explained previously it is applied if applicable as shown in Figure 27.

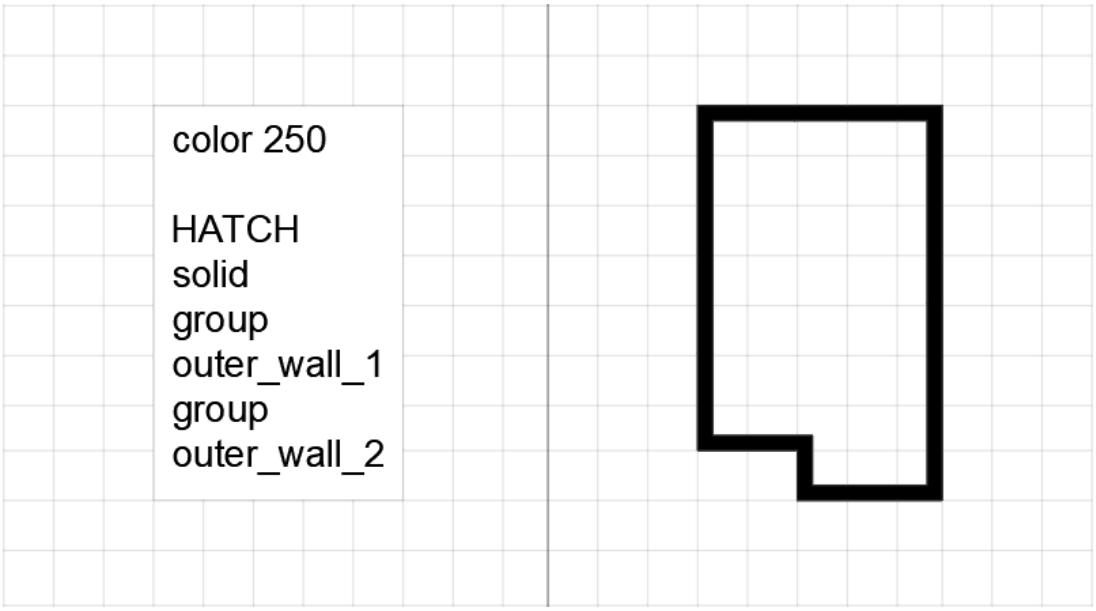


Figure 27: First step of outer wall drawings - reflecting hatching with a specifying pattern

3.6.4 Drawing inner walls

For each in the rooms involved in the plan layout, it is important to determine all the sites that corresponds to inner walls. For instance, the sides that are part of outer perimeter, door must be excluded from the inner walls. This process is ensured by the powerful library named shapely. It offers a wide range of polygon calculations including overlapping of different polygons, area calculation, perimeter calculation as well as sharing coordinates between two polygons. It is exactly the LineString module used to detect either two polygons (in this case rooms) are sharing common walls. The set of criteria is defined for the coordinates of polygons shared and not shared in combination with each other. Therefore, if the room is sharing coordinates with outer perimeter, then this coordinates should not be part of inner walls. In the other hand the common coordinates between two spaces which at the same time are not part of the

outer wall shall be accepted as inner walls. Furthermore, in the third phase these coordinates are represented as polylines, offset by the thickness of the inner walls, filled with the hatch pattern for the same elements and assigned in the respective layer. The figures shown below represent the scenarios mentioned in this section to draw the inner walls.

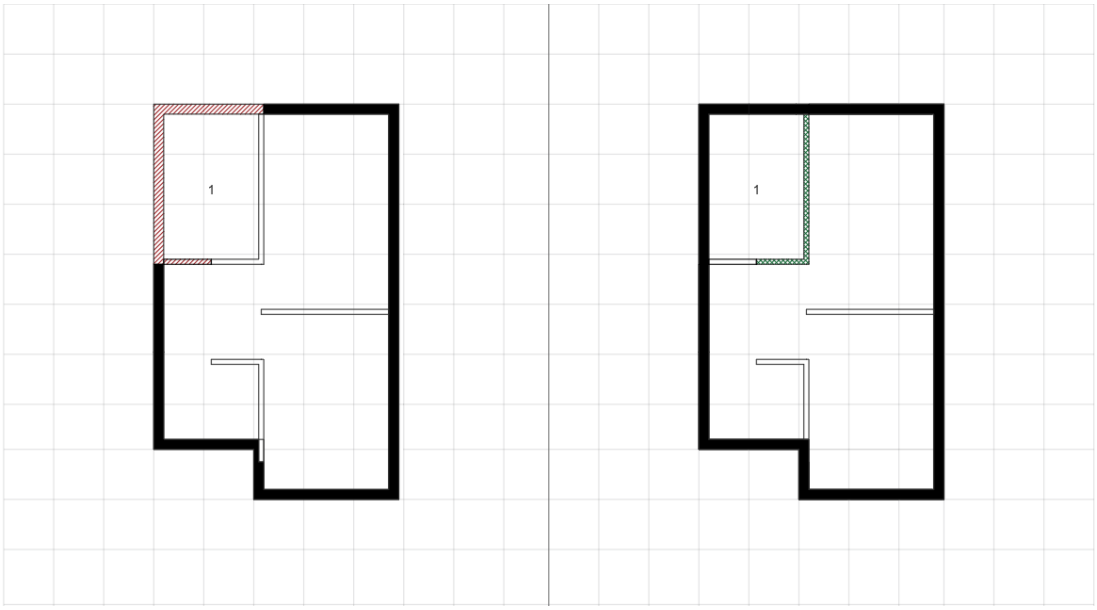


Figure 28: (On the left) example of rejected coordinates provided by python shapely library, (on the right) example of accepted coordinates provided by python shapely library

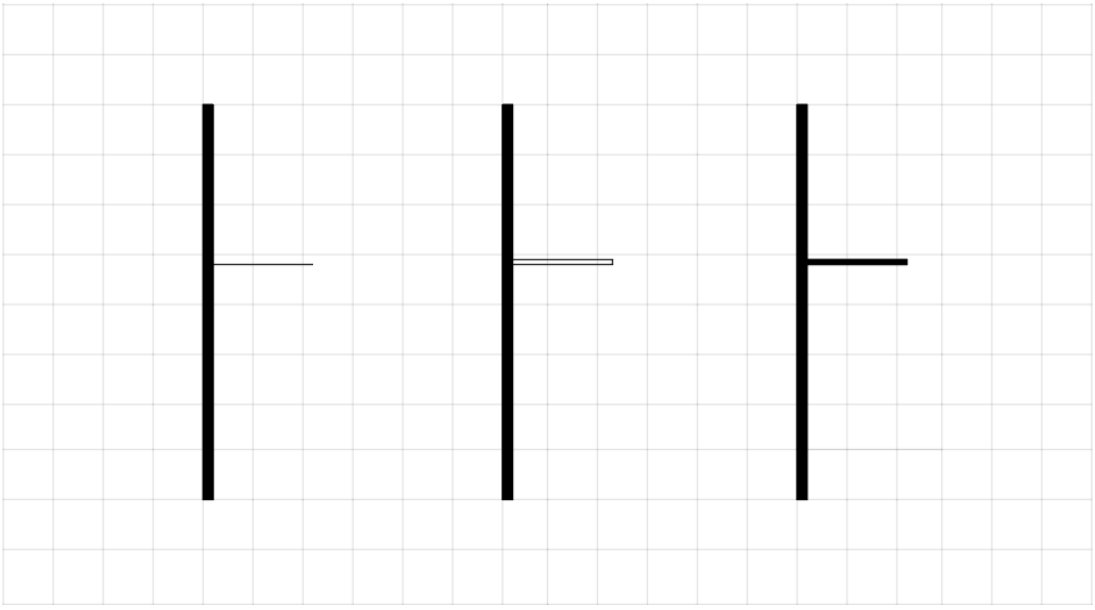


Figure 29: Demonstration of steps followed to draw inner walls: a) drawing the polylines, b) offset of the polylines, c) applying hatch

3.6.5 Drawing windows

To achieve third phase, preparing the scripts for AutoCAD, it requires several drawing tools in coordination with precise dimensions with respective to each room. The methodology followed to draw windows is based on the insertion of blocks. Initially a block is prepared manually representing the window. Afterwards, the code is prepared to calculate the center coordinate of each of the windows allocated in the plan layout. The drawings in the blocks defined before are centered in the 0,0,0 (x,y,z) coordinates to match with the previously calculated window origins. Finally, the script is coded to insert the block to the respective coordinates calculated previously. In the Figure 30 it is shown a representation of calculated coordinates in correspondence with the origin of window block.

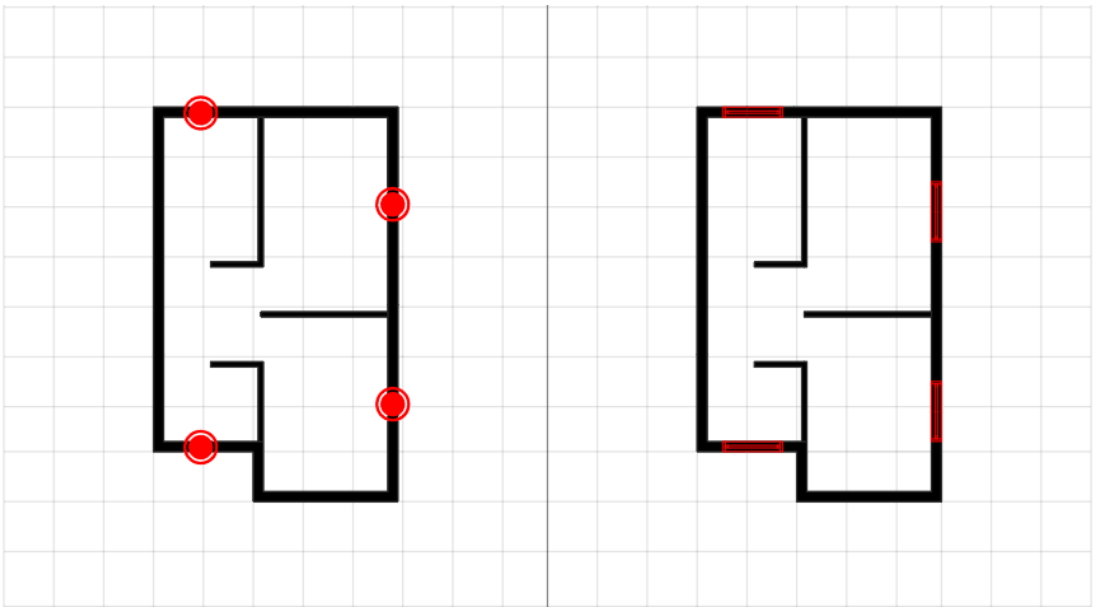


Figure 30: (In the left) representation of calculated coordinates for the arrangement of the window block, (in the right) the insertion of window blocks in the previously window calculations

Once the coordinates are calculated, the blocks are inserted individually at their respective places as represented in the right side of the Figure 30.

3.6.6 Naming the rooms

In order to name the rooms the script is adopted to utilize the single text command in AutoCAD. In this thesis, the text for each of the rooms is located in the center. Each of the room dimensions and coordinates for room corners has been imported in the second phase in python. Considering this data, it is easy to specify the center point for each of the spaces for plan layout. From this stage, the insertion of the text is done by considering the calculation of center coordinates, while other parameters such as text color, size and font are defined in the function. The figure below illustrates the naming of the rooms at their center points.

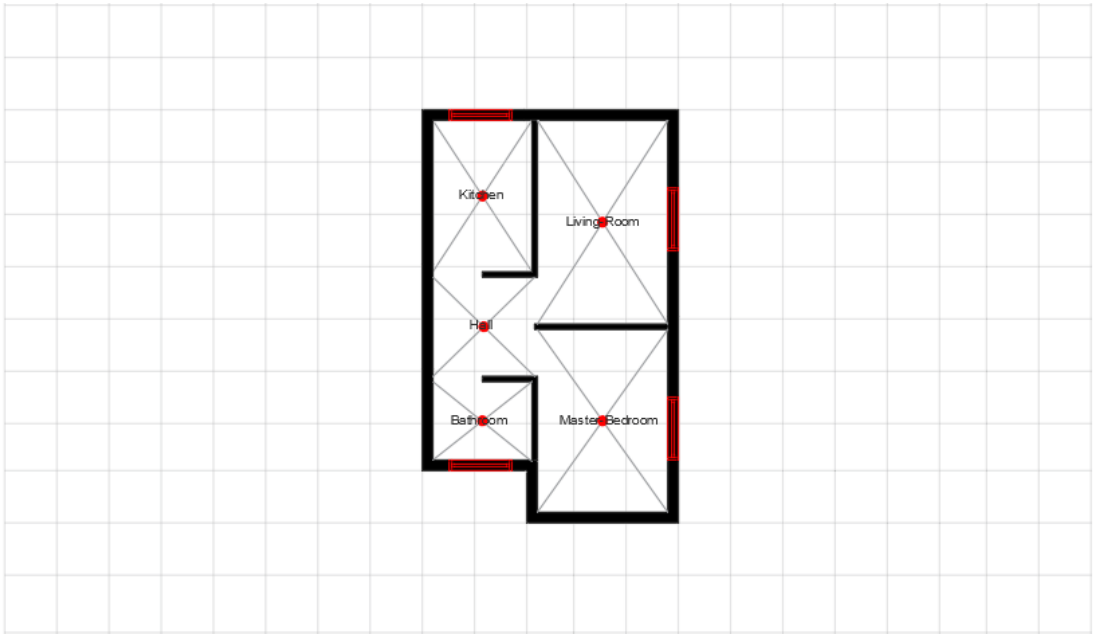


Figure 31: Calculation of center points and allocation of text while naming each of the rooms

3.6.7 Allocating doors and furniture

For the allocation of furniture, initially it is designed a matrix of rules and constraints for each of the rooms in the plan layout. The structure of the matrix is based in the four corners of each room to allocate furniture and doors. From previous data generated in the second phase, the coordinates for the doors are categorized in one of the corners of each room. The algorithm is prepared to automatically avoid the corner assigned to the door and use the other three remaining corners to arrange and orient the furniture involved in the respective unit space. To draw a meaningful plan, the rules set before in the matrix are utilized in this step. For instance, if door is located in the first corner, initially it is crucial not to place any furniture in the same corner and moreover set the ideal corner for the suitable furniture. Figure 32 demonstrates the rules on the left set for the living room in relation with the allocation of the furniture.

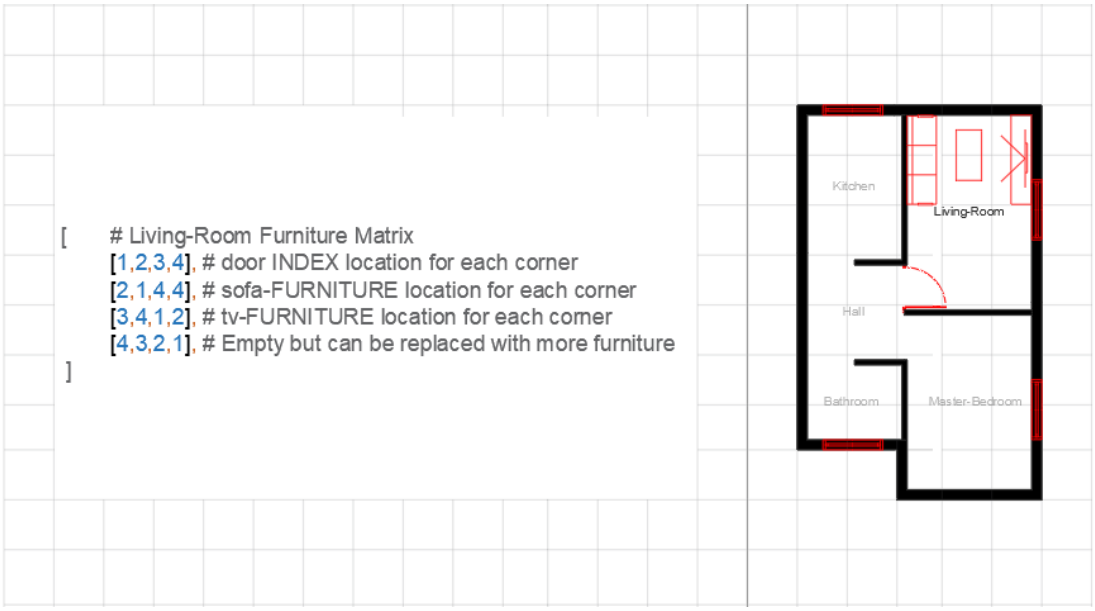


Figure 32: Sample of a part of the matrix used for the definition of rules and constraints for furniture and doors.

As shown, on the left it is presented the set of criteria for the allocation of the furniture and door based on the four corners of the room. The example considered is part of the living room which involves the door itself, the sofa and TV furniture. Focusing on the left criteria, reading the first column from top to bottom can be interpreted as follows: if the door is located at first corner, then sofa must be located at second corner, the TV in the third corner, while the fourth one is left internally empty but can be replaced by any furniture demanded by user. This case is illustrated by the representative example given on the right. Furthermore, reading the second column from top to bottom, consequently is interpreted as follows: if the door is located in the second corner, then the sofa must be located in the firs corner, TV in the fourth one while third corner (opposite to the door) is left empty intentionally. In the same way can be proceeded to interpret the third and the fourth columns, thus full-fling all the possible combinations on one room. In the same trend the matrix is structured for all the other rooms in the plan layout.

The allocation of furniture and door is done based on the corner coordinates following the methodology presented in the section named “drawing windows” which consists in the insertion of previously prepared blocks. To each of the blocks prepared, it is given different attributes as orientation, rotation, scaling and naming. The figure

below illustrates the possible combinations for the door orientation at the first corner.

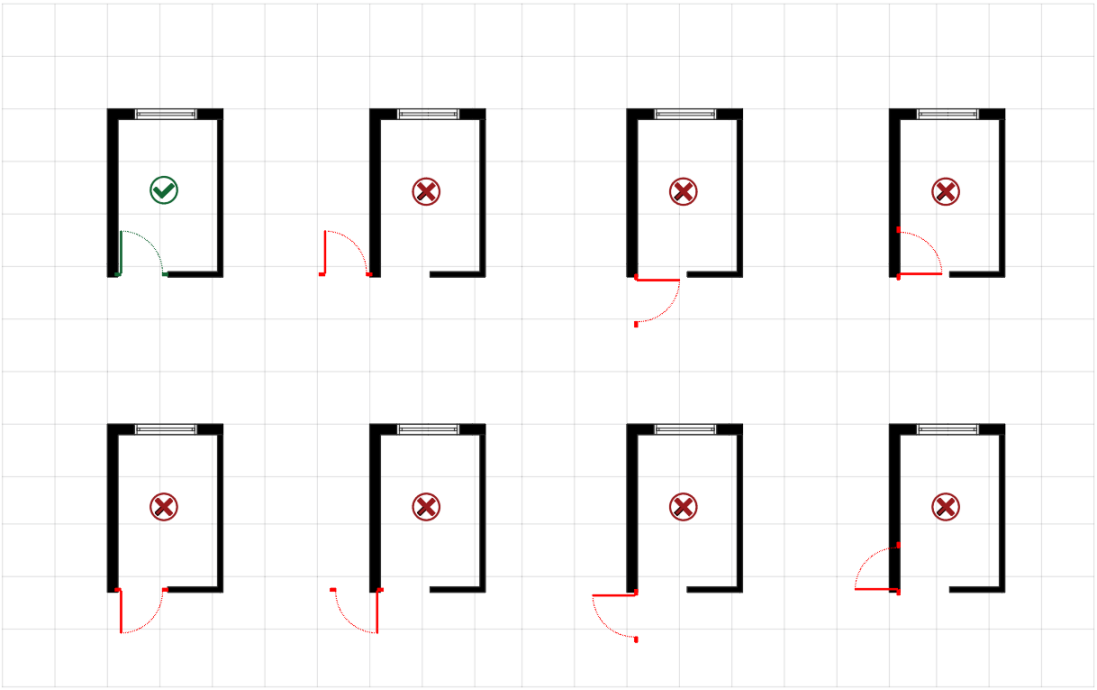


Figure 33: Possible orientations of the door for the first corner in one room

As shown in the Figure 33, out of eight possible scenarios only one can be accepted as the correct implementation. The logic has been implemented in the algorithm prepared to deal with allocation of furniture and doors.

3.6.8 Drawing dimensions

The dimensions are projected to be located in the four sides of the plan layout. This is one phase which clearly shows that the computer solutions sometimes do not provide the efficiency of the professional designers. This is illustrated by the fact that the script will consider sometimes any small dimensions between two different coordinates of the outer wall which could be ignored by the architect or merge with other coordinates to provide one dimension. Nevertheless, the coordinates are calculated based on the coordinates for each of the size of the plan layout with reference to the global centroid. For instance, the coordinates of the left side of the

plan are taking into account for every outer coordinates of the same side considering the range included from top to the bottom part as shown in Figure34.

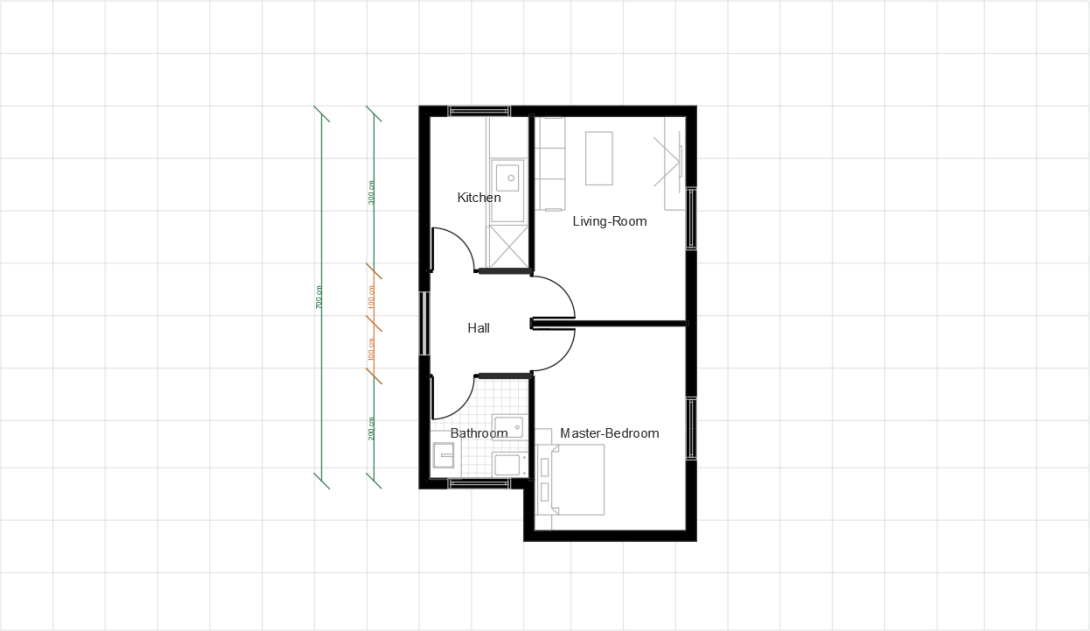


Figure 34: Illustration of the dimensions for the left side for the plan layout.

3.6.9 Preparation of the template

The methodology of importing blocks to outfit the third phase of AutoCAD drawings is implemented also for the preparation of drawing template. It is prepared manually once and saved as a block tool which is than inserted in the centroid of the plan layout and scaled to fit the entire drawing inside. Considering this solution, the architect has the opportunity to chance the template, or specifically some of its detail manually any time and the script will recognize the new updated template for new implementation. It is crucial to emphasize for this study that every block considered for furniture, door or template is saved in .dwg file and can be downloaded in addition to script file from the web-app provided.

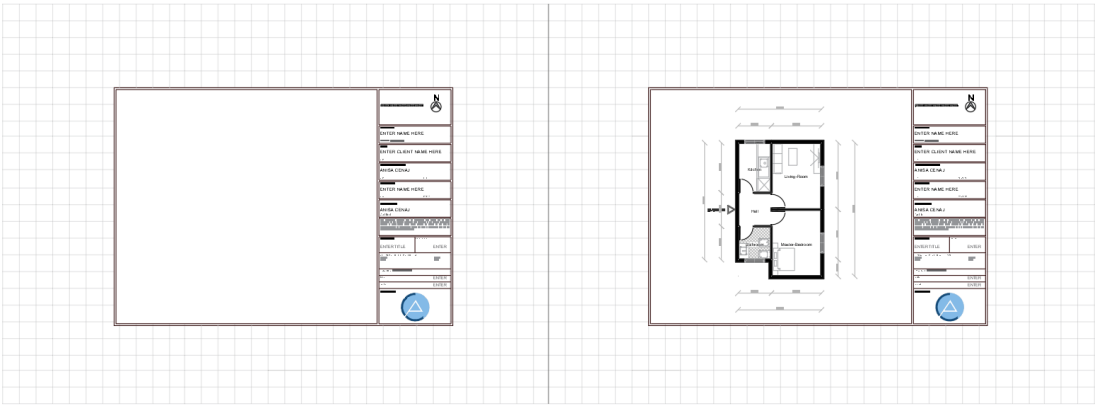


Figure 35: Demonstration of the arrangement of the template related to the plan drawing

3.6.10 Turning on the layers

To prepare the entire script it is a matter of programming skills and adequately sorting all the commands of AutoCAD in the script file. However, due to the huge amount of command lines which will be implemented by AutoCAD macros considering the short delay of time increases the probability of crashes and freezes of the entire command list. Another important issue is the reference point set considering the coordinates calculated by python which can interfere by the other existing objects in the working space of AutoCAD. To avoid this issue, after each of the drawing phases presented in the previous sections, their current layer is turned off. To finalize the scripting procedure, the last step considers the turning on of all layers. In the environment of python programming this is set up using nested loop for each of the layers assigned before, in addition to the commands needed to re-activate these layers.

In the Figure 36 it is shown a summary of the algorithm developed and used for the third phase to achieve the extracting process of plan layouts in AutoCAD software.

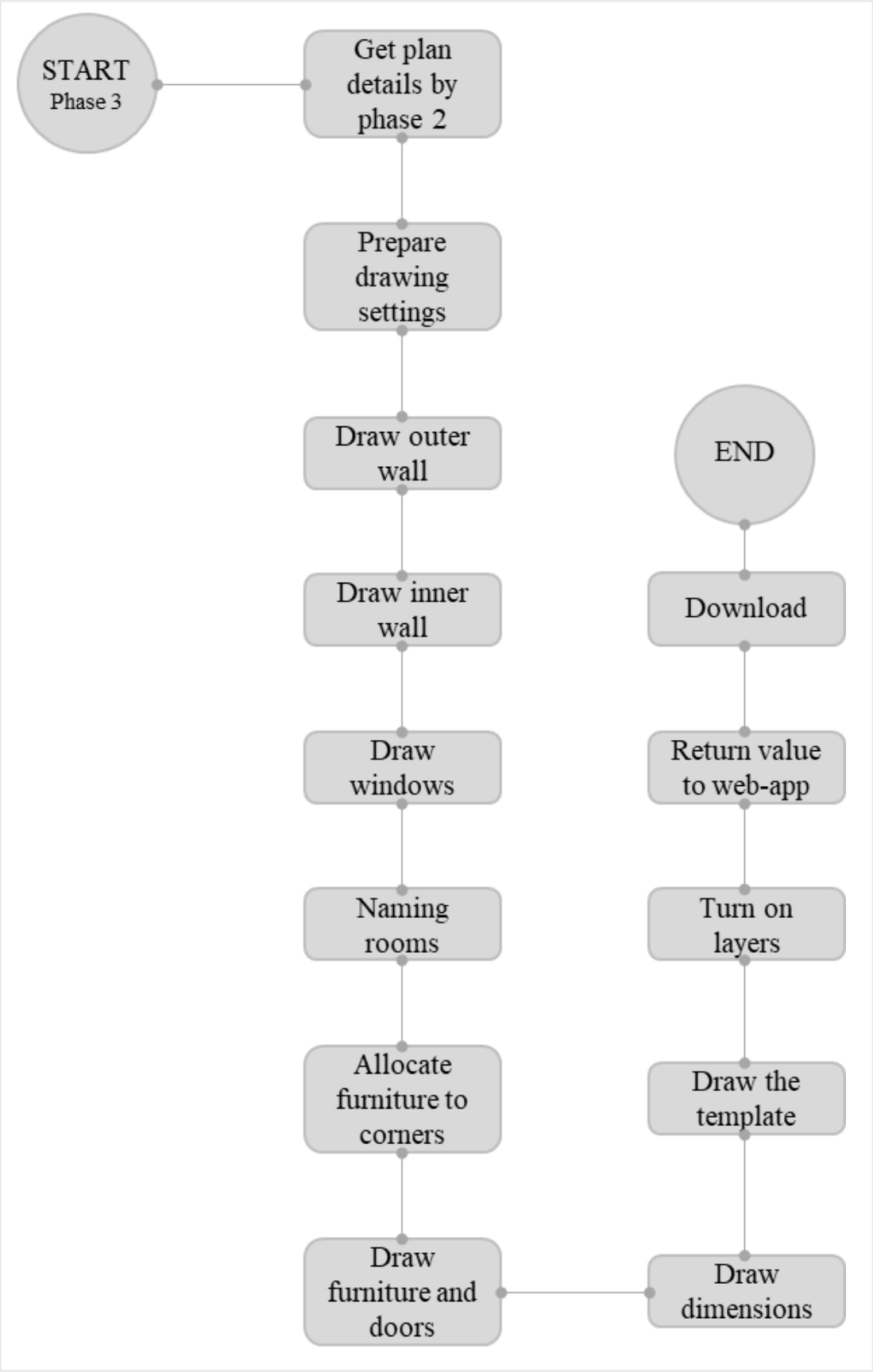


Figure 36: Algorithmic flow chart of the third phase used in plan layout design automation method

CHAPTER 4

ANALYSIS AND RESULTS

4.1 Overview

The implementation of code, constraints and overall automation idea demonstrated in the methodology chapter must be checked in order to confirm either the results are satisfactory or the code needs to improve more. For this purpose a set of analyses are conducted to verify the satisfaction of code outputs. The structure of the analyses it is divided into three groups; the analysis based on the algorithmic development checks, architectural layout checks and computational optimizations as shown in the upcoming sections. Each of the analyses provided is supported by examples taken directly form the code output. Furthermore, general statistics and comparisons are presented in tabular and graphical ways.

4.2 Analyses for algorithm checks

The analyses for the algorithmic checks includes every possible verification of the code, constraint, input implementation in the algorithm. Since the code itself it is divided in three main phases, the analyses are also categorized accordingly.

4.2.1 Verifications regarding first phase

The verification done for the first phase involve a set of trials from the initial user input considering the graphical user interface prepared up to the final raw plan layout generation. Each of the tests is presented as follows:

4.2.1.1 Demonstration of space layout assignment

For the assignment of the space layout, a graphical user interface (GUI) is prepared considering required inputs to allow the entrance of different spaces in a plan layout. The GUI is prepared as web-application aiming to have access flexibility for the utilization. As shown in Figure 37, the app provides various several options to assign the layout spaces. The user can add a room, add multiple versions of specific spaces (e.g. Living room), use pre-generated templates for housing plan typologies (1+1, 2+1, 3+1) and customize different setting options and finally run the analysis.

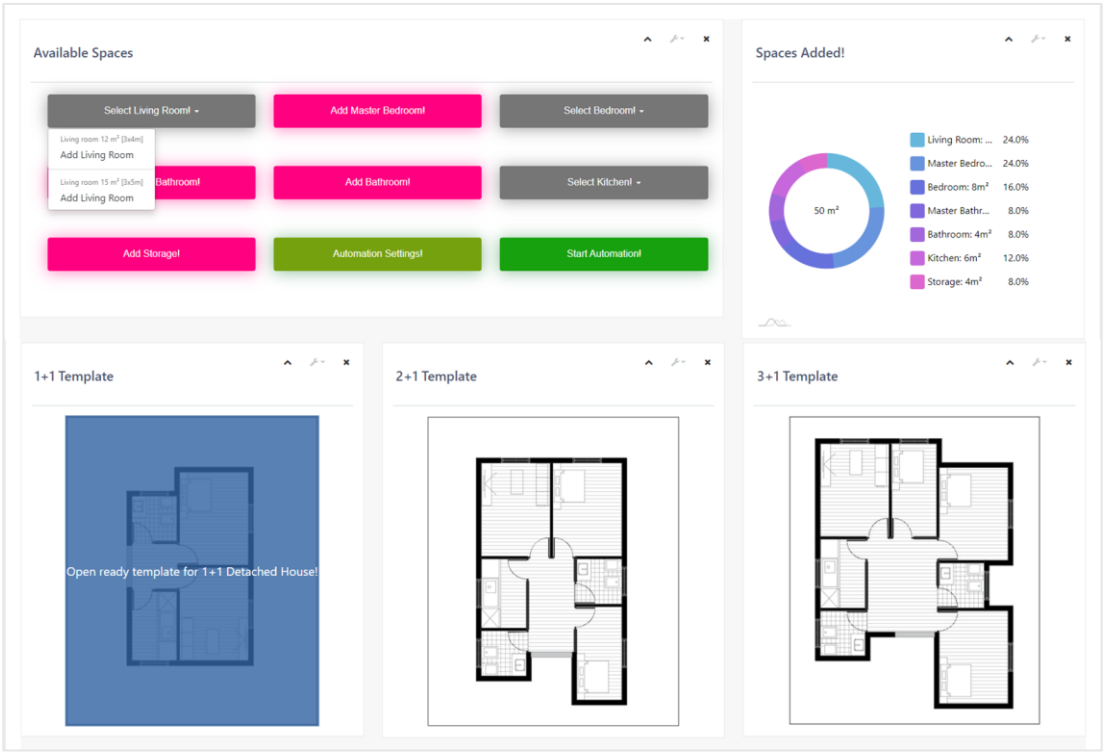


Figure 37: The GUI prepared as web application for user interaction

4.2.1.2 Verification of space orientation

Once several spaces are inputted from GUI, the user can run the analysis. The running process is independent from any interaction and make the entire procedure effortless. Nevertheless, the code must be verified for possible mistakes before

concluding the final outputs. Therefore, in this section it is considered the verification of the space orientations which are distributed based on integrated standards or user preferences in north, south, east or west directions. For testing purposes, the code is modified to assign the direction in a specific orientation, thus concluding either constraints are set properly. As shown in Figure 38 the rooms on the left are set to be placed only in North direction, in the middle in South and West direction and in the right in all compass directions.

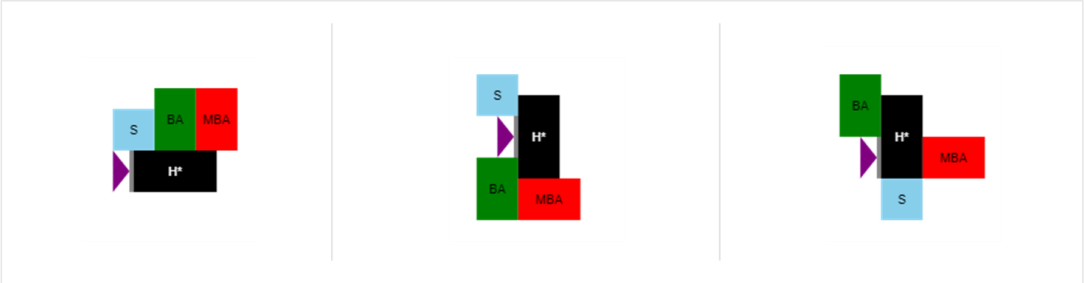


Figure 38: Analyses performed for space orientation based on compass direction

Confirmed also from the results gathered from these analyses, it can be concluded that constraints applied for the space orientation are configured correctly.

4.2.1.3 Verification of path coordinate calculation and corner preventions

The verification of the path coordinates and corner prevention of the first phase is done using a set of rooms while observing each of the possible combination generated. The Figure 39 demonstrates the correct generation of path coordinates for the entrance in west direction. The same analyses are performed for other spaces in different direction as well. In the end it is concluded that the algorithm prepared does a great job in calculating all path coordinates and makes sure no one is skipped. On the other hand, the corner restriction for the rooms is maintained while for the entrance is prevented. Again this indicates a smooth execution by the algorithm of the idea

proposed in this study to be generated. Nevertheless, the combinations must be constrained for space overlapping to provide a correct combination of plan layout. Moreover, Figure 39 illustrates an incorrect location of master bathroom which clashes with master bedroom. In the upcoming section more analyses are run in order to check the new features in the algorithm as regards to these constraints.

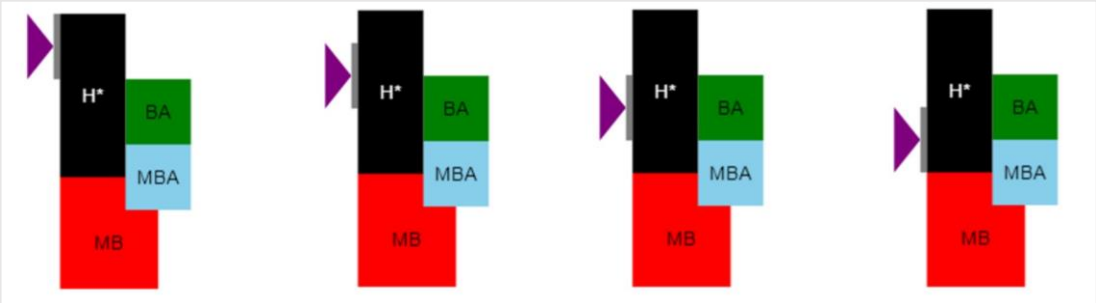


Figure 39: Several location of entrance confirming path coordinates

4.2.1.4 Permutation combinations, overlap prevention and raw plan generation

As illustrated in the previous section, in addition to the path coordinate generation, the room combinations in different plan layouts provide meaningful results from the algorithm. However, the clashes between the different spaces need to be checked. Turning on these constraint, results in less generated plan layouts but filtered from overlapping issues. As shown in Figure 40 the plan layouts not only maintain the corner minimum dimension, ensure the correct coordinate path in the right orientation but also gives satisfactory results that none of the spaces overlaps. Therefore, can be concluded that the algorithm for preventing overlapping works fine thus providing the final version of row plan layouts.

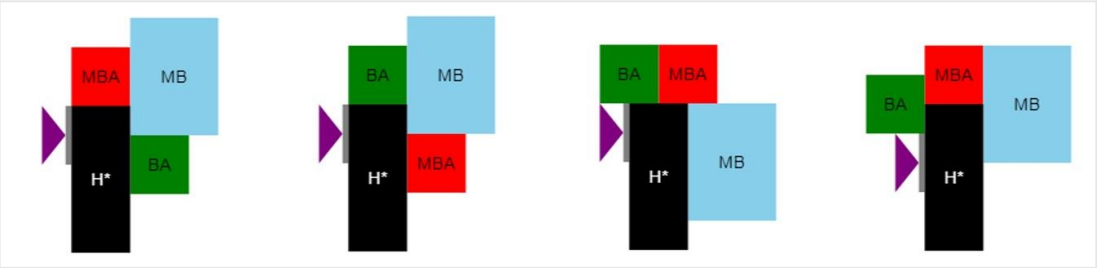


Figure 40: Analysis conducted by the algorithm for the overlap prevention

4.2.2 Verifications done for second phase

The second phase deals with filtering procedures used to the raw plan layouts in regard to the plan validation. The plan validation is represented by the goodness value which is based on three main fitness functions. These fitness functions are selected as; compactness, functionality and parcel occupancy which are explained in more details in addition.

4.2.2.1 Verification of general information

The fitness functions are based on different formulas which includes the perimeter and area of the building, the area of the parcel as well as the validation of the sharing walls between two consecutive spaces. For this reason, it is important to make necessary checks for these parameters before analyzing the other fitness functions. The figure below demonstrates an example of several parameters calculated by the algorithms on the left while on the right there are shown the same outputs by hand calculation.

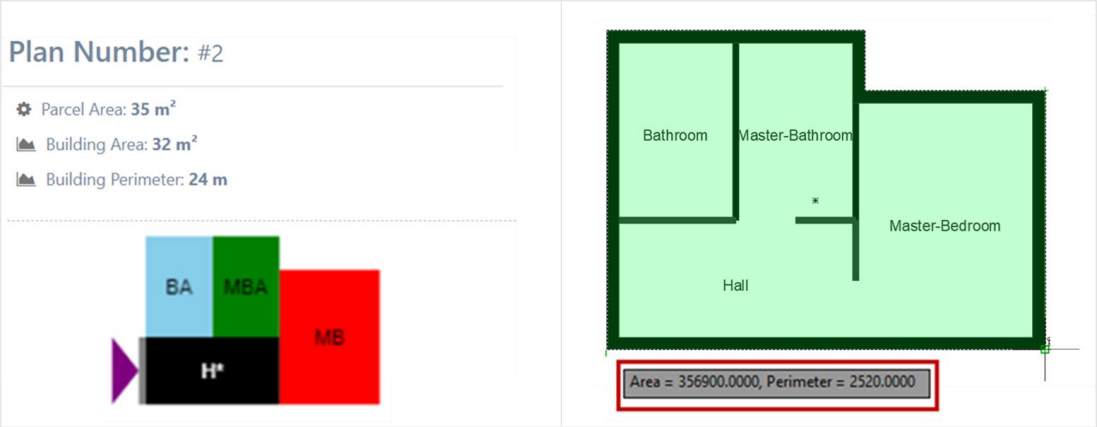


Figure 41: (On the left) parameters calculated by the algorithm, (on the right) the parameters calculated in AutoCAD

Results calculated with hand match exactly the results calculated by the algorithm, which shows a good correlation and proper code implementation.

4.2.2.2 Checks for compactness

The compactness is based in the ratio of perimeter and area of the building. The parameters are confirmed from the previous section. Therefore, the analysis based on the compactness, can be performed by setting different values of this fitness function among the others used. This would be represented by different percentages shown on the graphical user interface prepared for the second phase in the web-application as shown in Figure 42.

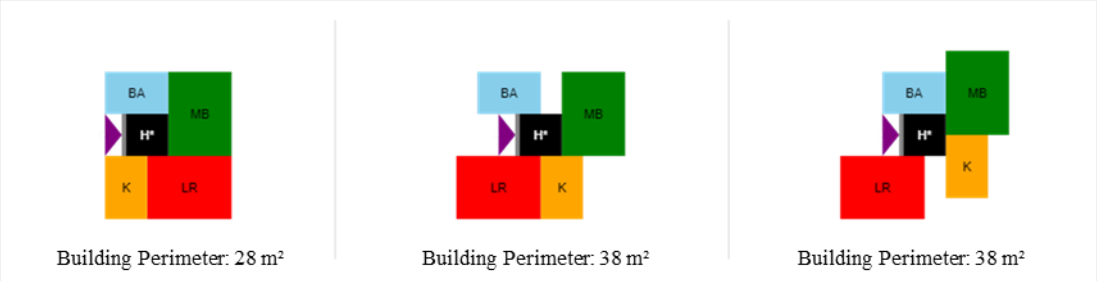


Figure 42: (on the left) a maximum compactness value reached, (in the middle) an average value of compactness reached, (on the right) a low compactness value reached.

The illustration shows that the percentage gives meaningful validation as belongs the compactness value. The plan with more compactness layout has the greater percentage while the other with weakest compactness got less value. This indicates a correct calculation of this parameter.

4.2.2.3 Checks for functionality

Similarly, the checks for functionality are done for the second fitness function. As shown from the Figure 43, depending in the validation criteria setup to the algorithm for the sharing walls, this parameter goes in line with the architectural logic. This also indicates that the calculation of the sharing walls and validation of the functionality fitness value is done properly.

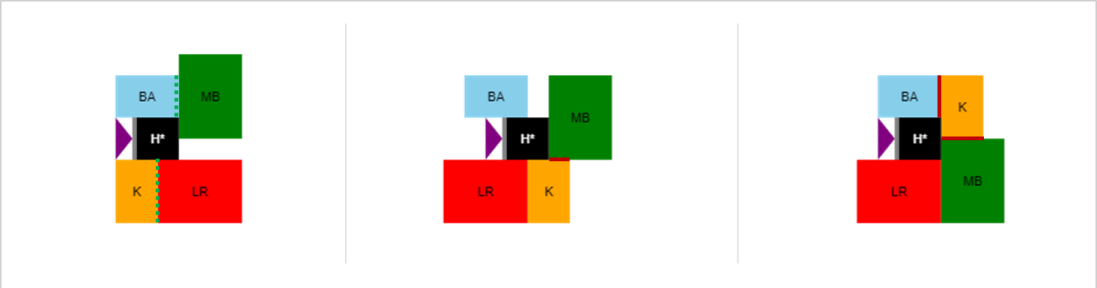


Figure 43: (left) a maximum functionality value reached, (middle) an average value of functionality reached, (right) a low functionality value reached.

4.2.2.4 Checks for parcel occupancy

The parcel occupancy is calculated considering the ratio of the building area versus outer most parcel area covering the external boundaries of the building. Considering this fitness function, the percentage of the parcel occupied is calculated. Figure 44, shows different cases of the parcel occupancy. As shown, the plan which fills the biggest percentage of the external boundaries gets the most percentage indicating the higher priority over the others. This also confirms the correctness of the code as regard to the parcel occupancy fitness value.

Figure 44: (on the left) a maximum parcel occupancy value reached, (in the middle) an average value of parcel occupancy reached, (on the right) a low parcel occupancy value reached.

4.2.3 Verifications done for third phase

The third phase involves the final AutoCAD drawings from one of the plan layouts selected by the user. This process integrates many functions which needs to be checked for possible mistakes or bugs. As the detailed description of each of these functions is given in methodology chapter, an interpretation on the verification of final results is presented in this section.

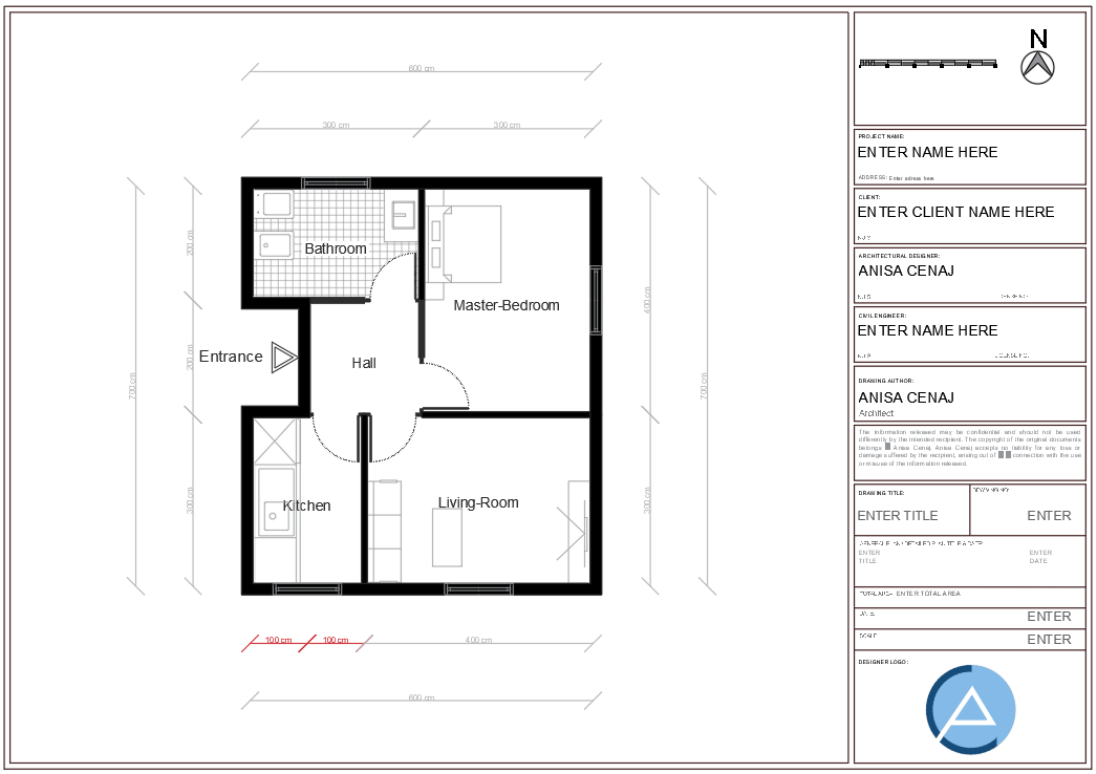


Figure 45: Finalized version of the AutoCAD drawing

Figure 45 shows the finalized version of the AutoCAD drawing for one of the plans selected by the user. As shown, the script does a pretty good work in almost all details. The drawing of outer walls, rooms and inner walls is done properly. In addition, the orientation, scaling, direction and location of furniture and doors seems to be precise and concise. The patterns for hatching, text and layers also provide an idea of human replication as concerns the AutoCAD drawings. The main advantageous of the third phase is that it allows to generate the script for AutoCAD in less than 0.3 seconds and the entire process is effortless. On the other hand, small issues can be detected to the dimensions. Occasionally the script gives additional and unnecessary dimension details in short distances. This can be categorized as one of the limitations of this study which can be improved in further ones.

4.3 Analyses for architectural layout checks

The analyses for the architectural layout checks include adjustments and

calibration of parameters mentioned in the above section which are important for the final output of the architectural plan layouts evaluated by goodness value. This is achieved by alternating several combinations between the fitness functions included in this thesis. Thus the compactness, functionality, parcel occupancy coefficients are adjusted in different scaling to verify in the end the most preferable layout as well as to set a default value which will be considered as calibration for the second phase. The scaling used for the fitness values are grouped into three main scenarios as shown in Table 18.

Table 18: Weight / coefficient of fitness values for different scenarios

	Compacness weight	**Functionallity weight**	**Parcel occupancy weight**
Scenario No. 1	80/10/10	10/80/10	10/10/80
Scenario No. 2	60/20/20	20/60/20	20/20/60
Scenario No. 3	40/30/30	30/40/30	30/30/40

In this way, combining between different scenarios and different fitness weight (coefficient) in total there will be 9 different scenarios. For instance, one scenario could be the first weight of compactness combined with the first weigh of functionality and parcel occupancy (C:80, F:10, P.O:10). Hence, combining between several coefficients while trying for different scenarios it is possible to conclude which among these gives promising results. The conducted analyses and their outputs are presented in the following sections.

4.2.4 Normalization of fitness parameters

As explained in the previous sections, the combinations are applied for the compactness fitness function is considered using the sample of 1+1 house typology from the ready library of the application. Several combinations are compared among them as shown in images below. Initially the comparison for the fitness values for compactness are done using the coefficients 80, 60, 40 as shown in the Figure 46. The

results are tabulated for the top six plans by providing the coefficients for the fitness values as 10, 20, and 30 respectively.

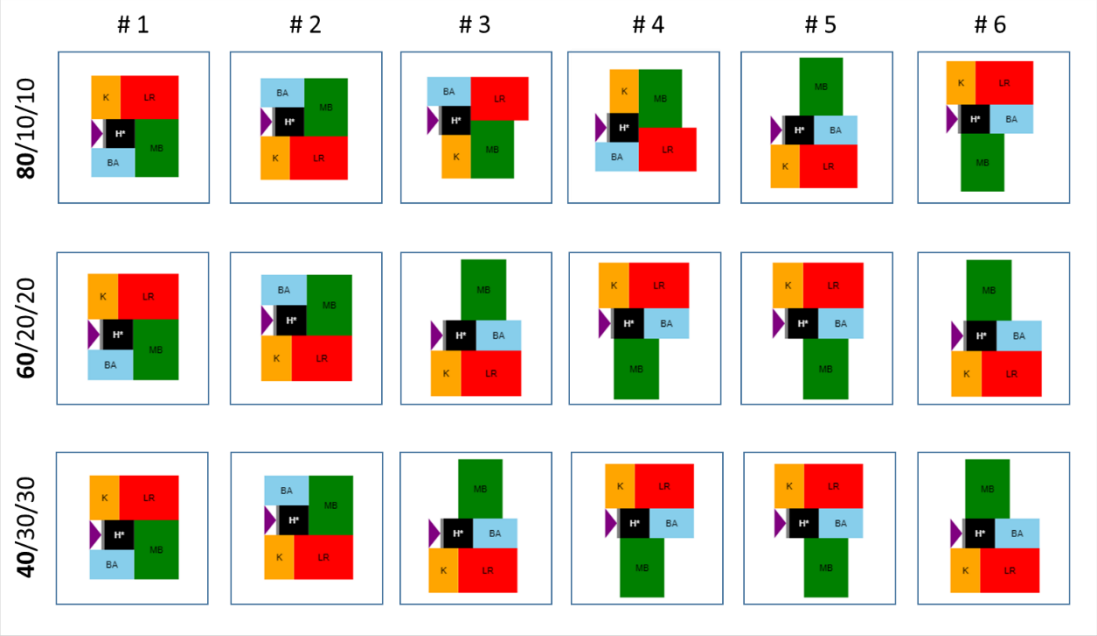


Figure 46: Comparative assessment for the top six plans based on compactness value

As illustrated also by the plans in the Figure 46, while changing the coefficients from 80 to 60 there is a re-evaluation for the last four plans (#3-#6). This indicates a fundamental influence of the compactness coefficient in the validation of the plan layouts. On the other hand, dropping the value from 60 to 40 does not provide any obvious changes for the considered templates. Hence, it can be concluded that for the compactness fitness value the measure coefficients which effect the validation are 80 and 60.

The second parameter used for comparison is the functionality considering the previously described coefficients as shown in Figure 47. Unlike the compactness impact, the trend for the functionality changes in all the categories selected for the validation. As shown in the figure dropping the values from 80 to 60 provides a great change in the plan layouts. Furthermore, comparing the value from 60 to 40 seems to follow the same trend while validating according to functionality fitness value.

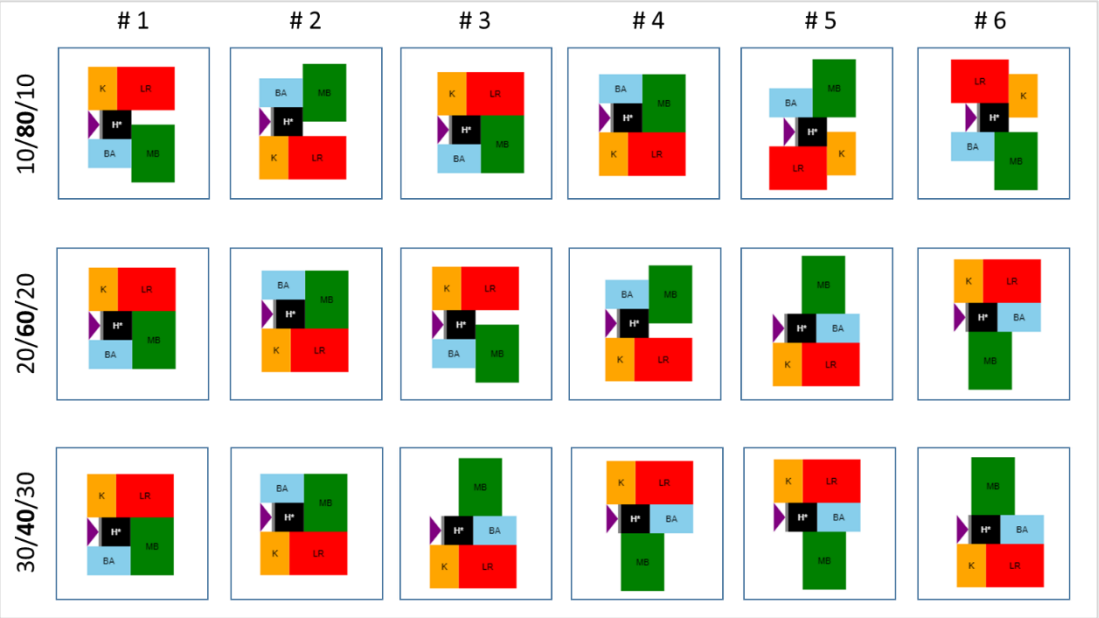


Figure 47: Comparative assessment for the top six plans based on functionality value

Therefore, it can be concluded that each of the coefficients used for the functionality impact directly in the validation of the generated plan layouts.

The analysis are extended also for the parcel occupancy fitness value considering the above mentioned parameters. As shown in Figure 48 the top four plans while comparing 80 to 60 fitness value, are validated with the same values.

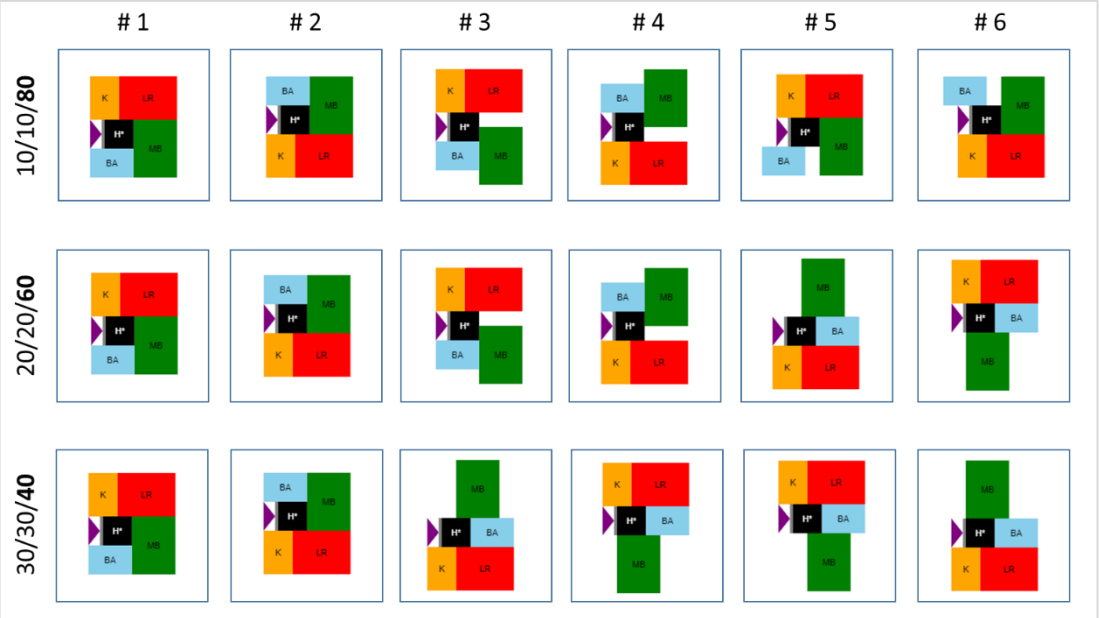


Figure 48: Comparative assessment for the top six plans of parcel occupancy value

The only difference which effects the validation is shown in the fifth and the sixth plan. Nevertheless, unlike the other previous two fitness values, the parcel occupancy seems to have major impact while comparing 60 to 40 coefficients. As shown in the figure, except the top two plans which almost fully occupy the parcel, the other four are validated in a different way compared to the other values. Therefore it can be concluded that for parcel occupancy, the most influential parameter seems to be between 60 and 40.

Another good comparison to suggest the final default values for each of the fitness values included in this thesis, is to show the impact of each of the categories in one graph. As mentioned before, there are considered three combinations for compactness, functionality and parcel occupancy. These combinations are same for all which considers coefficient as in the first group 80, 10, 10, in the second group 60, 20, 20 and in the third group 40, 20, 20. It is very important to emphasize that when the category example 80/10/10 is used, it demonstrates an impact of compactness over the other two fitness values. Therefore, for simplicity to demonstrate in the same graph all the values, labels are used instead of categories. For example C80 represents the category 80/10/10. Furthermore, C40 would represent the category 40/30/30. The same procedure is followed for other fitness parameters. For instance, F60 means the combination 20/60/20 is used. The accumulation of all combinations is presented in the Figure 49.

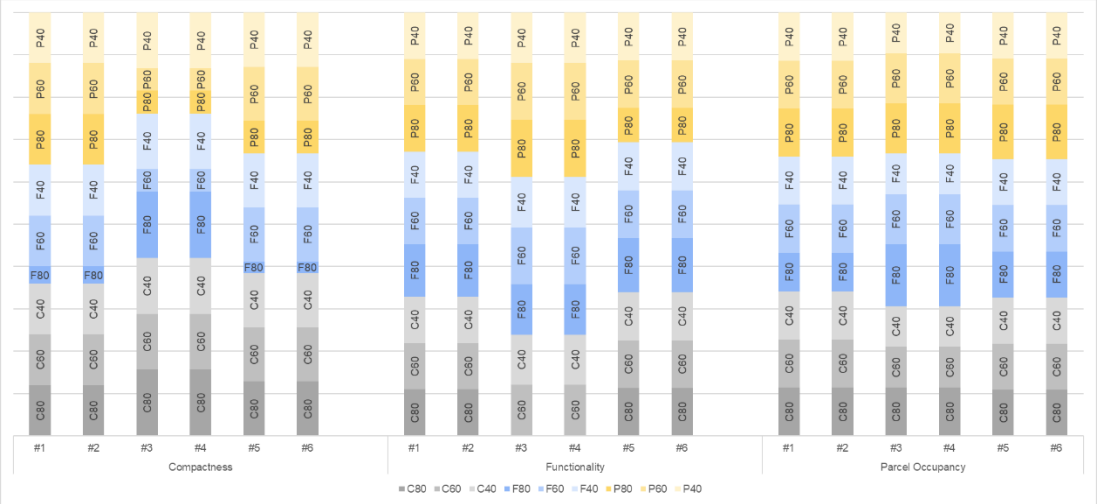


Figure 49: The influence of each fitness value category in compactness, functionality and parcel occupancy

As illustrated in the figure, all nine combinations are demonstrated for compactness, functionality and parcel occupancy. Reading from the bottom of the chart, the combination C80 shows a good performance of compactness in the generated plans, however it gives a major impact in the functionality in plan number #3 and #4. Furthermore, C60 shows good correlation of compactness for all the plans, acceptable functionality but the results in the parcel occupancy are fluctuating for plan number #3 and #4. The same trend is followed in the combination when C40 is used. Nevertheless, comparing these three combinations it seems that the combinations between C60 and C40 are more preferable.

Reading from the chart for functionality fitness parameter, when combination F80 is used, it shows very poor validation of compactness value for plan #1, #2, #5 and #6. The functionality itself is highly evaluated but the same trend as compactness is followed in the parcel occupancy. On the other hand F60 combination influences directly in compactness plan number #3 and #4, gives good results in functionality and acceptable results in parcel occupancy. Once again, when the combination with coefficient 40 (F40) is used, all the fitness parameters seem to validate highly top six plans. This proves that, coefficient 40 is a good indicator for functionality which does not compromise the values of compactness and parcel occupancy.

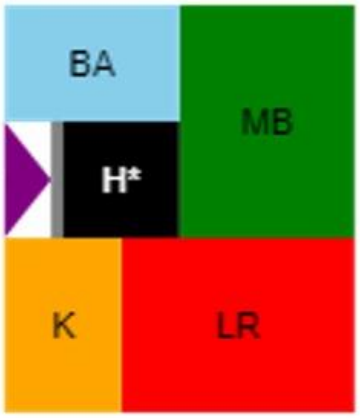
The value 80 used for parcel occupancy (P80), gives low evaluation of plan #3 to #6 for compactness values. Simultaneously, it effects highly plan number #5 and #6 in functionality parameter. Exactly the same behavior, is seem when using the category P60 for all three fitness parameters. However, setting the coefficient the parcel occupancy to 40 gives good results for functionality, acceptable results for compactness and parcel occupancy. Thus, it is once more confirmed that coefficient 40 provides more reliable validation for the automated plan layouts generated by the algorithm.

4.2.5 Customization of adjacency matrix parameters

The adjacency matrix parameters are tabulated and integrated in the algorithm considering the Table 17. The information is gathered from different studies,

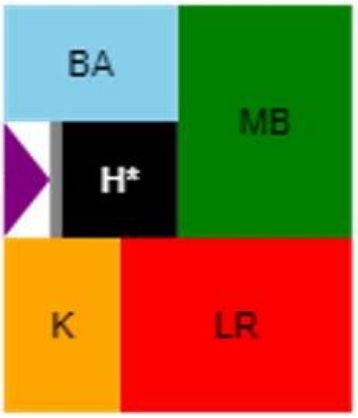
publications and guidelines given in references of this study. Nevertheless, there are different studies suggesting different values for the matrix adjacency matrix which in some cases they do not match with each other. Concluded from all the materials studied for this thesis, the entire process of setting the adjacency matric seems to be subjective. The values integrated in the algorithm are based on the majority and recent publications which seems to provide meaningful validations for the generated plan layouts.

The analyses conducted for the adjacency matrix parameters is based on the validation of functionality fitness value when these parameters are modified. In this way, it can be concluded that the algorithm considers the matrix properly without skipping any of its values. The example shown in Figure 50 corresponds to the plan layout validated two times by modifying the adjacency matrix.



Functionality Value = **88%**

	LR	MB	BA	K
LR	0	**-1**	-1	3
MB	**-1**	0	2	-1
BA	-1	2	0	-1
K	3	-1	-1	1



Functionality Value = **100%**

	LR	MB	BA	K
LR	0	**+1**	-1	3
MB	**+1**	0	2	-1
BA	-1	2	0	-1
K	3	-1	-1	1

Figure 50: The adjacency matrix modification parameters and their influence in the functionality value

The figure illustrates that, the functionality value changes from 88% to 100%

if these parameters are modified accordingly. Therefore, this subjectivity concept can be taken into control if these parameters are modified based on the user demand. In this study the adjacency matrix is integrated in the GUI prepared in the web application.

4.2.6 Calibration of algorithm

The calibration of the algorithm, takes into account a separate additional analysis to verify if any of the plans generated automatically is missed due to any possible coding error. The calibration presented in this study is based on in preparation of one plan manually and then assign the same number of spaces to the algorithm. Therefore, the outcome of the algorithm it is expected to produce a large numbers of plan generations which one of them corresponds with the manually drawn plan. To achieve this step, a plan with a living room, master bedroom and a bathroom is prepared in the framework of third year studio courses at Epoka University as shown in Figure 51 - left and as anticipated the algorithm it generates the same configuration as shown on the right of the Figure 51.

Figure 51: Analysis for calibration purposes. The original plan (on the left), the twin generated plan (on the right)

This proves that as designed in this thesis, the algorithm does not miss any of the plan generation in the automation procedure. Moreover it seems from the generated plan compared with the original one that just a few information are done differently from each other. For example, the inside door location is shifted by the script in other sides for bathroom and living room. Nevertheless, the generation satisfies very good result.

4.4 Analyses for computational checks (CPU)

The algorithm presented in this study is tested by multiple analyses which confirm for its accuracy and result satisfaction. However, the time duration to produce large number of plan layouts during first phase is relatively high. For instance, the overall number of permutation combinations in a $3+1^{+}$ typology is about seventeen trillion. After filtering procedure, the number of combinations reduces to sixty nine thousand. For this reason, it is understandable that the computational cost matters a lot in such studies. In this thesis, the duration of the algorithm is highly improved by parallelizing the execution of the code in several CPUs. This is done using multiprocessing library which considers the number of CPU cores of the computer [106].

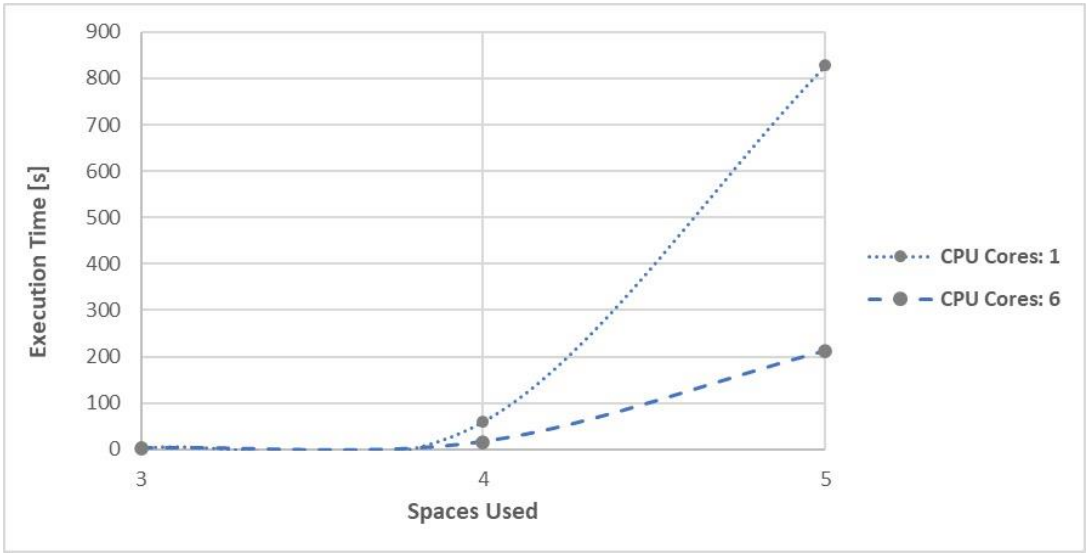


Figure 52: The improvement in execution time for scenarios with 1 CPU core (on the left) and 6 CPU cores (on the right) from the same device

From the analyses conducted, it results that the parallelization of the CPUs impacts highly the duration of the algorithm. For example, thirteen hour running code in standard unparalleled code with six physical cores device would be improved more than fourteen times (fifteen minutes) when parallelized fully. In addition, the same code executed in another device with 32 physical cores, takes less than six minutes. Hence, the acceleration of the running time is massively improved by this method. In the Figure 52 the statistics gathered from multiple analysis parallelized and unparalleled are shown for a 6 physical core CPU device. As seen from the figure, the parallelization does not show a very good job when considering few spaces but it improves tremendously the execution time when the number of spaces is increased.

CHAPTER 5

DISCUSSION AND CONCLUSION

5.1 Overview

In this study, several analysis are conducted for each of the phases and steps structured as algorithm to prove that the code is implemented in a right way. Furthermore, analyses are extended to validate the generated plan layouts according to the architectural design. The final outcomes are summarized in the previous chapter and concluded as follows:

- In the first phase, all the checks are in line with the structure designed for the algorithm and the final output as raw plan layout is successfully filtered from the constraints defined in the code.
- The fitness parameters included in this study, are calculated for each of the automated plan, remapped and then used in the goodness value formula during the second phase. The massive number of analysis conducted proves that the second phase, including the calculation of fitness parameters as well as the validation of each plan layout is done adequately.
- Moreover, a graphical user interphase (GUI) for the adjustment of fitness parameters is provided in this thesis and configured in the code for further user implementations.
- The second phase provides the generated plan in simplified shapes for an immediate user interaction which is supported additionally by the third phase.
- In the third phase the transferring of generated plan data is done to AutoCAD software considering detailed elements such as inner wall, outer wall, furniture, window, inner doors, entrance, dimensions, text, hatch, layers and drawing template.
- While considering the combination of nine plan categories in the same

chart, it is concluded that the most favorable combinations are when coefficients 40, 30, 30 are used. Moreover, considering the importance of the functionality in a detached house plan layout, in this thesis by default as well as suggested values for the fitness parameters used are considered 30 for compactness, 40 for functionality and 30 for parcel occupancy. These values are also suggested for other studies, however for the user satisfaction, the web application provided in the framework of this thesis provides a dynamic interaction, allowing professional designers to modify this values.

- The adjacency matrix is prepared based on recent publications and studies and presented in the graphical user interface (GUI). From the analysis performed, it is concluded that the matrix is integrated properly in the code. As the entire process of parameter selection in the matrix is subjective, the user is provided with the editable version of these parameters and free access to change according to their preferences.
- Due to the large number of generated plan layouts verifying either the code is set up properly or not, in this study it is presented a calibration considering a previously existing or draft drawn plan layout. The results shows significant matching between the existing and automated plan.
- The core part of this study is to provide an automation procedure for detached house in short amount of time. Therefore, the duration of code execution has a higher impact while utilizing this method. The parallelization of CPU cores presented in this study plays a vital role in shortening the overall execution time.

5.2 The study innovation

A deep investigation on previous studies is presented in the literature review considering the automation of plan generations. Different researchers have provided various techniques, constraints and methods in the generation of layouts using different tools [107, 61, 108, 51, 49, 26, 73]. The innovation of this study is based on

a different technique, methods tools and structures which are explained as follows:

- The proposed methodology for the realization of the entire automation process is based on the "Centrum" method. This method takes into account the hall/corridor as the main unit and a stationary point for all the other rooms.
- One of the main disadvantages of the automation procedure confirmed in literature review is the architect's ability to code and the duration of the algorithm execution [65, 62, 66]. This study uses Python as a suitable and practical programming language for beginners and implements one of the most sophisticated techniques based on the CPU parallelization to improve highly the execution time of massive generated layouts.
- Another contribution for the country and architect's society is the implementation of Albanian local standards in the algorithm.
- Furthermore, the local standards are not to be seen as obstacles as the product provided in this thesis allows the user to easily adopt any other norms.
- In literature, there are proposed a lot of additional tools to be integrated for the conversion of the data to regular AutoCAD drawing plans [71, 60, 65, 66, 68, 70, 69]. In this thesis a unique procedure is implemented to achieve this step utilizing the AutoCAD scripting method. It provides not only a direct integration in one programming language, but immediate and fast drawings directly in AutoCAD software utilizing all its features.
- In literature there are presented a lot of techniques which require the user interaction from downloadable software [59, 13, 51, 107]. This requires the user to download the software, adopt to the operation system of the device and repeat the same process in case of updates or upgrades. In this thesis for the first time, the output and user interaction it is integrated in one central platform which operates in cloud and can be accessed through a web application prepared. This ensures that any update or upgrade can be reflected in the real time without interrupting

the surface. Each user may have access through the log in pages considering the demo or full package provided by the developer.

- The main advantage of using this web application is that it is compactable with any operating system, any device either it is desktop, tablet or mobile running on windows, mac, android or iOS.
- Finally, the execution of the code can be performed by a central super computer established by a developer, so the user gets the most complicated results in a very short amount of time.

5.3 Limitations

As any study, the proposed methodology in the current thesis has its own limitations. Considering all the phases proposed and developed in the previous chapters, the main limitations are described as follows:

- The inputted shapes which the algorithm understands are based on the regular rectangular shapes.
- The algorithm does not provide a plan generation for balconies, loggias or other rooms that are not directly connected to the stationary point (example: master bathroom inside the master bedroom).

5.4 Future work

The improvements to the method provided in this study can be done in different ways and scales. Undoubtedly, the further developments and establishment of the entire code in cloud as well as preparing the customer packages would be a great business idea for the presented method. In addition, possible other improvements for the future work are shown as follows:

- Providing a more extensive study taking into account the addition of the

floor number and the integration of stairs in the building.

- Providing a similar study for residential apartments maybe a very interesting and good contribution for not only our country and architects but also in literature.
- The rectangular shape spaces maybe changed by more complicated geometrical designs to reach a different level of architectural layout.
- Furthermore, the hall can be extended to more flexible shape to be adopted better in the generated plan layout.
- A 3D modelling phase can be generated by AutoCAD scripting for drawing and rendering purposes.
- A constructive project must be prepared before the implementation in site, therefore, fundamental checks for the structural elements may be a good future work.
- The analyses can be extended by involving more fitness parameters as the influence of light in the interior spaces, thermal impact and acoustical coverage.

REFERENCES

[1] Chi-Han Peng, Yong-Liang Yang, Peter Wonka, "Computing Layouts with Deformable Templates," in *ACM Transactions on Graphics*, New York, NY, United States, ACM SIGGRAPH, 2014, pp. 1-11.

[2] Victor Calixto, Gabriela Celani, "A literature review for space planning optimization using an evolutionary algorithm approach: 1992-2014," in *XIX Congresso da Sociedade Ibero-americana de Gráfica Digital 2015*, 2015.

[3] R. Olcayto, "Eisenman: computers dumb down design," BdOnline, 16 May 2008. [Online]. Available: https://www.bdonline.co.uk/eisenman-computers-dumb-down-design/3113566.article. [Accessed December 2022].

[4] INSTAT, "Lejet e ndërtimit," Instituti i statistikave, Tiranë, 2022.

[5] Aliaj Sh., Koçiu S., Muco B., Sulstarova E., Seismicity, seismotectonics and seismic hazard assessment in albania, Tiranë: Akademia e shkencave e , 2010.

[6] Freddi F., Novelli V. R., Gentile R., Veliu E., Andreev S., Andonov A., Greco F., Zhuleku, "Observations from the 26th November 2019 Albania earthquake: the earthquake," *Bulletin of Earthquake*, vol. 19, pp. 2013-2044, 2021.

[7] E. Fasoulaki, "Integrated Design A Generative Multi-Performative Design Approach," *Massachusetts Institute of Technology*, 2008.

[8] S. Wolfram, "Cellular automata as models of complexity," vol. 311, no. 5985, pp. 419-424, 4 October 1984.

[9] Chad Adams, Hirav Parekh, Sushil J. Louis, "Procedural Level Design using an Interactive Cellular Automata," in *GECCO '17 Companion*, Berlin, Germany, 2017.

[10] O.K. Tonguz, Wantanee Viriyasitavat, Fan Bai, "Modeling Urban Traffic: A Cellular Automata Approach," *Communications Magazine, IEEE*, vol. 47(5), pp. 142 - 150, 2009.

[11] Jokar Arsanjani, Jamal and Helbich, Marco and Ali Mousivand, "A Morphological Approach to Predicting Urban Expansion," *Transactions in GIS*, vol. 18, 2013.

[12] Eduardo Valente, Camelia Avram, José Machado, Adina Astilean, "An Innovative Approach for Modelling Urban Road Traffic Using Timed Automata and Formal Methods," *Journal of Advanced Transportation*, vol. 2018, 2018.

[13] Anthony G. O., Yeh, Xia Li, Chang Xia , "Cellular Automata Modeling for Urban and Regional Planning," in *Urban Informatics*, Singapore, Springer Singapore, 2021, p. 865–883.

[14] Jianxin Yang, Wenwu Tang, Jian Gong, Rui Shi, Minrui Zheng, Yunzhe Dai, "Simulating urban expansion using cellular automata model with spatiotemporally explicit representation of urban demand," *Landscape and Urban Planning,* vol. 231, p. 104640, 2023.

[15] Amir Mahmood Ghafari, Muhammad Zaly shah, Omidreza Saadatian, Elias Salleh, "Cellular automata in urban planning and development," *Journal Design + Built,* vol. 5, 2012.

[16] Yan Liu, Michael Batty, Siqin Wang, Jonathan Corcoran, "Modelling urban change with cellular automata: Contemporary issues and future research directions," *Progress in Human Geography,* vol. 45, pp. 3-24, 2021.

[17] M. Hansmeyer, "Computational Architecture: L-Systems," [Online]. [Accessed December 2022].

[18] C. Jacob, "Evolving Evolution Programs: Genetic Programming and L-Systems," *Genetic Programming, ,* pp. 107-115, 1996.

[19] M. Hansmeyer, "Computatuonal Architecture," [Online]. Available: https://www.michael-hansmeyer.com/l-systems. [Accessed 2023].

[20] Ruoxi Sun, Jinyuan Jia, Marc Jaeger, "Intelligent Tree Modeling Based on L-system," 2009.

[21] Mariatul Kiptiah binti Ariffin, Shiqah Hadi, Somnuk Phon-Amnuaisuk, "Evolving 3D Models Using Interactive Genetic Algorithms and L-Systems," in *International Workshop on Multi-disciplinary Trends in Artificial Intelligence*, 2017.

[22] H. Gautier, R. Měch, P. Prusinkiewicz, C. Varlet-Grancher, "Architectural Modelling of Aerial Photomorphogenesis in White Clover (Trifolium repens L.) using L-systems," *Annals of Botany,* vol. 85, pp. 359-370, 2000.

[23] Peter Wonka, Michael Wimmer, rançois Sillion, William Ribarsky, "Instant Architecture," in *Association for Computing Machinery*, New York, NY, USA, 2013.

[24] Fasoulaki, E, "Integrated Design A Generative Multi-Performative Design Approach.," *Massachusetts Institute of Technology.,* 2008.

[25] Mine Özkar, Sotirios Kotsopoulos, "Introduction to shape grammars," *CM SIGGRAPH 2008 Classes,* p. 36, 2008.

[26] Ana Belčič, Sara Eloy , "Architecture for Community-Based Ageing—A Shape Grammar for Transforming Typical Single-Family Houses into Older People’s Cohousing in Slovenia," *Buildings,* vol. 13, 2023.

[27] J. Reis, "Shapes: Seeing and Doing with Shape Grammars," in *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*, Madrid, Spain, 2022, pp. 1-5.

[28] Jonathan Dessi-Olive, Timothy Hsu , "A Simulation-Validated Shape Grammar for Architectural Acoustics," *Nexus Network Journal,* vol. 24, pp. 55-73, 2022.

[29] "Surface Shape Grammar Morphology to Optimize Daylighting in Mixed-Use Building Skin," *ASCAAD 2021,* pp. 479-492, 2021.

[30] Jan Halatsch, Antje Kunze, Gerhard Schmitt, "Using Shape Grammars for Master Planning," in *Design Computing and Cognition*, Netherlands, Springer Netherlands, 2008, pp. 655--673.

[31] Hau Hing Chau, Xiaojuan Chen, Alison McKay, Alan de Pennington , "Evaluation of a 3D Shape Grammar Implementation," in *Design Computing and Cognition '04*, Springer Netherlands, 2004, pp. 357-376.

[32] M. A. Dias, "Informal Settlements: A Shape Grammar Approach," *Journal of Civil Engineering and Architecture,* vol. 8, pp. 1389-1395, 2014.

[33] G Stiny, W J Mitchell, "The grammar of paradise: on the generation of Mughul," School of Architecture and Urban Planning, University of California, Los Angeles, California, 2008.

[34] H. H. Chau, "Preserving brand identity in engineering design using a grammatical approach," *University of Leeds,* 2002.

[35] Andrew N. Sloss, Steven Gustafson , "2019 Evolutionary Algorithms Review," in *Genetic Programming Theory and Practice XVII*, Springer, Cham, 2020, p. 307–344.

[36] Maciej Nisztuk, Paweł B. Myszkowski, "Hybrid Evolutionary Algorithm applied to Automated Floor Plan Generation," *International Journal of Architectural Computing,* vol. 17, pp. 260-283, 2019.

[37] Vincent J.L. Gan, H.K. Wong, K.T. Tse, Jack C.P. Cheng, Irene M.C. Lo, C.M. Chan, "Simulation-based evolutionary optimization for energy-efficient layout plan design of high-rise residential buildings," *Journal of Cleaner Production,* vol. 321, pp. 1375-1388, 2019.

[38] Morteza Rahbar, Mohammadjavad Mahdavinejad, Amir H.D. Markazi, Mohammadreza Bemanian, "Architectural layout design through deep learning and agent-based modeling: A hybrid approach," *Journal of Building Engineering,* vol. 47, p. 103822, 2022.

[39] L. Caldas, "Generation of energy-efficient architecture solutions applying GENE_ARCH: An evolution-based generative design system," *Advanced Engineering Informatics,* vol. 22, pp. 59-70, 2008.

[40] Katarzyna Grzesiak-Kopec, Barbara Strug, Grazyna Slusarczyk, "Evolutionary Methods in House Floor Plan Design," *MPDI,* no. 8229, p. 14, 2021.

[41] Payman, "Swarm Intelligence," Jet Propulsion Laboratory, Pasadena, 2004.

[42] Eric Bonabeu, Marco Dorigo, Guy Theraulaz, "Swarm Intelligence - From Natural to Artificial Systems," in *Santa Fe Institute Studies in the Sciences of Complexity*, Oxford University Press, 1999.

[43] Yuichiro Yoshida, Hooman Farzaneh, "Optimal Design of a Stand-Alone Residential Hybrid Microgrid System for Enhancing Renewable Energy Deployment in Japan," *Energies 2020,* vol. 13, p. 18, 2020.

[44] Marco Dorigo, Vittorio Maniezzo, Alberto Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics),* vol. 26, pp. 29-41, 1996.

[45] Rob Roggema, Popov Nikolay, "Swarm Planning: Development of Generative Spatial Planning Tool for Resilient Cities," *Towards Smarter Cities,* vol. 1, pp. 519-527, 2015.

[46] A. Agirbas, "Façade form-finding with swarm intelligence," *Automation in Construction,* vol. 99, pp. 140-151, 2019.

[47] P. Polinceusz, "Structure of architecture--tensegrities in the construction of architectural space.," *Architecture, Civil Engineering, Environment,* vol. 12, no. 1, p. 45, 2019.

[48] Tran Q, Do V, Dinh T, "Traffic signal timing optimization for isolated urban intersections considering environmental problems and non-motorized vehicles by using constrained optimization solutions.," *Innovative Infrastructure Solutions,* vol. 7(5), 2022.

[49] Abbas Babazadeh, Hossain Poorzahedy, Saeid Nikoosokhan, "Application of particle swarm optimization to transportation network design problem," *Journal of King Saud University - Science,* vol. 23, pp. 293-300, 2011.

[50] Helin POLAT, Zeynep Yeşim İLERİSOY, "A Geometric Method on Facade Form Design with Voronoi Diagram," *Research Article,* vol. 3, no. 2, pp. 179 - 194, 2020.

[51] A. Abbas, "Voronoi Diagram Applications Towards New Sustainable Architectural Language," *Journal of Engineering Research,* vol. 6, no. 4, 2022.

[52] Yijun Lu,Wei Wu,Xuechuan Geng,Yanchen Liu,Hao Zheng, Miaomiao Hou , " Multi-Objective Optimization of Building Environmental Performance: An Integrated Parametric Design Method Based on Machine Learning Approaches.," *Energies,* vol. 15, 2022.

[53] Giulia Angelucci, Fabrizio Mollaioli, "Voronoi-Like Grid Systems for Tall Buildings," *Frontiers in Built Environment,* vol. 4, pp. 1-20, 2018.

[54] Kathleen M. Carley, Michael J. Prietula, "Social Science Computer Review," in *Computational Organization Theory,* Psychology Press, 1994, pp. 611-624.

[55] Michael W. Macy, Robert Willer, "From Factors to Actors: Computational Sociology and Agent-Based Modeling," *Annual Review of Sociology,* vol. 28, pp. 143-166, 2002.

[56] H. Yi, "Visualized Co-Simulation of Adaptive Human Behavior and Dynamic Building Performance: An Agent-Based Model (ABM) and Artificial Intelligence (AI) Approach for Smart Architectural Design," in *Sustainability*, Korea, 2020.

[57] John c. Kunz, Raymond e. Levitt, Yan Jin, "The Virtual Design Team: A Computational Simulation Model of Project Organizations," *Communications of the Association for Computing Machinery*, vol. 41, pp. 84-92, 1998.

[58] Gregory J Smith, Mary Lou Maher, John S Gero, "Designing 3D Virtual Worlds as a Society of Agents," in *Key Centre of Design Computing and Cognition*, Sydney, 2003.

[59] Christoph Sydora, Eleni Stroulia, "Rule-based compliance and generative design for building interiors using BIM," *Automation in Construction*, vol. 120, p. 103368, 2020.

[60] Xiao-Yu Wanga, Yin Yang, Kang Zhang, "Customization and generation of floor plans based on graph transformations," *Automation in Construction*, vol. 94, p. 405–416, 2018.

[61] Graziella Laignel, Nicolas Pozin, Xavier Geffrier, Loukas Delevaux, Florian Brun, Bastien Dolla, "Floor plan generation through a mixed constraint programming-genetic," *ELSEVIER ScienceDirect*, 2021.

[62] Machi Zawidzki, Kazuyoshi Tateyama, Ikuko Nishikawa, "The constraints satisfaction problem approach in the design of an architectural functional layout," in *Engineering Optimization*, England, Taylor & Francis, 2011, p. 943–966.

[63] Jeremy Michalek, Ruchi Choudhary, Panos Papalambros, "Architectural layout design optimization," in *Engineering Optimization*, England, Taylor & Francis, 2002, pp. 461-484.

[64] Xuejun Cao, Zhijun He, Yunhe Pan, "Automated design of house-floor layout with distributed planning," *Computer-Aided Design*, vol. 22, pp. 213-222, 1990.

[65] Paul Merrell, Eric Schkufza, Vladlen Koltun, "Computer-Generated Residential Building Layouts," *ACM Transactions on Graphics*, p. 13, December 2010.

[66] Wamiq Para, Paul Guerrero, Tom Kelly, Leonidas Guibas, Peter Wonka, "Generative Layout Modeling using Constraint Graphs," in *IEEE Xplore*, 2020.

[67] SIU-PAN LI, JOHN H. FRAZER, MING-XI TANG, "A CONSTRAINT BASED GENERATIVE SYSTEM FOR FLOOR LAYOUTS," *CAADRIA proceedings*, pp. 417-426, 2000.

[68] Nitant Upasani, Krishnendra Shekhawat, Garv Sachdeva, "Automated generation of dimensioned rectangular floorplans," *Automation in Construction*, vol. 113, p. 103149, 2020.

[69] Wenming Wu, Xiao-Ming Fu, Rui Tang, Yuhan Wang, Yu-Hao Qi, Ligang Liu, "Data-driven interior plan generation for residential buildings," *ACM Transactions on Graphics,* vol. 38, pp. 234:1-234:12, 2019.

[70] Wenming Wu, Lubin Fan, Ligang Liu, Peter Wonka, "MIQP-based Layout Design for Building Interiors," *Computer Graphics Forum,* vol. 37, pp. 511-521, 2018.

[71] Graziella Laignel, Nicolas Pozin, Xavier Geffrier, Loukas Delevaux, Florian Brun, Bastien Dolla, "Floor plan generation through a mixed constraint programming-genetic optimization approach," *Automation in Construction,* vol. 123, p. 103491, 2021.

[72] P. Svanerudh, "Architectural constraints for design automation of multi-storey titn'iber houses," *Automation and Robotics in Construction,* pp. 181-186, 1999.

[73] P. Charman, "A constraint-based approach for the generation of floor plans," in *Proceedings Sixth International Conference on Tools with Artificial Intelligence*, France, 1994, pp. 555-561.

[74] Jingyu Zhang, Nianxiong Liu, Shanshan Wang, "Generative design and performance optimization of residential buildings based on parametric algorithm," *Energy & Buildings,* vol. 224, p. 111033, 2021.

[75] Romualdas Bausys, Ina Pankrašovaite, "Optimization of architectural layout by the improved genetic algorithm," *Journal of civil engineering and management,* vol. 11, pp. 13-21, 2005.

[76] B. Zyrtare, Fletorja Zyrtare e Republikës së Shqipërise, Tiranë: Botim i Qendrës së Botimeve Zyrtare, 2015.

[77] E. Hasu, "Governing domestic space: Townhouserelated living, gardens and the homemaking process in Finland.," in *Proceedings of the 6th Annual Architectural Research Symposium* , Finland, 2014.

[78] Nigel Oseland, Ian Donald, "The Evaluation of Space in Homes: A Facet Case Study," *Journal of Environmental Psychology,* vol. 13, pp. 251 - 261, 1993.

[79] Helena Monteiro, Fausto Freire, Nelson Soares, "Life cycle assessment of a south European house addressing building design options for orientation, window sizing and building shape," *Journal of Building Engineering,* vol. 39, p. 102276, 2021.

[80] Nicolas Pardo, Christian Thiel, "Evaluation of several measures to improve the energy efficiency and CO2 emission in the European single-family houses," *Energy and Buildings,* vol. 49, pp. 619-630, 2012.

[81] Helena Monteiro, Nelson Soares, "Integrated life cycle assessment of a southern European house addressing different design, construction solutions, operational patterns, and heating systems," *Energy Reports,* vol. 8, pp. 526-532, 2022.

[82] Fletorja Zyrtare e Republikës së Shqipërise, Tiranë: Botim i Qendrës së Botimeve Zyrtare, 2015.

[83] Brijesh Mainali, Krushna Mahapatra, Georgios Pardalis, "Strategies for deep renovation market of detached houses," *Renewable and Sustainable Energy Reviews,* vol. 138, p. 110659, 2021.

[84] G. v. Rossum, "Python tutorial, Technical Report CS-R9526," Centrum voor Wiskunde en Informatica (CWI), Amsterdam, 1995.

[85] Y. v. Havre, "Python for Architects," 2012. [Online].

[86] time, "time — Time access and conversions¶," Python Software Foundation, 2021. [Online]. Available: https://docs.python.org/3/library/time.html. [Accessed 2023].

[87] S. Gillies, "The Shapely User Manual," Python Software Foundation , 2023. [Online]. Available: https://shapely.readthedocs.io/en/stable/manual.html. [Accessed 2023].

[88] importlib, "importlib — The implementation of import," Python Software Foundation, 2022. [Online]. Available: https://docs.python.org/3/library/importlib.html.

[89] tkinter, "tkinter — Python interface to Tcl/Tk," Python Software Foundation, 2021. [Online]. Available: https://docs.python.org/3/library/tkinter.html. [Accessed 2022].

[90] Python, "datetime — Basic date and time types," Python Software Foundation, 2022. [Online]. Available: https://docs.python.org/3/library/datetime.html. [Accessed 2022].

[91] itertools, "itertools — Functions creating iterators for efficient looping¶," Python Software Foundation, 2021. [Online]. Available: https://docs.python.org/3/library/itertools.html. [Accessed 2023].

[92] Igor Mujan, Aleksandar S. Anđelković, Vladimir Munćan, Miroslav Kljajić, Dragan Ružić, "Influence of indoor environmental quality on human health and productivity - A review," *Journal of Cleaner Production*, vol. 217, pp. 646-657, 2019.

[93] N. Gohardani, "Architecture and design research: Reflections in relation to the design process," *Archnet-IJAR,* vol. 5, 2011.

[94] Wenwen Li , Michael F. Goodchild & Richard Church, " An efficient measure of compactness for two-dimensional shapes and its application in regionalization problems," *International Journal of Geographical Information Science,* p. 25, 2013.

[95] Frolov, "Measuring shape of geographical phenomena - history of issues.," Soviet Geography Review and Translation, 1975.

[96] Miller, "A quantitative geomorphic study of the drainage basin characteristics in the Clinch Mountain area, Virginia and Tennessee," Columbia University Department, New York, 1953.

[97] L. Richardson, "A note: measuring compactness as a requirement of legislative apportionment," *Mid-west Journal of Political Science,* vol. 5, pp. 70-74, 1961.

[98] Santiago, R.S., Bribiesca, E., "State of the art of compactness and circularity measures," *International Mathematical Forum,* vol. 4, 2009.

[99] Zhao, Z.Q., Stough, R.R., "Measuring similarity among various shapes based on geometric matching.," *Geographical Analysis,* vol. 37, 2005.

[100] R. Osserman, "Isoperimetric inequality," *Bulletin of the American Mathematical Society,* vol. 84, p. 1182–1238., 1978.

[101] Eugénio Rodrigues, David Sousa-Rodrigues, Mafalda Teixeira de Sampayo , Adélio Rodrigues Gaspar, Álvaro Gomes, Carlos Henggeler Antunes , "Clustering of architectural floor plans: A comparison of shape representations," *Automation in Construction,* vol. 80, pp. 48-65, 2017.

[102] Feng Shi, Ranjith K. Soman, Ji Han, Jennifer K. Whyte, "Addressing adjacency constraints in rectangular floor plans using Monte-Carlo Tree Search," *Automation in Construction,* vol. 115, p. 103187, 2020.

[103] Javid Ahmadi,Seyyed Mehdi Maddahi, Reza Mirzaei, "Generative Design of Housing Spatial Layout Based on Rectangular Spaces," *Advances in Civil Engineering,* vol. 2023, 2023.

[104] Ying-CHun Hsu, RObert J. Krawczyk, "Space adjacency behavior in space planning," in *CAADRIA*, 2004.

[105] K. Shekhawat, "Automated space allocation using mathematical techniques," *Ain Shams Engineering Journal,* vol. 6, no. 3, pp. 795-802, 2015.

[106] P. S. Foundation, "multiprocessing — Process-based parallelism," Python, 2022. [Online]. Available: https://docs.python.org/3/library/multiprocessing.html.

[107] Bahraminejad, F. and Babaki, K., "Application Of Voronoi Diagram As An Architectural And Urban Planning Design Tool'," *Indian Journal of Fundamental and Applied Life Sciences,* pp. 1776-1783, 2014.

[108] P. Rubinowicz, "Chaos and Geometric Order in Architecture and Design," *Journal for Geometry and Graphics,* vol. 4, no. 2000, pp. 197-207, 2000.

APPENDIX

More information about the web application of the centrum automation procedure can be found below:

www.anisacenaj.com/centrum-gd