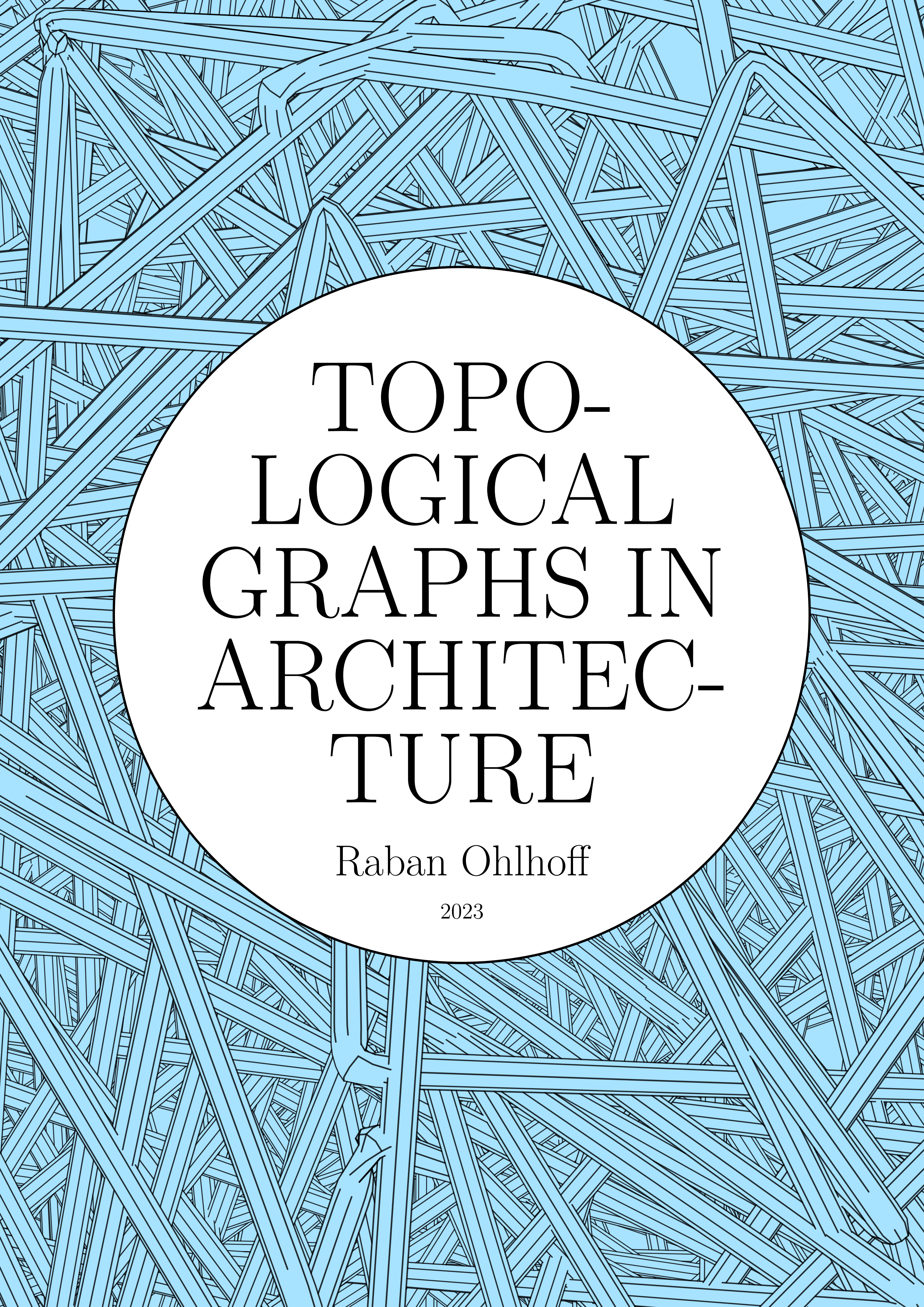


TOPO-LOGICAL GRAPHS IN ARCHITEC-TURE

Raban Ohlhoff

2023

August 2023
Faculty of Architecture La Cambre Horta
Université Libre de Bruxelles

Thesis Supervisor : David Erkan
Second Supervisor : Gian Marco Paldino
Internal Jury Member : Gianluca Bontempi

Contents

It is the fact of space that creates the special relation between function and social meaning in buildings. The ordering of space in buildings is really about the ordering of relations between people. Because this is so, society enters into the very nature and form of buildings. They are social objects through their very form as objects. Architecture is not a 'social art' simply because buildings are important visual symbols of society, but also because, through the ways in which buildings, individually and collectively, create and order space, we are able to recognise society: that it exists and has a certain form. — Hillier and Hanson 1989, *The social logic of space*

Abstract

Traditional architectural practices often rely on creative intuition and experience rather than systematic analysis and data-driven decision making. With the increasing availability of computational tools and data science techniques, there is an opportunity to bring mathematical and computer science concepts and methods into the architectural context to challenge traditional practices and offer potential improvements. This thesis explores the application of graph theoretical and topological concepts in architecture and investigates the use of graph machine learning methods in the context of architectural analysis, with a particular focus on energy efficiency as a key performance metric. To this end, a synthetic architectural dataset containing geometric, categorical, dimensional, energetic, topological and relational information is generated by integrating various space partitioning algorithms combined with architectural control functions into an automated generation pipeline. Subsequently, a classification model and a regression model are trained on the generated knowledge graph dataset to evaluate the prediction and classification accuracy in terms of energy efficiency. The resulting dataset and the code for generating and training the model will be made publicly available to further research in the field of graph machine learning in architectural applications. This research demonstrates the potential of a closer integration of various mathematical concepts and computer science methods into the architectural design and verification process, and shows the potential of applying knowledge graphs for the abstraction, representation and analysis of architectural objects.

Keywords — Architecture, Graph Theory, Topology, Machine learning, Simulation

Resumé

Les pratiques architecturales traditionnelles s'appuient souvent sur l'intuition créative et l'expérience plutôt que sur l'analyse systématique et la prise de décision fondée sur des données. Avec la disponibilité croissante d'outils informatiques et de techniques de science des données, il est possible d'introduire des concepts et des méthodes mathématiques et informatiques dans le contexte de l'architecture afin de remettre en question les pratiques traditionnelles et d'offrir des améliorations potentielles. Cette thèse explore l'application de la théorie des graphes et des concepts topologiques à l'architecture et étudie l'utilisation des méthodes d'apprentissage automatique des graphes dans le contexte de l'analyse architecturale, avec un accent particulier sur l'efficacité énergétique en tant que mesure clé de la performance. À cette fin, un ensemble de données architecturales synthétiques contenant des informations géométriques, catégorielles, dimensionnelles, énergétiques, topologiques et relationnelles est généré en intégrant divers algorithmes de partitionnement de l'espace combinés à des fonctions de contrôle architectural dans un système de génération automatisé. Ensuite, un modèle de classification et un modèle de régression sont entraînés sur l'ensemble de données graphique de connaissances généré afin d'évaluer la précision de la prédiction et de la classification en termes du rendement énergétique. L'ensemble de données résultant et le code de génération et d'entraînement du modèle seront mis à la disposition du public pour faire avancer la recherche dans le domaine de l'apprentissage automatique des graphes dans les applications architecturales. Cette recherche démontre le potentiel d'une intégration plus étroite de divers concepts mathématiques et de méthodes informatiques dans le processus de conception et de vérification architecturales, et montre le potentiel de l'application de graphes de connaissances pour l'abstraction, la représentation et l'analyse d'objets architecturaux.

Mots-clés — Architecture, Théorie des graphes, Topologie, Apprentissage automatique, Simulation

I would like to express my sincere gratitude to all those who have contributed to the completion of this thesis.

First and foremost, I would like to thank my supervisor, David Erkan, for providing invaluable advice, support and feedback throughout the research process. Your expertise, patience and willingness to challenge my ideas have been instrumental in the creation of this thesis.

I would also like to give special thanks to Gian Marco Paldino, Professor Wassim Jabi, Professor Gianluca Bontempi and all the participants who generously gave their time, insights and experience to this thesis. Their contributions have helped me to achieve my research goals.

I am grateful to my family and friends for their unwavering encouragement, understanding and motivation. Their love and support have carried me through the ups and downs of this journey.

Finally, I would like to acknowledge the contribution of this institution and the faculty members who have constantly challenged me throughout my academic journey. Thank you for providing me with the knowledge, skills and resources that have enabled me to undertake this research.

Without the support of these individuals and institutions, this thesis would not have been possible. Thank you.

List of Figures

List of Tables

Glossary

10-fold cross-validation Evaluation metric of a machine learning model through division of the training data into 10 subsets, training on 9 of them and testing on the remaining one. After repetition of the process, the final performance is the average of all 10 performance values. 96, 99, 102

activation Function that determines the output of an ANN layer based on the input combined with their corresponding weights. 93, 95

aperture An opening, such as a window or door, which connects two mediums with each other. 32, 82–84, 87, 109, 111, 131

axial map Graphical representation of spatial relationships. 22, 34, 36, 51

bayesian optimisation Technique for finding the global optimum of a function through construction of a probabilistic model. 48, 96

binary tree Hierarchical data structure with at most two subelements per node. 14, 15, 39, 40, 59, 68, 79, 80

boolean operation Mathematical concept that combines geometric or topological elements using set operations such as union, difference or intersection. 32, 52

cartesian coordinates System for representing points in a two- or three-dimensional space by their orthogonal distance from the origin. 26, 142

cellcomplex Three dimensional entity describing a topological space composed by multiple cell objects. 31, 32, 63

centrality Measure of influence of an entity in a graph structure. 22, 54, 123

centroid Geometric center of a shape or point set. 68, 71

cohomology Operation in algebraic topology which measures the discontinuity or higher-dimensional characteristics of an entity. 30

continuous function Function without sudden changes which is defined for all given points. 30

convex Shape or set that remains above any line connecting its points. 34, 67–69, 71, 73

convolution Mathematical operation that creates a single function based on the combination of two input functions. 28, 96

cross-entropy Loss function that measures the deviation between predictions and actual values. 95

decision tree Model used to classify or predict based on binary decisions. 98–100, 128

degree of compactness Measure quantifying the degree to which an object deviates from a perfectly circular shape. 82

deterministic Fully predictable set of rules or algorithms. 42, 48, 74, 77

dictionary Data structure in Python that stores pairs of keys with their corresponding values. 84, 86, 87

downhill simplex algorithm Optimisation algorithm that iteratively computes a geometric shape in order to find the optimal solution. 79

ensemble learning method Machine learning technique which combines predictions from multiple individual models for the sake of performance or generalisation. 60, 99, 100

epoch Amount of times a machine learning model passes through the entire dataset during the training process. 94, 96

euler characteristic Numerical measurement of topological properties. 15

f1-score Combination of precision and recall metrics into a single value. 96, 97

fitness function Function for evaluating the performance of a solution in optimisation algorithms. 39, 41

glazing percentage Ratio of the window area to the total facade area of an architectural object. 84

greedy algorithm Algorithmic method that makes local choices at each iteration in order to find an optimal solution on a general level. 39

holdout Machine learning technique where a percentage of the data is used only for evaluating the model's performance rather then for training. 94, 96

homotopy Continuous transformation of two functions or topologically defined elements. 30

isovist *Space Syntax* concept that represents the visible area from a specific point in space. 33, 51, 124

knowledge graph Graphical representation of semantic information about real-world data. i, 6, 63, 64, 66, 84, 86, 87, 106–108, 110, 128, 140

learning rate *Hyperparameter* that defines the step size and pace during deep learning processes. 96

lloyd algorithm Iterative method based on Voronoi diagram computation, used to regularise the calculated Voronoi regions. 71, 73

loss Measure of deviation between prediction and actual values in the context of machine learning. 95, 96

manifold Topological space that can be represented through *Euclidean geometry* and thus can be described by cartesian coordinates. 30–32, 52, 55, 56, 125, 135

negative log-likelihood Loss function that measures the distance between prediction and actual values by computing the negative logarithm of the *likelihood function*. 95

node degree The amount of edges connected to a node in a graph structure. 22, 25

non-euclidean Not adhering to *Euclidean geometry*. 26

one-hot-encoded Feature encoding technique where each individual category is converted into a list of zeros with only a one in the category's position. 131, 147

open source Software or technology with a freely accessible source code. 5, 6, 8, 19, 42, 60, 107, 134, 135, 138, 139

origin bias The existence of patterns based on the place or source of origin. 5, 7, 60, 110, 113

overfitting When a model performs well on the training data but fails to generalise to unseen data. 97, 99, 100, 105

pooling Operation in ANNs that reduces the dimensionality of the input features through calculation of maximum or average values. 93, 95, 96, 129

precision Proportion of TP classes among all positive predicted classes. 64, 96, 97

python High-level programming language with a focus on simplicity and readability. 9, 31, 80, 81, 86–88, 93, 135, 136, 138, 139, 142, 143

quantile Statistical operation that divides a probability distribution into specified intervals thus identifying probability thresholds. 63, 86, 140, 147

recall Proportion of correctly predicted positive classes among all TP classes. 96, 97

sampling Process of selecting a specific amount of datapoints from a dataset for analysis or model training purposes. 93, 94, 129

site energy consumption Total amount of energy consumed by a building at its location, thus excluding the consumption for excavation, transportation and material fabrication. 87, 93, 144, 147

standard deviation Measure of the distance from the mean. 102

stochastic Randomly determined processes or algorithms. 24, 42, 45, 46, 48, 73, 74

subset random sampler Method that randomly selects a set of datapoints from a dataset. 94

underfitting When a model lacks the capacity to capture the patterns in the data. 105

validation Process of evaluating a model on a separate dataset in order to assess its accuracy. 93, 94, 96

Acronyms

AEC	Architectural Engineering and Construction. 6, 8, 20, 50–53, 55–57, 98, 106, 124, 127, 134–136, 138, 139
ANNs	Artificial Neural Networks. 128, 130
API	Application Programming Interface. 31, 131, 134, 135
ASHRAE	American Society of Heating, Refrigerating and Air-Conditioning Engineers. 85
BEM	Building Energy Modelling. 46
BIM	Building Information Model. 3, 19, 20, 51–55, 57, 60, 63, 66, 80, 82, 84, 85, 87, 110, 111, 124, 125, 129, 135, 136, 138, 142, 144, 147
Brep	Boundary Representation. 84, 142, 144
CAD	Computer Aided Design. 3, 4, 110, 125, 136
CFD	Computational Fluid Dynamics. 136
DA	Daylight Autonomy. 42
DF	Daylight Factor. 42, 55
DGCNN	Deep Graph Convolutional Neural Network. 57, 64, 93, 94, 96, 98, 99, 102–105, 109–111, 113, 149
DGL	Deep Graph Library. 87, 93, 94, 135, 139, 143, 147
DST	destination. 144
DXF	Drawing Interchange Format. 135
EPW	EnergyPlus Weather Format. 84, 143
FEA	Finite Element Analysis. 136
FN	False Negative. 97
FP	False Positive. 97
GAEs	Graph Autoencoder. 29
GANs	Generative Adversarial Networks. 56, 57, 127, 128, 130

GATs Graph Attention Networks. 29

GCNs Graph Convolutional Networks. 28, 29

GF Glare Factor. 42

GIS Geographic Information System. 138

GNNs Graph Neural Networks. 28, 29, 57, 113, 129

GRNNs Graph Recurrent Neural Networks. 29

HVAC Heating, Ventilation and Air Conditioning. 137

ID identification. 84, 143, 144

IFC Industry Foundation Classes. 19, 20, 51, 52, 84, 111, 112, 124, 135, 136, 142, 144

IoT Internet of Things. 134

JSON JavaScript Object Notation. 84, 140, 144

LCA Life Cycle Assessment. 137

MAE Mean Absolute Error. 98

MLP Multi-layer Perceptron. 98–100, 103–106, 111

MSE Mean Squared Error. 95, 96, 98

MST Minimum spanning tree. 25, 82, 83

NREL National Renewable Energy Laboratory. 137

OSI Open Source Initiative. 135

OSM OpenStudio Model. 85

OSS open source software. 134, 135, 138, 139

RMSE Root Mean Squared Error. 96, 98, 102–105

RNNs Recurrent Neural Networks. 29, 128

SQL Structured Query Language. 85

SRC source. 144

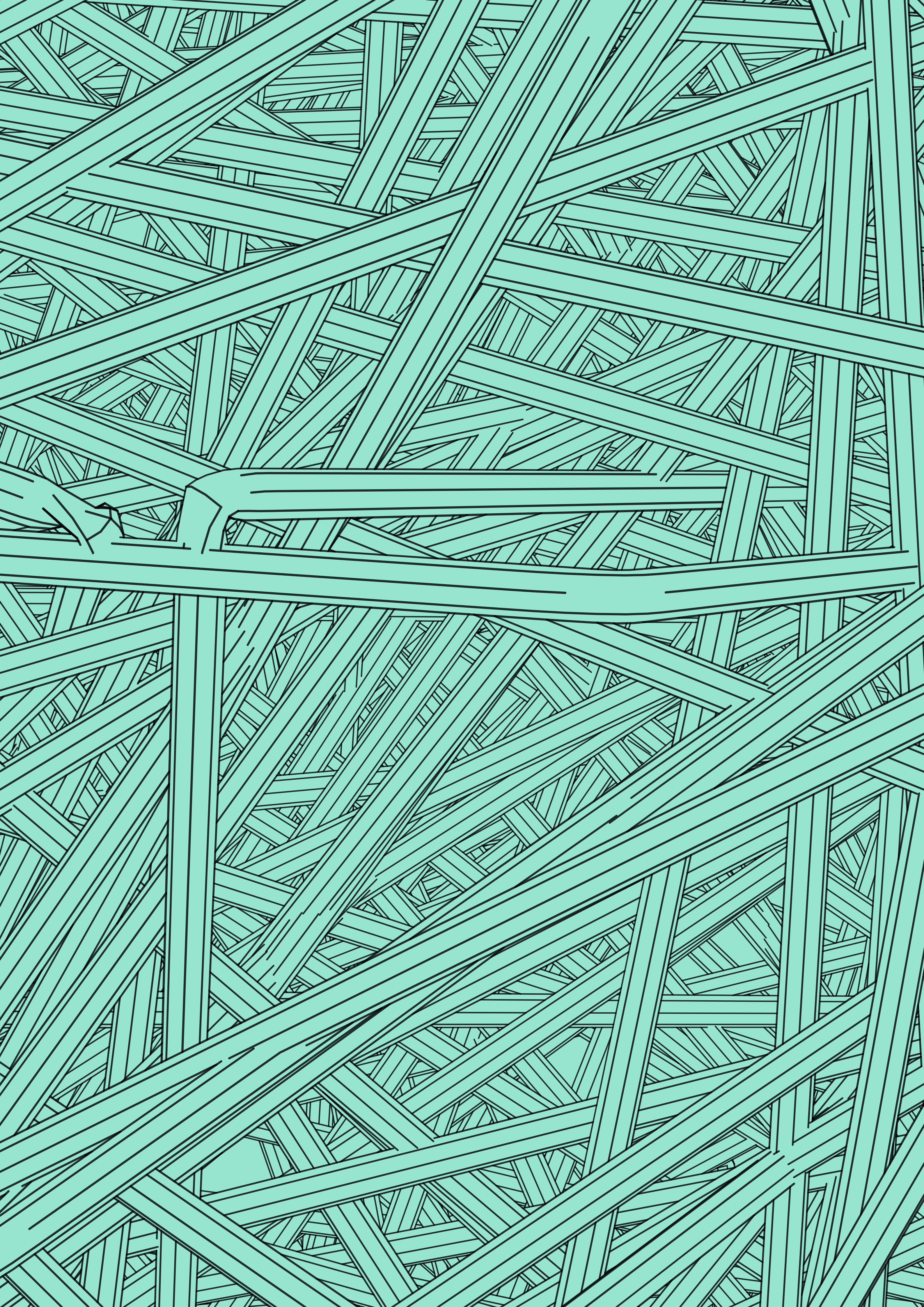
TP True Positive. 97

UDI Useful Daylight Illuminance. 42

VGA Visibility Graph Analysis. 20, 51, 124

Part I

Overview



Chapter 1

Introduction

1.1 Digitalisation in Architecture

In recent decades, advances in technology and digitalisation have led to significant changes in a wide range of professions. Creative fields such as art and design have benefited significantly from the application and integration of various computer- or algorithm-based methods. Architecture, as a discipline at the intersection of technology and artistic practice, plays an interesting role in the adaptation of digital processes[1]. Whereas a few decades ago the profession of architecture was characterised by pencil and paper, today the mouse and computer screen constitute the principal working tools in the architectural design process.

This is only the visible surface of the digitisation of the architectural profession, as the entire working process of architects has been automated, optimised and structured by computer-based tools. With the development of computer-aided design methods such as BIM or more broadly CAD, the static, inflexible nature of representation methods in architecture has been remedied, opening up a wide range of new possibilities in all design phases. Today, a complete design process in a multi-dimensional environment is the norm, allowing the simulation, integration and manipulation of three-dimensional models in conjunction with information on materials, dimensions, construction details and many more. A constant back and forth between conceptual design decisions and construction details is thus largely seamless and is evident in many contemporary designs[2] based on the fusion of detail and creative or programmatic intent.

If we look at the history of architecture, it becomes clear how closely mathematics and architecture are intertwined, for example in the programmatic conception of projects or in the application of geometric rules at the plan level. However, it also becomes apparent that this interplay usually manifests itself at a highly theoretical level[3] and thus often has little tangible impact on the habitability of architectural objects, which can be explained by the lack of accessibility of fundamental mathematical concepts for architects. Therefore, in the design reality, there is unfortunately still a certain distance between useful mathematical concepts and their concrete, beneficial application in the conception phase. Concepts such as topology and graph theory are almost exclusively found at a very scientific level

[1]Chaillou 2022, *Artificial Intelligence and Architecture: From Research to Practice.*

[2]Pena et al. 2021, “Artificial intelligence applied to conceptual design. A review of its use in architecture”.

[3]Baglivo and Graver 1983, *Incidence and symmetry in design and architecture.*

in academia and are therefore difficult to access for traditional or less privileged offices with limited scientific resources.

The application of computational scientific methods[4], on the other hand, appears to have become more democratic with the widespread adoption of market-leading architectural software and the availability of third-party extensions. However, only a minority of traditional architectural practices have a clear overview of machine learning-based tools, and even fewer know how they work or how to make sense of them. In addition, such tools are often embedded in specific software, making interoperability difficult and thus becoming a software-specific feature. It is precisely this lack of clarity about the tools supported by artificial intelligence that hinders the general adaptation of such useful functionalities and contributes to the inequality between underprivileged architecture firms and the monopoly position of dominant proprietary architecture software. One possible reason for this situation is the need for essential resources for the development of such machine learning based models, such as time, economic resources such as budget, energy and computing power, but also access to adapted, evaluated and diverse training datasets.

1.2 Current Challenges

Need for a Universal Space Syntax Language The lack of a universal spatial syntax language in architectural design hinders the development of standardised guidelines and theoretical foundations. This results in arbitrary topological relationships between individual architectural elements, which can lead to inconsistencies and inefficiencies in design. A universal language would allow for better communication and understanding within the field and ultimately contribute to a more effective design practice. Furthermore, the skillful application of spatial syntax theories during the design process would lead to an increase in architectural quality during the occupancy of the constructed objects.

Limited Design Feedback During Early Project Stages Current design practices often lack feedback in the early stages[5], leading to arbitrary decisions and potential inconsistencies between initial drafts and detailed elaborations. Implementing a system that provides feedback throughout the design process on various parameters, such as performance or architectural feasibility, would allow architects to make more informed decisions and balance the creative and technical aspects of their projects.

Special Training and Computationally Intensive Simulations The setup and execution of complex physical simulations and intelligent CAD tools can be demanding and laborious, requiring special training and significant computational power[6]. This makes it difficult for non-specialist users to obtain accurate results and prevents continuous verification during the design process.

Need for Mathematical Foundations in Relationship Descriptions Understanding the spatial structuring and programmatic aspects of architecture requires research into the mathematical foundations of relational descriptions. By developing analysis methods

[4]Caetano, Luís Santos, and Leitao 2020, “Computational design in architecture: Defining parametric, generative, and algorithmic design”.

[5]Paterson et al. 2013, “Real-time Environmental Feedback at the Early Design Stages”.

[6]Chatzivasileiadi et al. 2018, “The effect of reducing geometry complexity on energy simulation results”.

based on such topological properties that can accurately represent and concretise programmatic intentions, architects can more effectively optimise spatial organisation, energy efficiency, natural lighting and load-bearing capacity.

Lack of Publicly Available Graph Datasets There is currently a significant gap in the availability of comprehensive graph datasets based on architectural objects[7]. This poses a challenge to researchers who need reliable data for algorithm development and testing. The creation and publication of such datasets would enable the academic community to make more substantial advances in the field and ultimately improve the traditional architecture practice.

Poor Application of Topological Concepts Conventional architectural projects often struggle to apply topological concepts effectively, making it difficult to automatically integrate and query rule-based information in *Building Information Modelling.* Addressing this issue would help to ensure a coherent spatial design and improve overall project outcomes.

Data Form-Specific Challenges Architectural objects can be represented through various media, but reducing them to a single one often leads to a loss of information. Developing machine learning models that can efficiently manage diverse datasets and account for *origin bias* in architectural training data would help to ensure more meaningful and context relevant suggestions in the design processes.

1.3 Contribution

Feedback Loop in Early Design Stages This research contributes to the field by exploring methods and ways to implement feedback loops in the initial design stages. These feedback loops would allow architects to take decisions based on key parameters such as energy consumption, ultimately leading to more efficient and sustainable building designs and avoiding costly back and forth between detail drawing and conceptual design.

Exploration of Relational Datasets and Machine-Assisted Optimisation The thesis investigates the need for relational datasets in graphical form to enable machine-assisted optimisation of architectural parameters. It further explores the extraction of insights through analysis methods based on graph based input data, providing a foundation for further research and practical applications in this area.

Creation and Publication of an Architectural Graph Dataset To address the lack of publicly available graph datasets based on architectural objects, this work contributes by creating (chapter 4) and publishing (section 6.1) a comprehensive graph dataset which can serve as a valuable resource for researchers and practitioners in architecture and related fields.

Meta-Analysis of Open Source Tools in Architecture Furthermore, this research includes a meta-analysis of open source tools available to the architectural community

[7]Alymani, Jabi, and Corcoran 2022, "Graph machine learning classification using architectural 3D topological models".

(section A.3). By investigating and promoting open source file formats and software packages, the study aims to increase the interoperability of architectural data and encourage freedom in the AEC industry.

Application of Graph Theory in Architecture An essential part of this work focuses on the detailed analysis of possible applications of graph-theoretical concepts in the architectural context. For this purpose, the mathematical foundations are explained, graph-based analysis methods and their advantages are reviewed, and a concrete application of the theory is explored and evaluated in the experimental part of this thesis.

Analysis and Application of Topological Tools The thesis provides a thorough analysis of existing architectural applications of topological tools and their potential for graph-theoretic concepts. It explores the usefulness of topological transformations and analysis methods, ultimately demonstrating the value of an integration into the architectural practice.

Machine Learning Applications in Architecture The research further contributes by exploring the application of machine learning in the architectural profession. Specifically, graph machine learning methods are developed for predicting energy consumption and efficiency of architectural designs. This constitutes an essential step towards a closer connection between new technological, scientific methods and traditional architectural practices.

Generation of a Synthetic Architectural Dataset This work produces a synthetic architectural dataset, generated using parametric algorithms and respecting architectural rules while maintaining geometric variance. This dataset, published as an open source resource, bridges the gap between different scientific fields, thus confirming the position of architecture as a polyvalent science. Furthermore, the documentation and publication of the generation pipeline is intended as a basis for subsequent work aiming at the generation of synthetic architectural data.

1.4 Research Questions

Within the context described in the previous section, this manuscript identifies four main research questions related to graphs and topology, feedback in early design stages, architecture and machine learning and synthetic architectural datasets. By exploring these research questions, this manuscript aims to shed light on the potential benefits and challenges of integrating scientific and mathematical methods into architectural practice. These research topics are examined in detail to provide insights into how they can be applied to enhance creativity, coherence and efficiency in architectural design, as well as to explore new possibilities for the field of AEC. The questions are as follows:

- What is the role and potential benefits of integrating *knowledge graphs* and topological methods into everyday architectural practice? How can abstract relational information from graphs be meaningfully applied to project design, and what are the challenges and opportunities that arise from its use?
- How can design feedback be provided in the early stages of the design process to enhance creativity and coherence between initial design and detailed elaboration in later project stages? What are the benefits and challenges of integrating indicative

simulation variables in early design stages and what methods can be used to optimise this process?

- What role can machine learning models play in architectural design and what are their current practical and academic research applications? What potential benefits can trained models provide for design feedback iteration, and how feasible is it to use annotated graphs as input to machine learning methods for the abstract representation of geometric information?
- How can architectural datasets be created synthetically and to what extent can the information they contain be abstracted? What are the benefits, methods and potential issues associated with automatic floor plan generation and its application methods? Can synthetic dataset creation remedy origin bias and how can the creative diversity and adaptability offered by automatic, regulated synthetic dataset creation be evaluated?

1.5 Outline

This thesis is divided into two parts: Part I: *Overview*, which deals with the explanation of the basic concepts as well as the state of the art, and part II: *Contribution*, which documents the experiments carried out along with their evaluation and conclusions. Finally, part III of this thesis consists of the *Appendix*, which contains further reading, additional information and code samples.

In **Chapter 1: *Introduction***, the context for the study is set. Section 1.1: *Digitalisation in Architecture* discusses the increasing use of digital tools in everyday architectural practice, followed by an overview of the current challenges associated with the digitalisation process in section 1.2. The contributions of this work are presented in section 1.3 and the research questions driving the study are outlined in section 1.4.

Chapter 2: *Preliminaries* delves into the basic concepts used throughout this study. Section 2.1 introduces the principal concepts of graph theory, and its applications in architecture are discussed in section 2.1.2. The topological analysis methods and tools used in this work are covered in section 2.2. The role of simulation in architectural design and the different possible applications are explored in section 2.3, with particular emphasis on energy performance and optimisation methods in sections 2.3.1 and 2.3.2.

Chapter 3: *State of the Art* reviews the literature on graphs and topology in architecture in section 3.1, feedback in early design stages in section 3.2), architecture and machine learning in section 3.3, and synthetic architecture datasets in section 3.4.

Chapter 4: *Synthetic Dataset Generation* details the process of generating a synthetic graph dataset based on architectural objects for subsequent use in this study. It includes discussions on different space partitioning algorithms in section 4.1, the integration into a parametric generation framework (section 4.2), the application of architectural rules (section 4.3) and post-processing techniques such as information annotation, energy performance simulation, data mapping and graph retrieval detailed in sections 4.4.1 to 4.4.4. The chapter concludes with an evaluation of the results in section 4.5 and a discussion about synthetic data generation in section 4.5.2.

Chapter 5: *Graph Machine Learning* presents the structure and hyperparameters (sections 5.1 and 5.2) of the proposed graph-based machine learning approach, as well as the evaluation metrics used for classification and regression tasks in sections 5.3.1 and

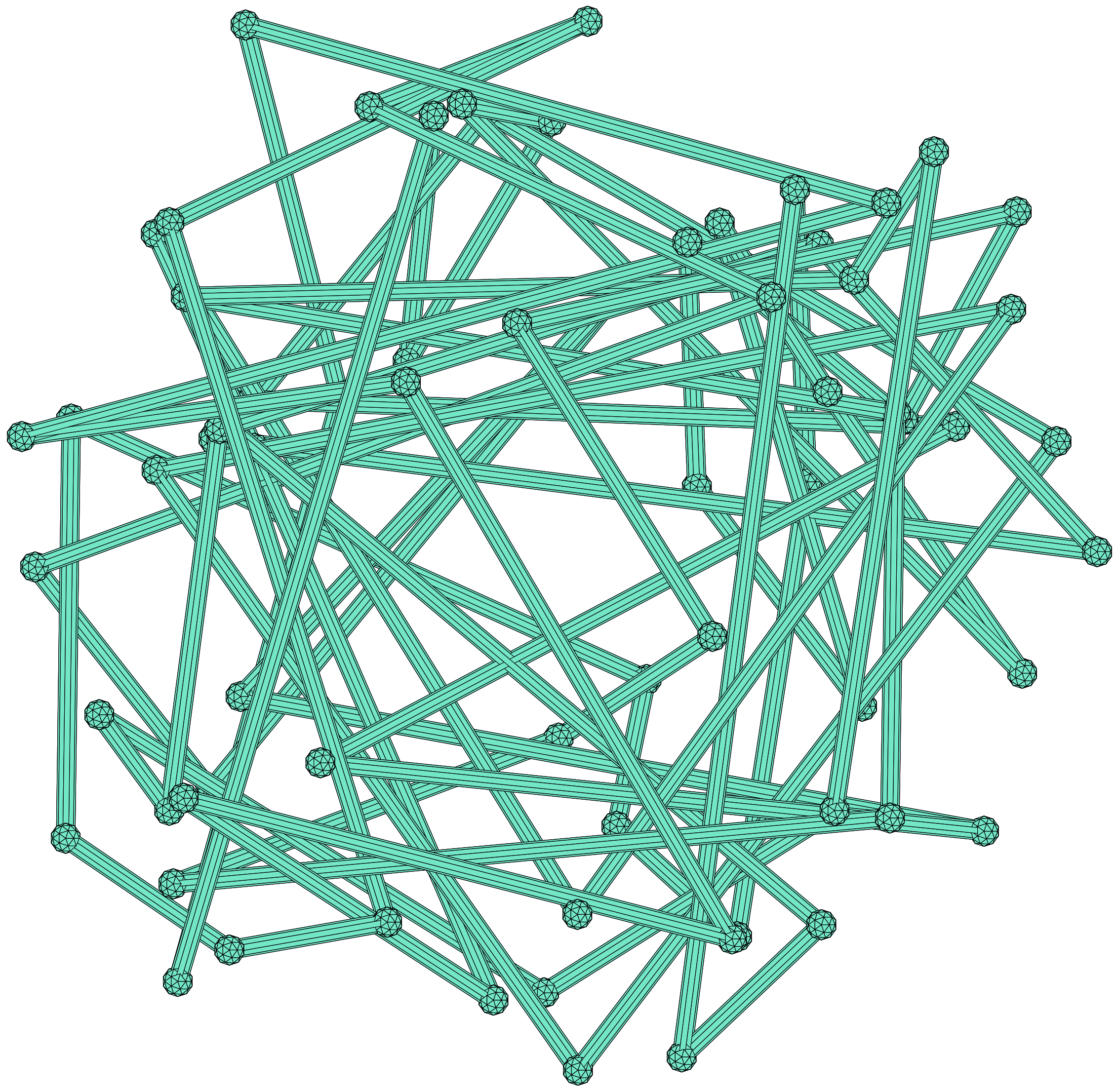


Figure 1.1: Graph Structure

5.3.2. This explanation is followed by the introduction of the comparative framework of both models in section 5.3.3. The results and comparison of the different methods are presented in section 5.4, where conclusions about the results of each model are drawn in sections 5.4.2.2 and 5.4.3.2.

Finally, **Chapter 6:** ***Conclusion and Future Perspectives*** summarises the contributions and provides answers to the previously announced research questions in sections 6.1 and 6.2. This is followed by an outline of the learned lessons, future research directions, limitations, added value for architects and concluding remarks (sections 6.1 to 6.6). The manuscript concludes with a bibliography and two appendices: A and B, where the first part of the appendix provides additional content such as further literature, a proposal for a potential application of the experimental results, and a discussion on the topic of open source in the AEC industry (sections A.1 to A.3). The part B of the appendix presents and explains the raw data, divided into geometrical, graphical and informational data in sections B.1 to B.3.

Chapter 2

Preliminaries

This chapter introduces the basics for understanding the essential concepts and terminology. First, a step-by-step description is given of what graph theory defines and what the properties and components of graph structures are. Then, the described concepts and characteristics are examined in their application in the architectural context, where the digital modelling process as such is considered, but also a variety of different graph-based analysis methods and their advantages are explained. Finally, in the context of graph theory, graph-based machine learning is introduced and its functionality, structure and application are explained.

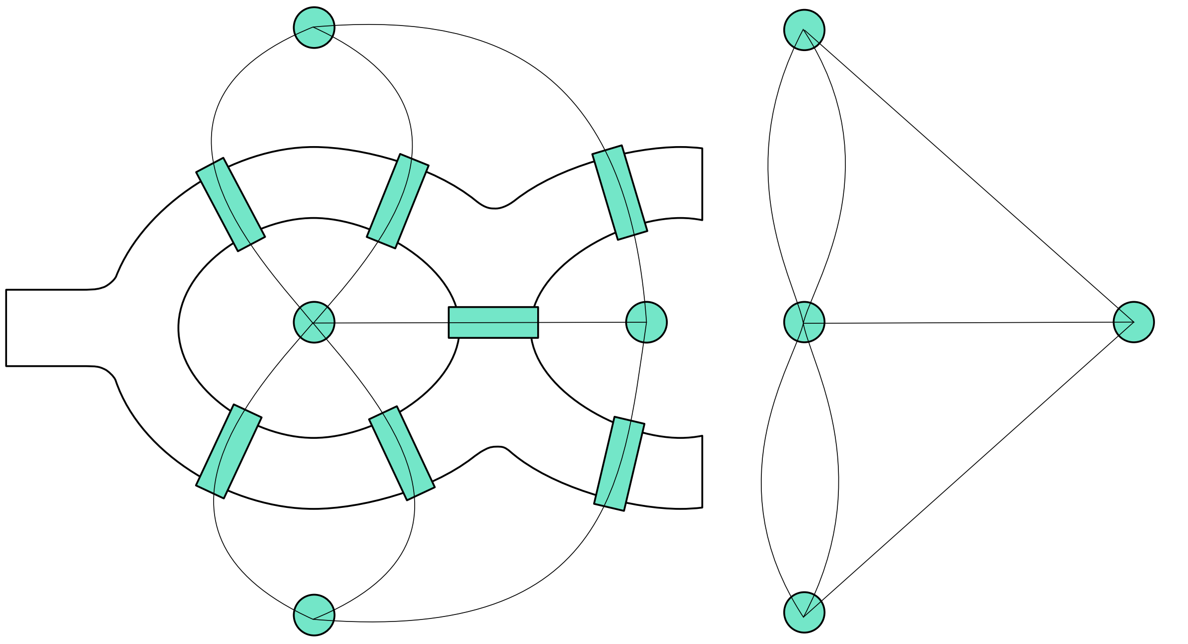
The following section focuses on the introduction of the term *topology* in a mathematical and architectural context. For this purpose, the topological analysis method is explained using the *Python* library *TopologicPy* and its structure. Then the main concepts of *space syntax* theory are recapitulated and illustrated with examples, followed by an explanation of the concept and use of *shape grammar* implementations. This section concludes with a consideration of the basic concepts of different spatial partitioning methods with their respective advantages and disadvantages in the application of automatic floor plan generation.

The last section of the contribution chapter deals with the explanation of the concept of simulation and its different variants as well as possible applications. After a detailed description of the main simulation applications in architecture and engineering, the focus is on energy performance simulation. Finally, different optimisation families and their essential algorithms and functionalities are explained.

2.1 Graph Theory

Graph networks and their derivatives surround us in our everyday lives and influence us on a wide variety of levels. Yet this simple mathematical concept seems to be considered only rarely or at a highly academic level. When considering the fastest geographical route from point A to point B, or how many rooms in a Renaissance castle have to be traversed to get from the reception atrium to the noble chambers, these are graph-theoretical questions that can be solved through established mathematical methods.

Historically, the study of graph theory can be traced back to a problem known as the *Königsberg Bridge* problem. It is an urban connection problem involving the city of *Königsberg*, through which the river *Pregel* flows (figure 2.1). The city consisted of two main islands in the middle of the river and also extended on both sides of the riverbeds,



(a) Connection Graph (b) Simplified Representation

Figure 2.1: Königsberg Bridge Problem

being connected by exactly seven bridges as shown in figure 2.1a. The question that led to this famous problem[1] was whether it was feasible to cross each bridge exactly once and arrive back at the end of the path precisely at the starting point. On a mathematical level, it is a question of traversing the represented network consisting of points and their connections only once, but not repeatedly.

A closer look at this example reveals that the nodes represented are not defined by their geographical position, but only by their relationship or connection to the other points of the network (figure 2.1b). This demonstrates one of the fundamental characteristics of graph structures.

Applications of graph theory can be found in a variety of ways in many fields, such as the analysis and representation of molecular structures in chemistry, where individual atoms represent the nodes and the edges represent the respective compounds to which the molecules owe, among other things, their chemical properties. Another well-known example is the representation of social networks, where the nodes represent single individuals or groups and the edges between them model their relationships. Areas such as transportation networks, traffic infrastructures, computer programs, data structures, financial markets, economic systems and ecological biospheres or phylogenetic trees are also well known and intuitive applications of practical graph theory.

2.1.1 Graphs

But what makes a graph a graph? What is its basic structure and what are its properties? First of all, it is important to establish a general definition of a graph in order to explain and make intelligible its properties and functions in the following steps. A graph is a mathematical object from *discrete mathematics* and *combinatorics* which is composed of

[1]Kantor 2005, “A tale of bridges: topology and architecture”.

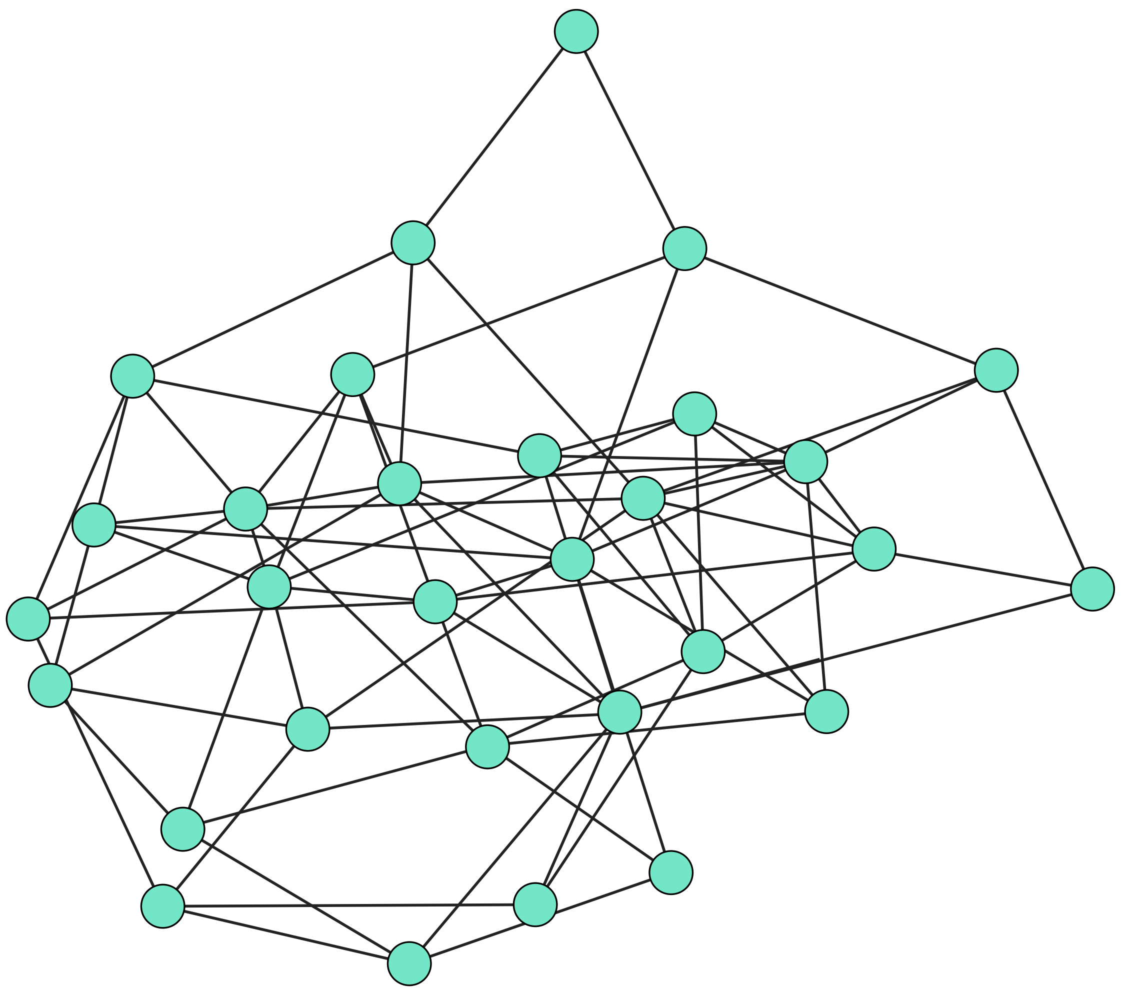


Figure 2.2: Graph Network

a finite, non-empty set of *nodes*, also called *vertices*, and a finite, unordered set of *edges*. Vertices represent points in the graph structure which may be connected by edges.

Since graphs are dimensionally independent, a visualisation of their structure is often misleading, as edges may appear to intersect when in fact they exist completely independently, or vertices may be positioned locally adjacent to each other, but without having any commonality with each other. This abstraction and deceptive dimensionality becomes particularly important to internalise when graph structures are related to geometric objects. As can be seen in figure 2.2, the complexity of the two-dimensional representation of graph structures increases significantly as soon as a certain number of nodes and edges is exceeded. Usually, vertices and edges are labelled with letters or numbers in order to refer to them individually.

In addition, edges can be *unidirectional* or *bidirectional*, meaning that there is a one-way or two-way relationship of the two connected vertices to each other. This is usually represented by arrows along or on the edges (figure 2.3). Once a graph structure has one or more edges with a specific direction, it is called a *directed graph*, which distinguishes it from an *undirected graph*. Another special property of edges in a graph is that they can be weighted, which means that a hierarchy can be created between the edges and thus the designation of the graph changes from *unweighted* to *weighted graph* (figure 2.5).

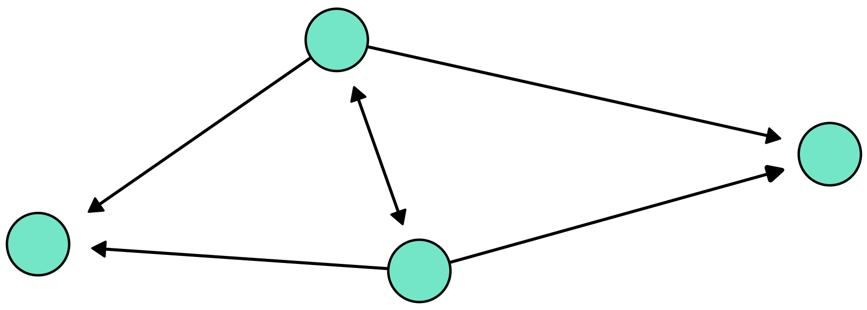


Figure 2.3: Directed Graph

Regarding the naming convention[2], in mathematical terminology a graph is described by a G and contains vertices $V(G)$ with their corresponding edge set $E(G)$, so $G = (V, E)$. An edge consists of a vertex pair $e = (u, v)$ and is usually abbreviated to uv. For undirected edges $e = (u, v)$ and $e = (v, u)$ holds, since an edge connecting vertex u to vertex v also connects vertex v to u. The number of vertices in G is notated as $|V|$ and the set of edges as $|E|$.

2.1.1.1 Components

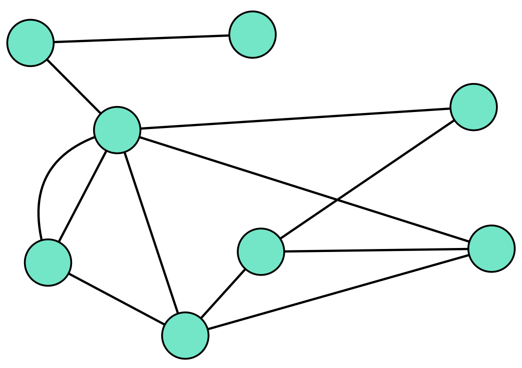
A graph consists of two basic components, which have already been introduced as nodes and edges. Nodes, or vertices, are the basic units of the graph and are usually labelled with a unique identifier, such as a letter or number. The total number of nodes $|V|$ in a graph is called the *order of the graph* and is usually represented by the letter n.

Edges connect the nodes in the graph and provide information about which nodes are directly connected. An edge is therefore defined by an ordered pair of connected nodes and is often represented by the notation (u, v), where u and v are the two nodes connected by the edge. If the graph is undirected, this means that the edge can be traversed in either direction and is often written as (u, v) or (v, u). If the graph is directed, the edge is represented by an arrow from u to v and is written as (u, v). The total number of edges in a graph is called the *size of the graph* and is usually represented by the letter m. For undirected graphs, the relation is $m \leq n * (n - 1)/2$, while for directed graphs it is $m \leq n * (n - 1)$[3].

There are several special cases for the two components of graph structures, such as *isolated nodes* (figure 2.4b), which represent points that have no connection to other points of the graph and thus have no edges. A special case for edges are *loops*, which have the same start and end vertex and thus connect a node to itself: $e = (u, u)$. Furthermore, two points u and v can be connected by multiple edges, which means that $E(G)$ contains either (u, v) or (v, u) more than once (figure 2.4a). Multiple edges can occur in directed graphs, but also in undirected graphs.

As already described by the possibility of weighting edges, information can be added to the individual elements, since each element can be referenced precisely and independently. For example, nodes can contain the names of individuals in a social network, or edges

[2]Wilson 1979, *Introduction to graph theory*.
[3]Earl and March 1979, "Architectural applications of graph theory".


(a) Multigraph

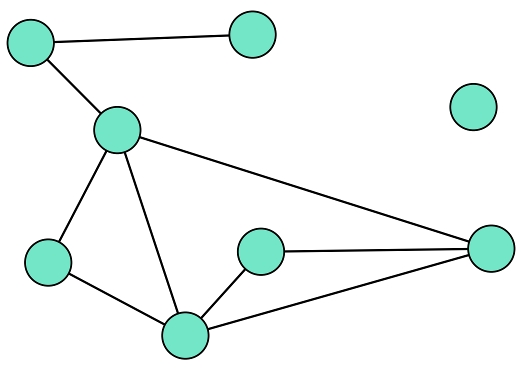
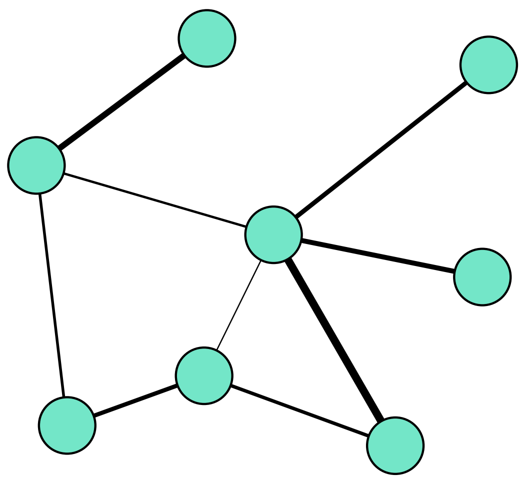

(b) Isolated Node

Figure 2.4: Graph Variations


(a) Weighted Edges

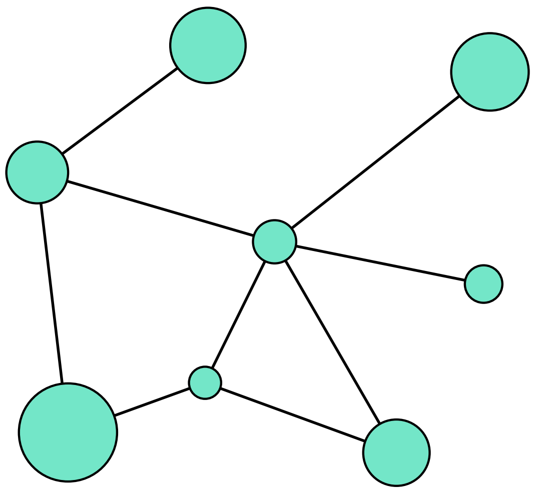
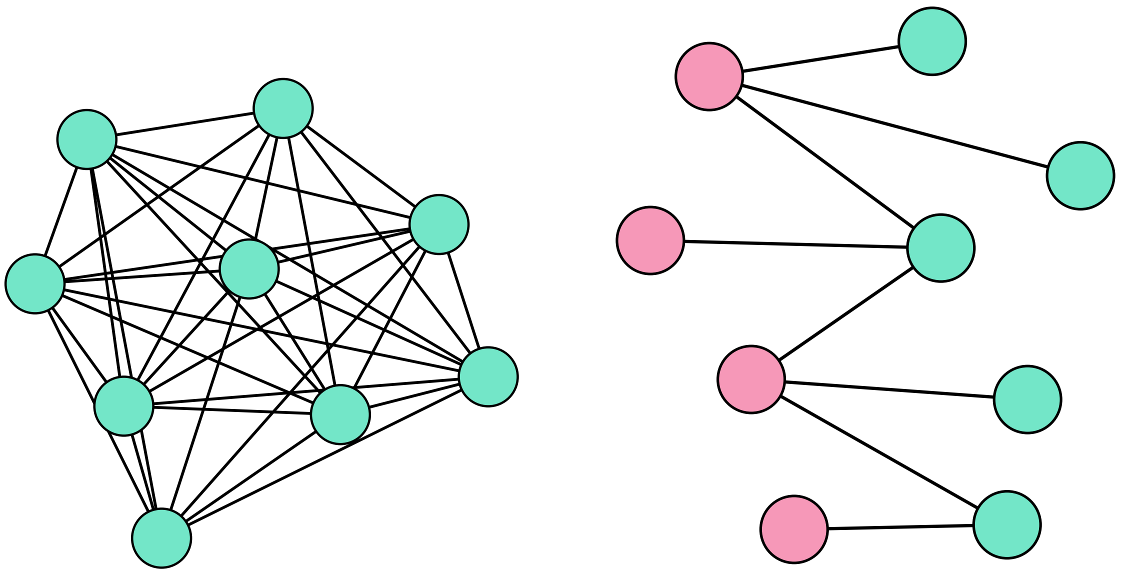

(b) Weighted Nodes

Figure 2.5: Weighted Graphs

can be classified into different categories, such as the type of relationship that different actors in a play have with each other, such as related, in love, married, and so on. Value notation is also possible, where individual nodes can be referenced by the age of the people, or edges can hold the number of months in the relationship. A common example of such graphical knowledge information is Zachary's karate club[4], which serves as an example of how relational semantic information can be translated into graphical form.

Another component of a graph are *subgraphs*, which form a new graph structure from a subset of the vertex set $V(G)$ of the original graph G and contain the edges corresponding to the vertices. Similarly, *supergraphs* are all graphs formed by adding vertices or edges to a graph structure. This implies that if F is a subgraph of G, then G must be a supergraph of F.

[4]Sanchez-Lengeling et al. 2021, "A gentle introduction to graph neural networks".



(a) Complete Graph (b) Bipartite Graph

Figure 2.6: Graph Properties

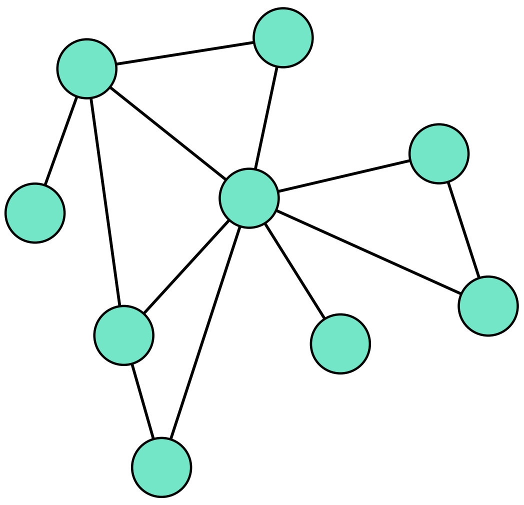
2.1.1.2 Properties

There exists a large variety of different forms and special cases in which graph networks can occur. Directed graphs, also called *digraphs*, which are defined by the order of vertex pairs, have already been introduced. Similarly, the properties of weighted graphs, which are defined by adding a weight function w and thus formulated as $G = (V, E, w)$, have been explained. However, these two cases are far from being the only special cases of graph structures. A graph can be called a *complete graph* if every single vertex is connected to every other vertex by an edge (figure 2.6a), allowing the number of edges in a complete graph to be calculated by the formula $K(n) = n*(n-1)/2$ for undirected graphs. Another subset is called a *bipartite graph* when the set of vertices of the graph can be divided into two disjoint subsets such that each edge of the graph connects a vertex from the vertex subset A to a vertex from the subset B (figure 2.6b). In this case, the bipartite graph can be represented as $G = (A, B, E)$, where A and B represent the two vertex subsets and E is the edge set connecting A and B.

Another essential concept in graph theory are *trees*[5] (figure 2.7b). They are circle-free graphs in which there is no path from a vertex back to itself. Like conventional graphs, they consist of a subset of vertices and edges, but the edges are directed, and so trees are directed graphs. The analogy with biological trees is that there is only one vertex in a graph tree, called the root, which marks the starting point of the tree. All vertices of the object have exactly one parent node, except for the root node. However, each vertex can have several child nodes. A well known variant of trees is the so-called *binary trees*, which differs from conventional trees in that each vertex has only two child nodes.

Of interest in geometric representations of graphs are certain graph structures where, by definition, no edge intersects when represented on a plane. This particular property of graphs is called *planar* (figure 2.7a). In this case, the faces formed between the edges can

[5]Wilson 1979, *Introduction to graph theory.*


(a) Planar Graph

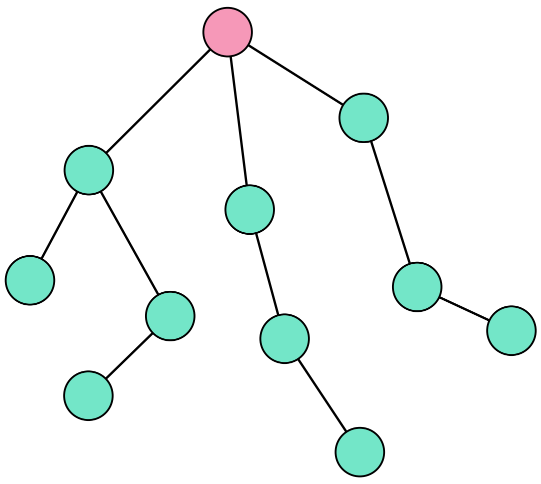

(b) Tree

Figure 2.7: Graph Properties

be considered, and it holds that the *Euler characteristic* described by $|V| - |E| + |F|$[6] always gives 2 for planar graphs, where $|F|$ is the number of faces formed by the edges and $|V|$ and $|E|$ represent the number of vertices and edges. Planar graphs offer interesting geometric interpretations and allow for an easy understanding once represented in the two-dimensional space.

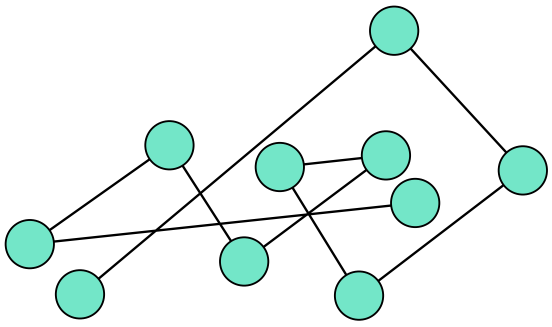
2.1.1.3 Representation

As previously mentioned, graphs are space-independent representations since the position of vertices per se is not bound to local information and edges, contrary to intuitive interpretation, do not represent a geometric connection between pairs of vertices, but only their relationship to each other. Since we as individuals can only realise visual representations in three dimensions, and in the case of immobile representations are generally limited to two dimensions, a visual representation of spatially independent mathematical concepts and objects, such as graphs, always involves a certain degree of abstraction. In order to avoid misinterpretation of the information to be conveyed, it is important to remain conscious of this abstraction when considering visualised graphical information, especially when graphs are derived from geometric figures.

Numerical, tabular representations can counteract this risk of confusion by their inherently abstract form of representation. However, this level of abstraction is a major drawback of tabular representations of graph structures, since relationships between nodes are more difficult to perceive and a greater amount of representative information is generally required. Whereas in a visual representation a non-existent edge is simply not drawn, in a tabular representation a value is required to indicate its non-existence.

The main visual representations of simple graph structures are node diagrams (figure 2.9) and their individual special cases such as binary trees or directed graphs. This type of representation is particularly suitable for planar graphs and when it is necessary to quickly convey an overview of the entire graph structure and the relationship of individual nodes.

[6]Dawes and Ostwald 2013, "Applications of graph theory in architectural analysis: past, present and future research."


(a) Unordered Representation

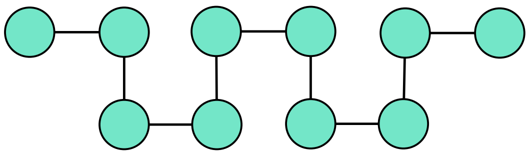

(b) Ordered Representation

Figure 2.8: Ordering of Graph Nodes

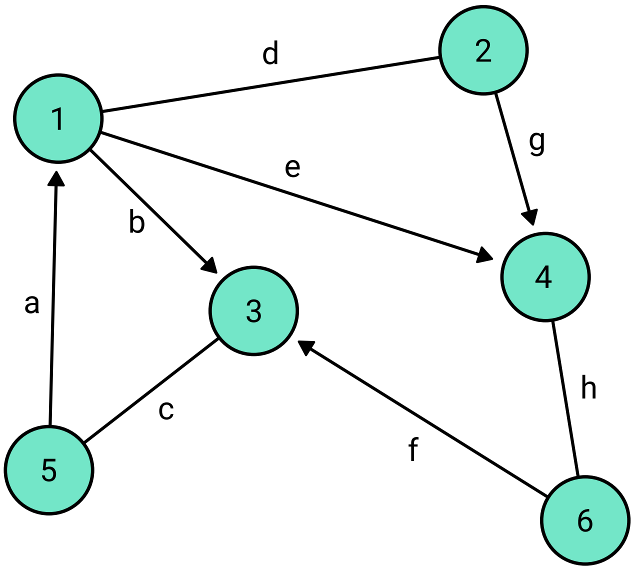


Figure 2.9: Graph Representation

There are no explicit visual rules for representing the various components of graphs, but it is common to represent vertices as points or circles of the same size, and edge connections as straight lines between points if possible, or curves if straight lines are not possible. For directed edges and graphs, the orientation is usually symbolised by an arrow on the edge[7]. As a general convention, there should be as few crossings between graph edges as possible to avoid dimensional confusion (figure 2.8).

The most common numerical and tabular representations of graphs are the *adjacency matrix* (table 2.1a), the *adjacency list* (table 2.1c), the *incidence matrix* (table 2.1b) or the *edge list* (table 2.1d). Here, edge lists provide the most intuitive representation, since it concerns simply a listing of the references to the individual vertex pairs connected through edges.

A slightly modified form of this listing is the adjacency list, where for each vertex in the graph the references of the vertices connected by edges are listed. Incidence and adjacency matrices are two tabular forms of representing relational networks and differ

[7]Wilson 1979, *Introduction to graph theory.*

	1	**2**	**3**	**4**	**5**	**6**
1	0	1	1	1	0	0
2	1	0	0	1	0	0
3	0	0	0	0	1	0
4	0	0	0	0	0	1
5	1	0	1	0	0	0
6	0	0	1	1	0	0

(a) Adjacency Matrix

	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**
1	-1	1	0	1	1	0	0	0
2	0	0	0	1	0	0	1	0
3	0	-1	1	0	0	-1	0	0
4	0	0	0	0	-1	0	-1	1
5	1	0	1	0	0	0	0	0
6	0	0	0	0	0	1	0	1

(b) Incidence Matrix

1	2	4	3
2	1	4	
3	5		
4	6		
5	3	1	
6	4	3	

(c) Adjacency List

(1,2) (1,4) (1,3) (2,1) (2,4) (3,5) (4,6) (5,3) (5,1) (6,3) (6,4)

(d) Edge List

Table 2.1: Numerical Representation Methods

mainly in that in the case of the adjacency matrix the rows and columns represent the same vertices and thus always result in square tables, which are mirrored diagonally in the case of undirected graphs. The values of the table are binary and indicate whether an edge exists between the two reference vertices or not.

In the case of the incidence matrix, one row or column is the reference to the edges and the other remains the reference to the individual vertices. If an edge j is connected to vertex i, then the entry in the i-th row and j-th column of the incidence matrix is a 1 if the edge is outgoing from vertex i, otherwise it is a -1 if the edge is incoming to vertex i. All other entries are 0. An incidence matrix is usually used for directed graphs, but can also be used for undirected graphs.

2.1.2 Graphs in Architecture

The application of graph theoretic methods of analysis has been an integral part of architectural research since the beginning of the last century[8]. The discrete mathematical concepts can be beneficially applied at various levels of design practice, providing insight into spatial and relational structures that would be difficult to relate without their intervention.

In concrete applications, graph theory can be used to analyse, optimise and plan designs and buildings[9]. The use of graphs as a modelling tool also makes it possible to represent and analyse complex structures[10] such as road networks, public spaces or buildings in a simplified form (figure 2.10 and 2.11). Both quantitative and qualitative aspects of the structure can be considered, such as the length and temporal duration of paths, the number and type of connections, or the visual relationships between individual architectural elements.

In academic architectural research, graph theory is also used to study the spatial properties of buildings and urban spaces[11]. In doing so, it can help to study and understand the structure and organisation of spaces as well as the relationships between them. It can

[8]Earl and March 1979, "Architectural applications of graph theory".

[9]Dawes and Ostwald 2013, "Applications of graph theory in architectural analysis: past, present and future research."

[10]Lakshmi, Madhumathi, and Sindhuja 2017, "Graph theory and architecture".

[11]Napong 2004, "The graph geometry for architectural planning".

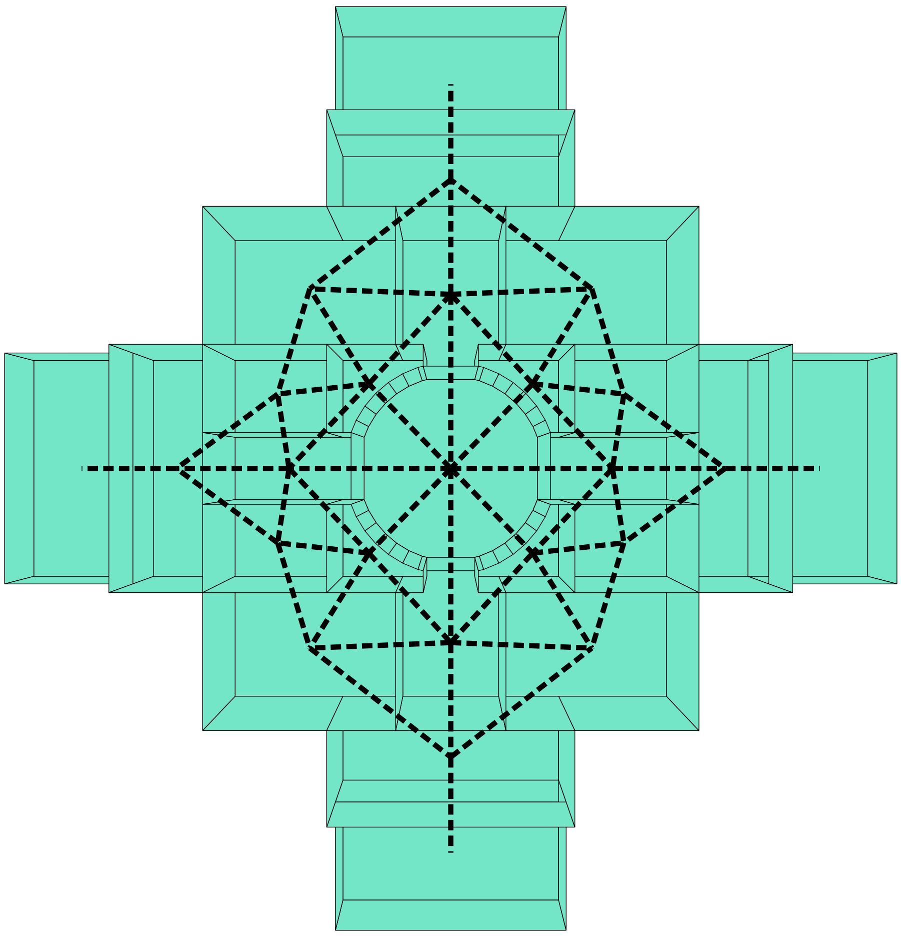


Figure 2.10: Connectivity Graph of Palladio's Villa La Rotonda

equally be used to analyse and evaluate urban design concepts or to develop new urban designs.

The implementation of graph-theoretical methods in project elaboration at the conceptual level can also play an important role, since the abstract and relational information of the mathematical object can form an essential function in the development of programmatic and structural organisation. In some projects, it may also be of great value to construct a detailed social or biological network in order to understand, analyse and optimise existing systems or desired relationships between individuals, groups of actors or families.

The potential of graphs to carry additional information at their nodes and edges is equally shown to be of great value in the architectural context, as it becomes possible to add geometric, dimensional, physical or social information to the network, thus creating as representative an image of reality as possible. In subsequent steps, this allows the simulation and analysis of the effects of architectural interventions.

Graph theory can furthermore be usefully applied to the development of new structures. By modelling the structure as a graph, the designer can test and evaluate different layouts and configurations. For example, the number of connections or the length of paths can be empirically evaluated to maximise the efficiency of the structure while satisfying aesthetic and functional requirements.

A key aspect explored in this thesis in relation to graph structures in the architectural

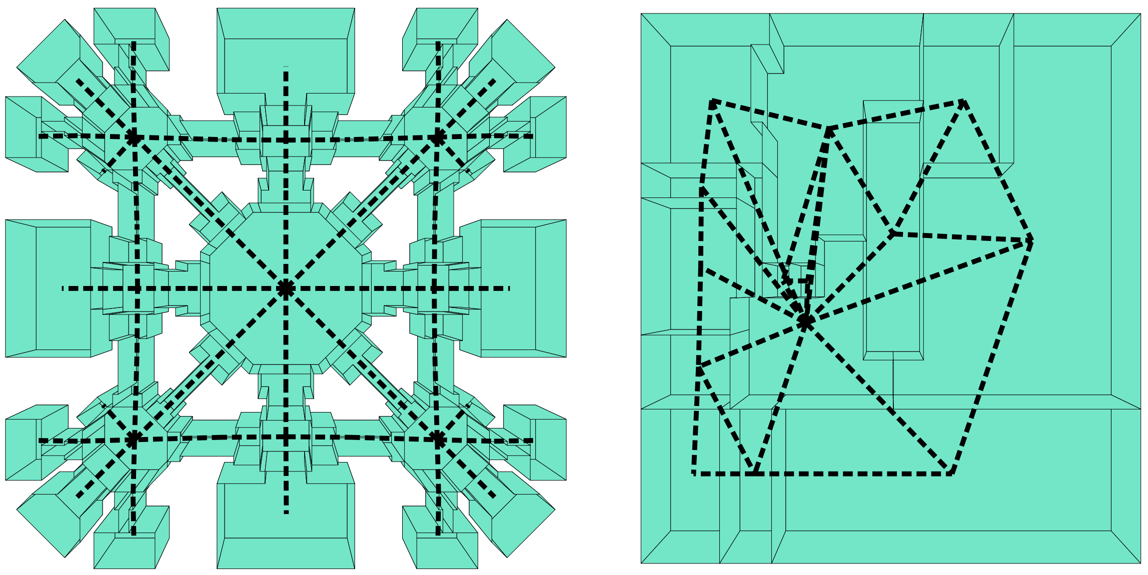

(a) Taj Mahal (b) Villa Savoye

Figure 2.11: Connectivity Graphs of Architectural Examples

design process is the analysis of spatial and elemental relationships. By defining spaces and architectural components as nodes and their relationships to each other as edges, designers can explore and understand these interconnections. This can be used, for example, to represent and evaluate the visibility between different spaces, or to measure the efficiency of the connections between them in order to achieve optimal spatial organisation.

2.1.2.1 Graphs in BIM

Building Information Model, or BIM, refers to the process of creating, managing and using digital architectural models of buildings and architectural elements, in which all relevant data, such as dimensions, materials, components, installations and technical systems, are stored and linked in a central digital model. The aim of BIM is to optimise the planning and construction of projects by bundling and coordinating the flow of information. By using BIM, stakeholders involved in design and construction, such as architects, engineers and contractors, can work more efficiently and collaboratively, reducing errors and costs through centralised data exchange and reconciliation.

An essential aspect of the concept described is interoperability between the parties involved, but also data exchange between established architecture and engineering software solutions. To ensure this, open standards such as the IFC file format are often used, which, due to its open source characteristics, allows data to be shared between different BIM software tools and to be read and processed optimally. Developed and maintained by the *BuildingSMART International* organisation, the IFC file format, which stands for *Industry Foundation Classes*, is a human-readable text file containing structured and hierarchically categorised information about the given architectural object. The information contained and stored is object-oriented and can therefore assign an unlimited number of attributes to a wide range of constructive entities.

Elements in IFC projects can be of diverse nature, such as project sites, levels, walls, ceilings, windows and doors, and attributes can be attached as information about these individual elements, such as a description of their materials, dimensions and relationships to each other[12]. The direct implementation of graphical relationship descriptions in BIM modelling processes has not yet been established, partly due to the aforementioned representation difficulties caused by their abstract structure. However, looking at the compositional logic of IFC file types and their application in everyday architecture, it becomes clear that the integration of graphical representations could be of great benefit for collaboration and analysis capabilities.

Due to their hierarchical structure of interrelated objects, Industry Foundation Classes lend themselves as BIM models to representing the objects as nodes and the relationships between them as edges. This graphical representation allows additional relational information, such as local distances or conceptual links, to be embedded in the basic hierarchical structure of the file format. The integration of such graph-based representations could be then used to analyse and optimise aspects of building design such as space planning, energy and resource optimisation. Such integration would provide a variety of new automatable analysis capabilities and enable additional levels of abstraction in the architectural design process.

2.1.2.2 Graph Analysis

In addition to their organisational and hierarchical capabilities, graph elements offer a variety of different analytical methods that can be of significant use in the design process as well as in the optimisation phase of architectural, urban or landscape projects. In order to provide an overview of these analysis methods, the most important ones are presented and demonstrated in order to evaluate the usefulness of graphs in the field of *Architectural Engineering and Construction*.

Space Syntax Analysis *Space syntax* analysis methods[13] are a set of tools that allow the structure of an architectural object to be considered in terms of its social role. This means observing how and through what interactions the structure of a building influences the human use of a space. This allows architects and designers to optimise the connections between spaces, as well as the arrangement of doors and passages, to make the pathways and connections within a building more effective in their use of space and more user-friendly.

Visibility Graph Analysis By analysing *visibility graphs*[14] (figure 2.12), conclusions can be drawn about spatial perception at the human level. In this way, certain aspects such as privacy, openness towards public spaces and accessibility can be analysed and optimised in order to provide the most appropriate spatial perception. The *Visibility Graph Analysis (VGA)* method is used to create visibility maps that assign values to the entire surface of the objects being analysed, providing information on how observable they are from all other points in the space.

Graph Clustering Unlike the previous methods, *graph clustering* (figure 2.13) is an abstract form of analysis in architectural applications. Once a network has been created based on geometric or topological concepts, it can be divided into individual subgroups by observing the *clustering coefficient* of the nodes. This allows the designer to abstract

[12]Schultz and Bhatt 2011, “Toward accessing spatial structure from building information models”.

[13]Hillier and Hanson 1989, *The social logic of space.*

[14]J. H. Lee, Ostwald, and H. Lee 2017, “Measuring the spatial and social characteristics of the architectural plans of aged care facilities”.

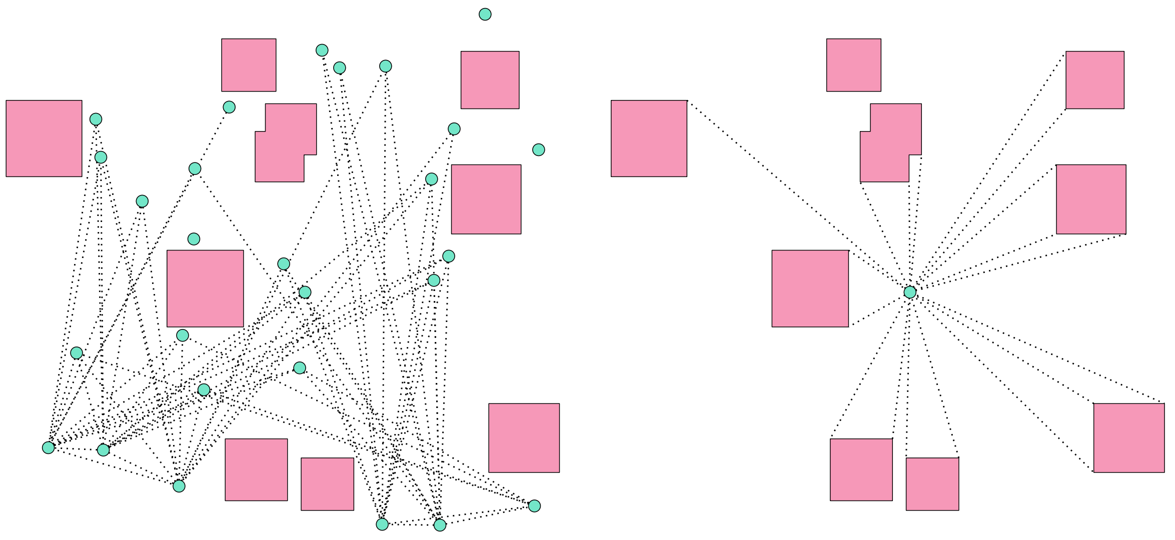

(a) Visibility Between Nodes (b) Visibility with Obstacles

Figure 2.12: Visibility Graphs

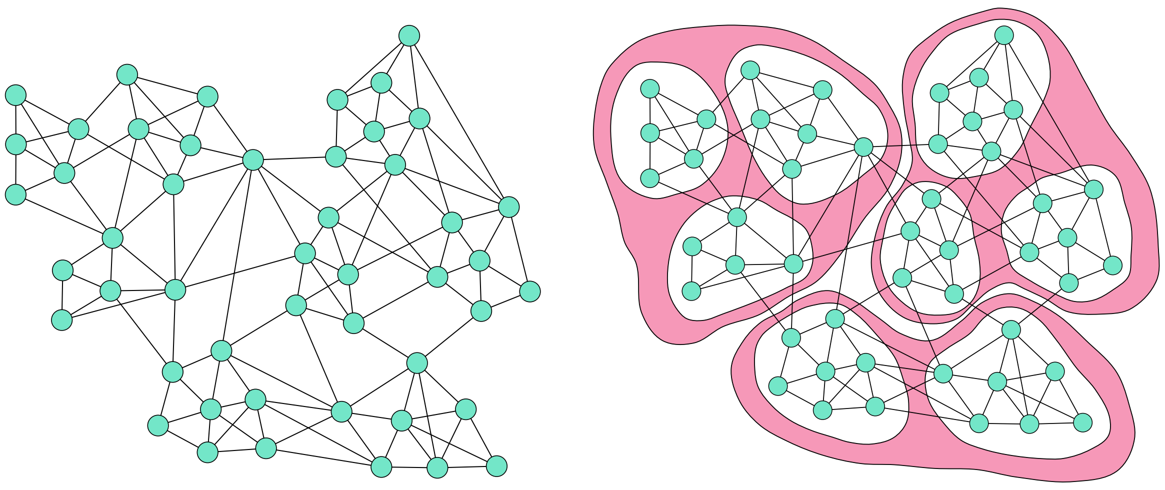

(a) Network (b) Clustering

Figure 2.13: Graph Clustering

complex projects and environments and understand their interrelationships as well as the distinctions between them in order to make specific design decisions in a meaningful way. This method unfolds its full potential when dealing with complex buildings or urban structures.

Multiscale Graph Analysis This method allows the analysis of a system represented by points and edges at multiple levels or scales. By being able to analyse at multiple scales, significant insights can be obtained about the interactions between individual elements of different layers, allowing for the design of more effective and, above all, more integrative design concepts. A meticulous elaboration of the graph structure to be analysed is necessary to successfully represent this interaction between the individual subsystems.

Congestion Analysis *Congestion analysis* methods originate from urban applications, where the risk of congestion (figure 2.14) is calculated as a function of the geometric

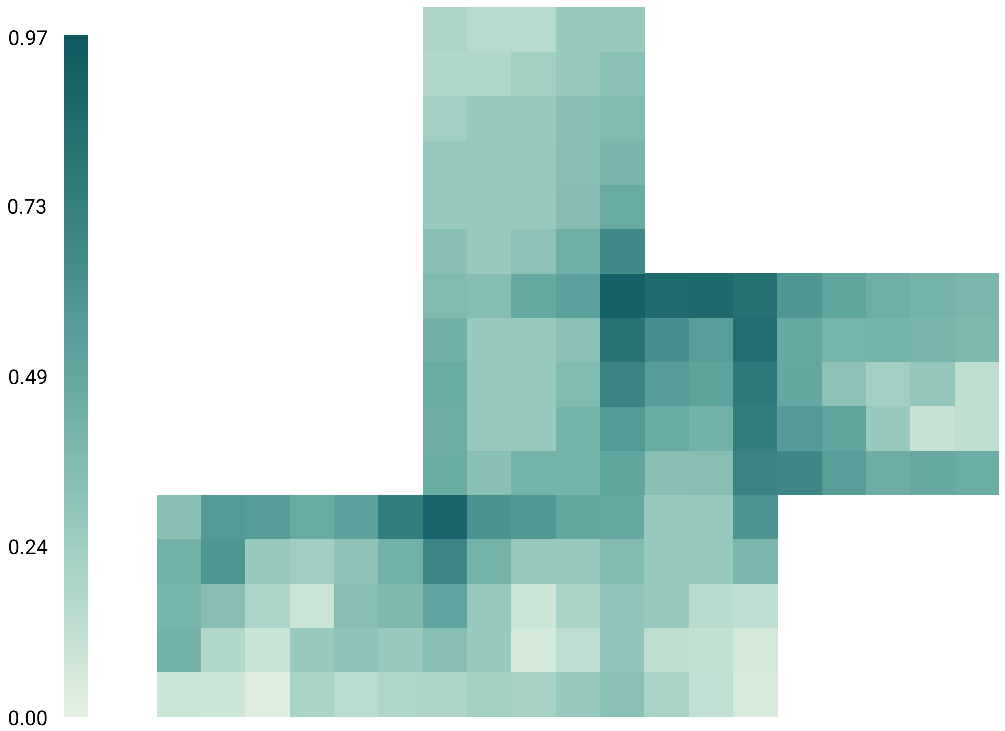
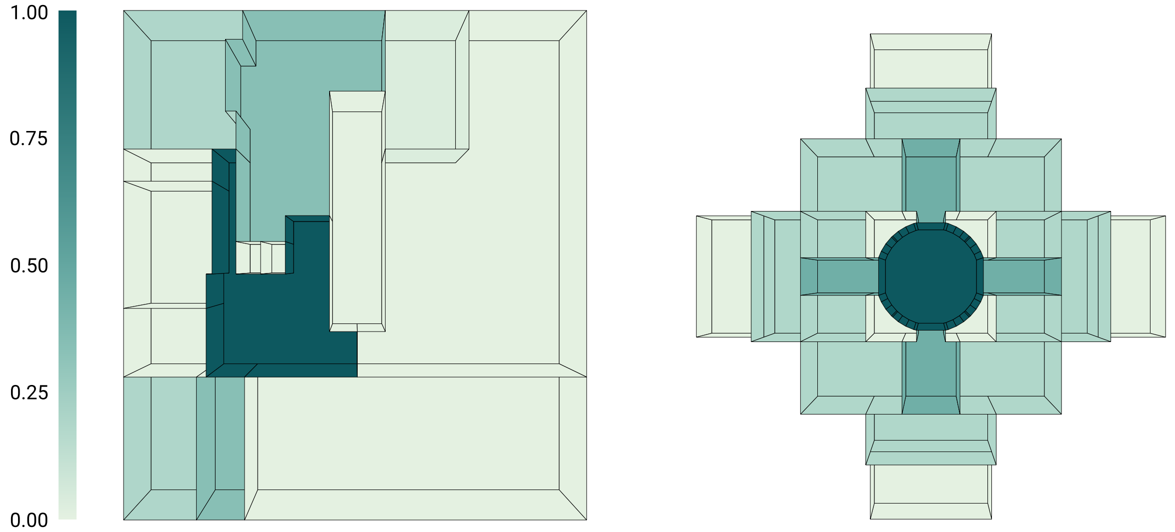


Figure 2.14: Congestion of Floor Plan

properties of the infrastructure network. However, this method of analysis has also found useful applications in building architecture, where the set of possible paths from one point to an opposite point is computed. Values are thus calculated for the entirety of the plan being analysed, which can provide information about the overlap of these paths and describe the risk of congestion within the building at all points. This method therefore represents an essential tool that can lead to the optimisation of circulation within a building through morphological changes.

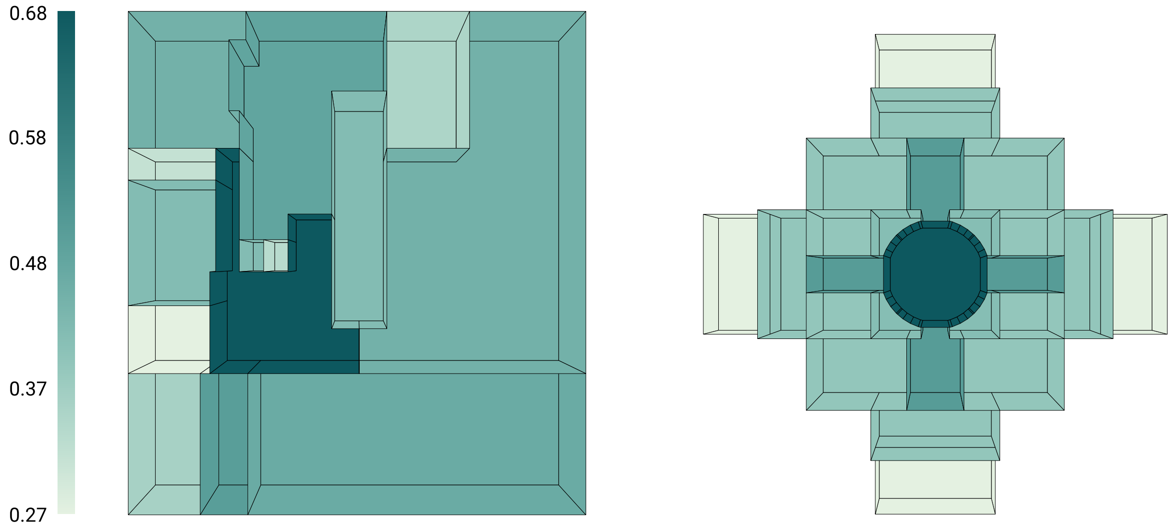
Centrality Analysis Using graph-based representations of complex systems and geometric or topological structures, the *centrality* of individual elements within the system can be inferred. By comparing and summing the number of edges of each node, conclusions about the degree of centrality can be drawn. In addition, examining properties that influence centrality, such as the *node degree* along certain paths, provides insight into the hierarchy and structure of the system. A practical application of this method of analysis is to observe the *axial map* of a building complex to verify whether particular rooms meet or exceed the desired level of centrality. There are two main types of centrality measures: *closeness* (figure 2.16) and *betweeness* (figure 2.15). While the former quantifies how close each node is to all other nodes in the network, the latter measures the degree to which a node is located on the shortest paths between pairs of other nodes in the graph structure.

Frequent Pattern Analysis Another method of graph analysis more commonly used in urbanism is called *frequent pattern mining*, which uses specific algorithms to identify repeating patterns within the graph network, and can be used to examine certain urban constellations for possible problems or even advantages. In architectural applications, this method can also be used to classify complex building structures, for example to identify all the vertical staircases in a building complex or to categorise the individual corridors.



(a) Villa Savoye (b) La Rotonda

Figure 2.15: Betweeness Centrality



(a) Villa Savoye (b) La Rotonda

Figure 2.16: Closeness Centrality

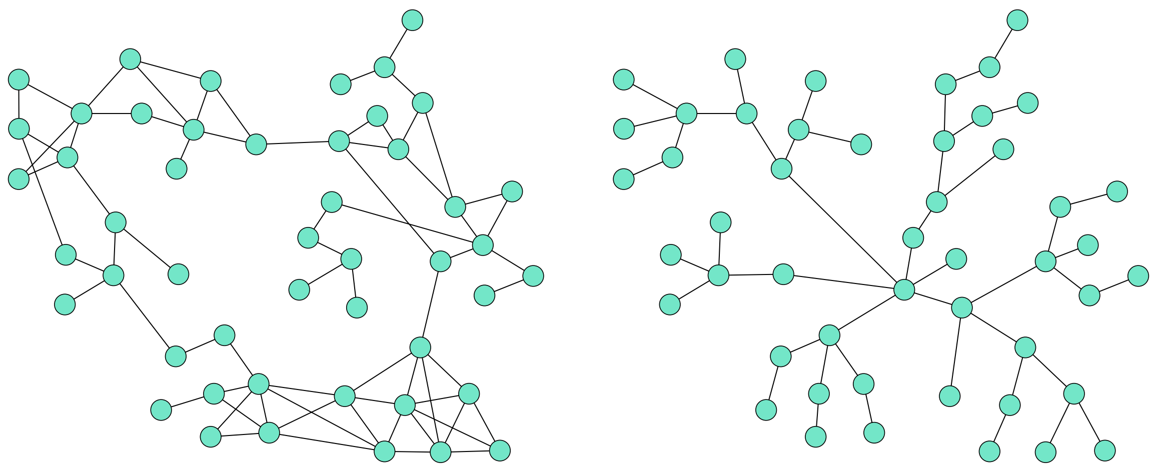

(a) Circle (b) Star

Figure 2.17: Graph Morphologies

Connectivity Analysis The most important information conveyed by graphs is the connectivity of the nodes. In fact, the existence of an edge between individual elements in a network effectively indicates a relationship between them. It is therefore not surprising that a common method of graph analysis in both urbanism and architecture is *connectivity analysis*[15]. This method involves taking individual elements of the system as starting points, trying to analyse their connection to other elements, and ultimately understanding the full interconnectedness of objects and making design proposals accordingly.

Community Detection Similar to graph clustering, *community detection* is used to identify groups of elements in a system that share similar properties or characteristics. However, this method looks at both the information carried by each node and the edge properties of each link to identify individual groups. In this respect, this method can be used by the designer to understand existing systems, such as ecological systems, or to visualise the grouping of building elements in a designed building based on their attributes and connections to each other.

Morphological Analysis In the example of *morphological analysis* of graph networks, the general formal aspects (figure 2.17) of the network are considered. Depending on the level of abstraction of the represented data, information about the morphology of the structure can be obtained. In an architectural application, for example, this allows the identification of the ground relationship[16] of a particular building or other formal aspects such as strong elongations or circular structures on a geometric-formal level.

Random Walks A graph entity can be tested for possible design related issues by applying the *random walk* method, which consists of starting iteratively or in parallel at randomly chosen nodes of the network and walking along an equally randomly chosen path. This initially purely *stochastic* method makes it possible to reveal paths of movement that might have remained unknown to the designer without the use of this method.

[15]Dawes, Ostwald, and J. H. Lee 2021, "Examining control, centrality and flexibility in Palladio's villa plans using space syntax measurements".

[16]Alymani, Mujica, et al. 2023, "Classifying building and ground relationships using unsupervised graph-level representation learning".

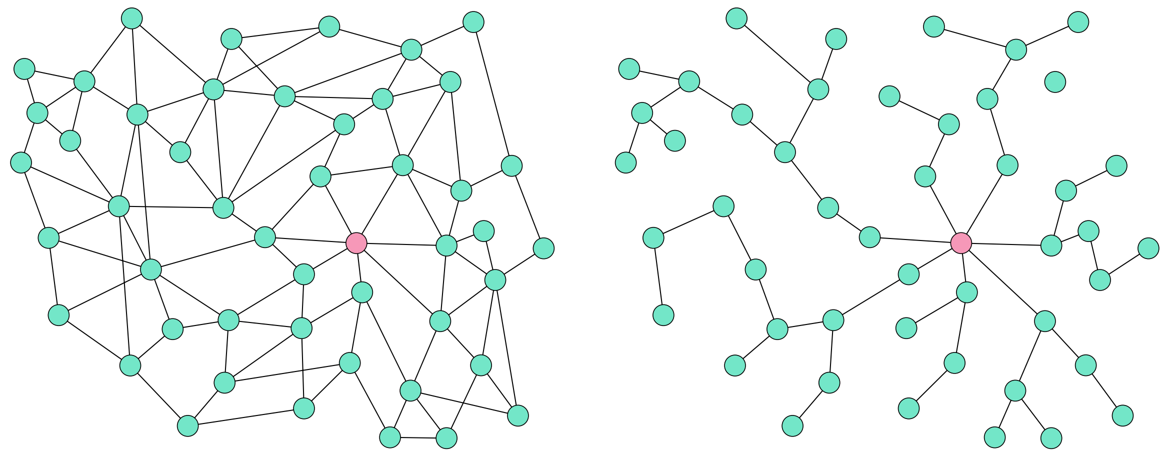

(a) Base Graph (b) Minimum Spanning Tree

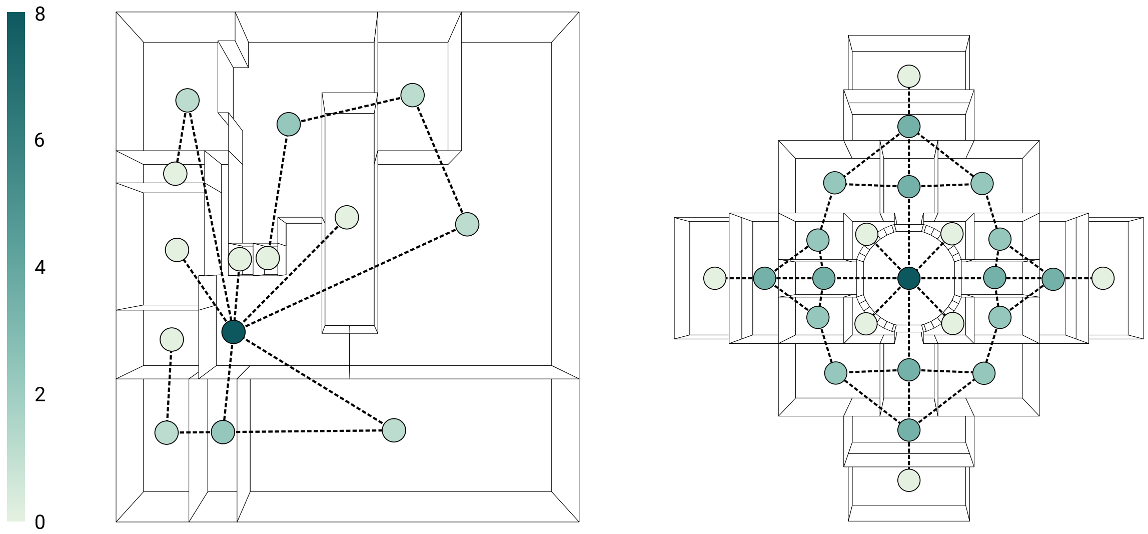
Figure 2.18: MST Computation

A concrete application could be an automatic analysis of the architectural usability of the designed object, which can reveal unwanted dead ends or missing connections between rooms.

Minimum Spanning Trees An elementary method of mathematical graph theory is to create a subgraph of the original graph such that it connects all vertices. The particularity of this subgraph, however, is that each pair of nodes of this graph is connected by only a single edge, thus representing the mathematical concept of a tree. The formation of this *spanning tree* is fundamental to a number of graph analysis methods, including the computation of a *Minimum spanning tree*, which describes the spanning tree of a graph that has the lowest total edge weight (figure 2.18). For unweighted graphs, this is equivalent to the lowest number of edges. It becomes apparent from this definition that the application of such minimal spanning trees in a complex building can provide essential information about the shortest overall connection of all represented elements on a two- or even three-dimensional level.

Network Flow Analysis Originating in engineering and computer science, *network flow analysis* can provide important information about the performance of a defined oriented system where a particular capacity or resource is to be optimised. It involves the construction of a directed graph with a starting point and one or more end points. The individual edges and nodes can receive information about a maximum allowable quantity of the flowing medium, representing the individual constraining instances of the flow network. This method is most commonly used to test the performance and capacity of systems such as plumbing, power grids, sewer systems or even ventilation systems. However, the flowing medium can also consist of crowds and can therefore be used to test the capacity of architectural designs, particularly in public buildings such as concert halls, conference centers or hospitals, to accommodate a certain number of visitors.

Node Degree Analysis A relatively intuitive method of graph analysis is to look at the node degree of each node (figure 2.19). This can provide information about the degree of integration of the particular element under consideration into the system at hand, and thus provide conceptual support in the pursuit of a well-connected architectural object.



(a) Villa Savoye (b) La Rotonda

Figure 2.19: Node Degree

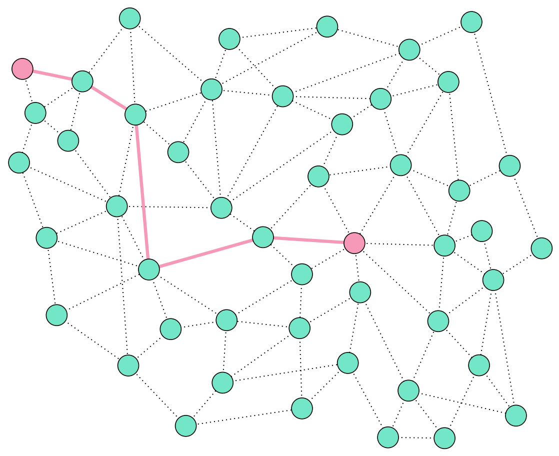
Geometric Graph Theory The methodology of *geometric graph theory* is concerned with the analysis of spatial networks which, unlike *non-euclidean* graphs, describe a geometric representation of objects represented in graph form. Thus, edges in these geometric graphs describe exact metric distances between individual vertices, and the individual vertices refer to exact two- or three-dimensional *cartesian coordinates*. In an architectural application, this allows a purely geometric view of the represented system, as opposed to the analysis of topological or abstract programmatic information. This enables the designer to calculate the total distance in metric values between two nodes or, for example, the exact length of the longest paths from certain rooms in a building complex to the nearest emergency exit.

Shortest Paths Analysis The *shortest path* problem (figure 2.20) in graph theory describes the search for the shortest connection between any two arbitrarily chosen vertices of a graph. In other words, it is about finding the shortest sequence of different vertices connected by edges, starting with an arbitrary node of the graph and ending with another arbitrary node. There are several different algorithms for finding this shortest path with different computational speeds. As a method of analysis in architecture, the search for the shortest path is an essential concept to topologically analyse the architectural object and to identify widely separated spaces or to strategically position important elements such as emergency exits or lifts.

2.1.2.3 Graph Machine Learning

In the field of architecture, the application of machine learning has rapidly gained importance in the recent years[17]. These computer science methods make it possible to analyse large amounts of data and use established algorithms to identify patterns or make predictions that are relevant to the design and planning of architectural objects. Machine learning can thus help to optimise architectural projects and improve the performance of these objects in a variety of ways, for example by analysing the behaviour of spatialities

[17]Belém, Luis Santos, and Leitão 2019, "On the Impact of Machine Learning: Architecture without Architects".


(a) Shortest Path I

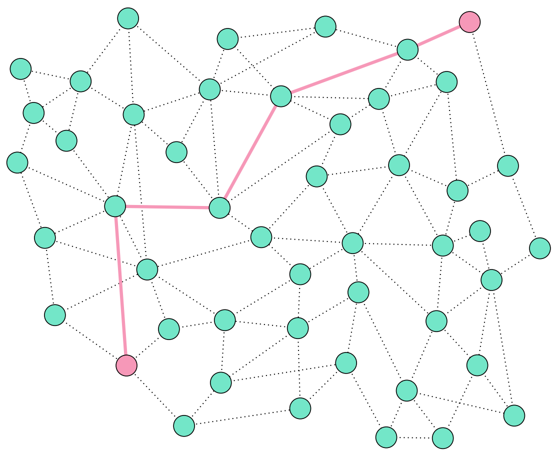

(b) Shortest Path II

Figure 2.20: Shortest Path Analysis

and materials under different conditions. For instance, machine learning can be used to optimise certain parameters or dimensions that improve a building's energy requirements or load-bearing capacity. Similarly, data science methods can be of great benefit to architects in analysing user behaviour and designing spaces. However, there are also some challenges in applying machine learning to architecture[18], such as the need for high quality, evaluated and maintained datasets, and the sometimes high complexity of the algorithms.

In principle, there are two main types of machine learning tasks, called *classification* and *regression*. Classification involves dividing data into predefined classes or categories, while regression involves making continuous predictions based on numerical values. For example, categorising building elements into windows, doors and walls would be a classification task, while predicting the number of people in an office building at a given time would be a regression problem. In addition, there are different types of training procedures in data science, which come with their own set of benefits and challenges. The two main methods, *supervised* and *unsupervised learning*, are basic categories of machine learning that differ in the type of data provided. In supervised learning, algorithms are trained on labelled data, where the desired value or class is known in advance. In unsupervised learning, the models are trained on unlabelled data where no clear answers are given. There is also *semi-supervised learning*, a hybrid of supervised and unsupervised learning, where both labelled and unlabelled data are used for training.

Graph-based machine learning is the term used to describe the relatively new branch of data science that relies on datasets in graphical form and therefore involves a number of special procedures. Training machine learning on graph data, as opposed to the more common tabular data, has the significant advantage that the trained models receive correlations and relationships as input data, making it possible to identify graph-specific qualities or problem sets. In the context of graph-based machine learning, both classification and regression tasks can arise. For example, the task may be to classify nodes in a graph based on certain features, or to make predictions about the weights of certain edges in a weighted graph. Similarly, the training methods can vary, so that both supervised

[18]Pena et al. 2021, "Artificial intelligence applied to conceptual design. A review of its use in architecture".

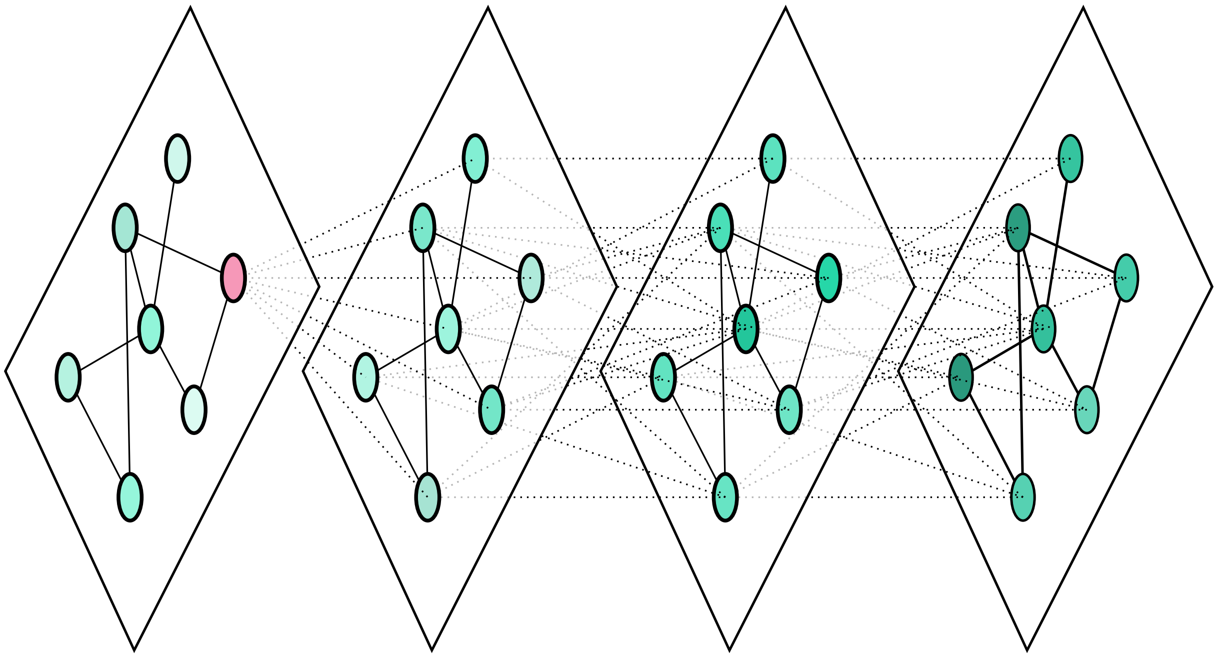


Figure 2.21: Graph Neural Network Scheme

and unsupervised learning can be applied in the graph context. For example, the goal can be to learn the classification of nodes in a graph by training on labelled data, or the objective may be to discover hidden patterns or commonalities in the graph by clustering or dimension reduction without the need for labelled data. Accordingly, the hybrid of the two training variants, semi-supervised learning, can be used to deal with a limited amount of labelled graph data as well as a large number of unlabelled graphs.

The discipline of graph-based machine learning in data science has seen a considerable amount of new research[19] in recent years and is in constant flux. Nevertheless, four main groups of *neural network* methods have emerged as useful *deep learning* approaches.

Graph Neural Networks A broad family of machine learning models that focus on graph data processing are *Graph Neural Networks (GNNs)*. By propagating information within the graph and aggregating features from neighbouring nodes, they produce more meaningful representations of each vertex. Through this iterative process, GNNs can perform tasks such as *node classification, link prediction*, and *graph-level prediction*. GNNs have proven to be extremely powerful in a number of domains, including architecture, for a variety of tasks. There are several variants of GNNs, each with its own specific techniques and applications. The principal groups are presented below.

Graph Convolutional Networks GNNs architectures called *Graph Convolutional Networks (GCNs)* have been developed specifically for processing data represented in graph structures. They perform *convolutions* on the graph by aggregating local information from adjacent nodes to create new representations or features that capture both the local structure and the properties of the nodes (figure 2.21). The convolution process uses both the features of the nodes and the topology of the graph to capture complex patterns and relationships. These networks can be used for tasks such as *graph clustering*, *node and edge classification* or *edge prediction*. By harnessing the topological information

[19]Velikovi 2023, "Everything is Connected: Graph Neural Networks".

within graph structures, GCNs can outperform conventional GNNs.

Graph Attention Networks A special type of GNNs called *Graph Attention Networks (GATs)* introduces an attention mechanism to evaluate the relative importance of neighbouring nodes in aggregating information, such as evaluating the influence of different architectural components within a building design. By using attention, GATs can assign different weights to different nodes in the graph, focusing on the most relevant nodes in a given context. This selective attention not only improves the efficiency of the model, but also allows it to capture more fine-grained, contextual relationships within the graph.

Graph Autoencoder Unsupervised learning models called *Graph Autoencoder (GAEs)* encode graph data, such as architectural layouts, into a compact *latent representation* that can be subsequently decoded into the original graph structure. GAEs consist of two primary Graph Neural Networks: an *encoder*, which transforms the input graph into a low-dimensional representation, and a *decoder*, which reconstructs the graph from the compressed representation. This latent representation can be used for tasks such as dimension reduction or graph generation. GAEs are able to efficiently learn expressive representations and provide insights into the underlying patterns and relationships within the data.

Graph Recurrent Neural Networks *Graph Recurrent Neural Networks (GRNNs)* are another class of GNNs that incorporate *Recurrent Neural Networks (RNNs)* components to model dynamic, *sequential data* on graphs, such as evolving architectural designs or construction processes. GRNNs are particularly useful for problems where the graph structure or node attributes change over time, such as in temporal networks or dynamic structures. By combining the expressiveness of GNNs with the sequential modelling capabilities of RNNs, GRNNs can capture both spatial and temporal dependencies within the data, enabling accurate prediction and in-depth analysis of temporally evolving graph structures.

2.2 Topology

Architectural thinking is usually closely linked to the understanding and analysis of geometric methods, since dimensions, surfaces and angles constitute an elementary part of the discipline. However, if we consider only the space generated by surfaces and boundaries, the metric components can be disregarded for the time being. In fact, the space defined by the domain of a sphere has the same properties, regardless of size or distortion transformations, as long as we do not consider the interior as a metric unit of volume.

This becomes interesting as soon as we compare the interior of a sphere with the interior of a cube. In both cases it is a space enclosed by a continuous surface without openings to the surrounding environment. To illustrate this theoretical understanding of space, the example of a doughnut and a coffee cup is often used, since although the two shapes have little in common at first sight, they are indistinguishable on a purely spatial-formal level. In fact, a coffee cup can be transformed into a doughnut by certain matrix transformations without breaking or joining the faces that make up the object. This approach is called *topology* because it deals with the study, *logos*, of spaces, *topos*. In other words, topology is the branch of mathematics that deals with the analysis of solids undergoing continuous

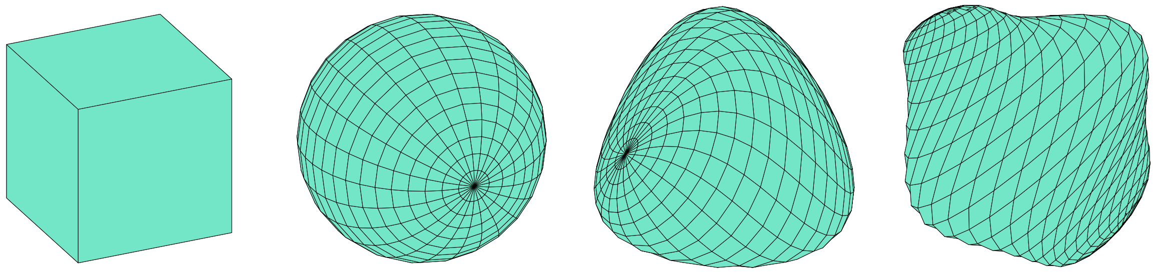


Figure 2.22: Topological Deformations of a Cube

deformation without being opened, closed, torn, joined or self-overlapping[20] (figure 2.22).

Topology is based on a formal definition of the space concept, in which space is a set of points associated with certain properties and relations. The basic concept is to consider open and closed sets of points in space, where the dimensionality of the space can be unbounded. An *open set* is a set in which each point has a certain radius around it in which there are no other points. A *closed set* is a set that contains all its boundary values, which are the boundaries between the set and the outside world. Based on these basic concepts, other terminologies such as *continuous functions*, *homotopy* and *cohomology* have been developed. A continuous function is a mapping between two spaces that preserves the structure of the space. A homotopy is a continuous transformation between two functions that change the space in an equivalent way. Cohomology, on the other hand, deals with the study of geometric objects by analysing their characteristic classes.

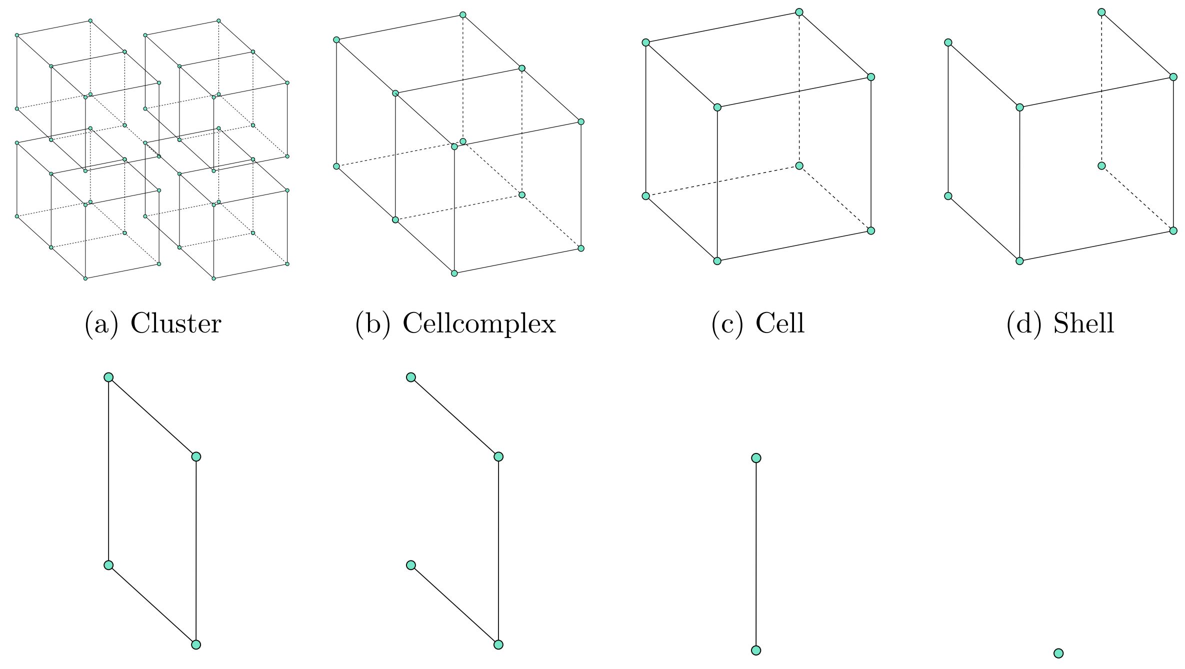
Topological studies are divided into three main areas[21]: *algebraic topology*, *differential topology* and *geometric topology*. Algebraic topology studies space by means of algebraic methods and focuses on the analysis of homotopy groups, cohomology groups and other algebraic instances. Differential topology, on the other hand, studies space by analysing its *smooth functions* and *differentiable structures*. Geometric topology studies space using geometric methods and investigates questions of shape and structure, such as the study of *manifolds* and the curvature of spaces.

Through the concept of topology, new tools for architectural consideration of spatial structuring become apparent and their formal application via transformation and analysis in the design process demonstrates their usefulness.

2.2.1 Topological Analysis

Topological methods of analysis are integral to the discipline of architecture, although they are often not clearly recognisable as such to the observer or even the designer. In fact, architecture can in a sense be considered part of the topological discipline, since architecture, according to certain definitions, is about the elaborate division and enclosure of the three-dimensional space that surrounds us, which is achieved by connecting and assembling various constructive components. In this respect, topological methods of analysis in architecture offer a range of possibilities for investigating the properties and relationships between different elements in an architectural system. This involves the analysis of spatial or elemental structures that cannot be reduced to geometric properties

[20]Kantor 2005, "A tale of bridges: topology and architecture".
[21]Kelley 1955, *General topology*.


(a) Cluster (b) Cellcomplex (c) Cell (d) Shell

(e) Face (f) Wire (g) Edge (h) Vertex

Figure 2.23: Topologic Components

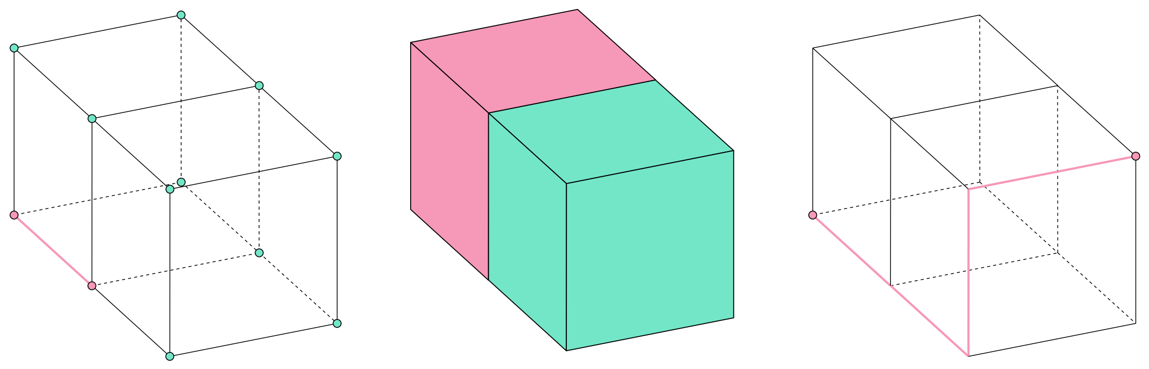
such as length, width or height, but rather the topology of space, in other words how the elements are connected to each other or how the respective forms are constructed. In particular, the spatial language tool *Topologic*[22] and its Python API *TopologicPy*, as well as its analytical capabilities, are considered here as a concrete demonstration of topological methods on architectural objects and elements. The software tool allows n-dimensional bodies to be decomposed into their constituent parts and higher-dimensional structures to be created from n-dimensional basic parts, thus enabling a topological understanding of the formation of boundaries and space.

The hierarchical-categorical structure of elements into topological categories (figure 2.23) such as *vertex*, *edge*, *wire*, *face*, *shell*, *cell*, *cellcomplex* and *cluster* allows formal transformations without changing the basic topology of the geometric bodies or even performing specific topological deformations. The functionality is based on the decomposition of shapes into *non-manifolds*, so that, for example, a cell body is composed of three or more contiguous closed faces, which become a two-dimensional shell object once this shape is opened through a whole in the boundary of the body.

Using this setup, three main topological relationships can be queried[23] in an object-specific manner. The *hierarchical relationship* (figure 2.24a) of topological elements refers to the composition of objects by their subelements. For example, the edge of a cell object is a *subtopology* of that same cell object and is referenced as such in the data structure. This also works the other way round: if three vertex elements are connected by edges, their *supertopology* is a wire object.

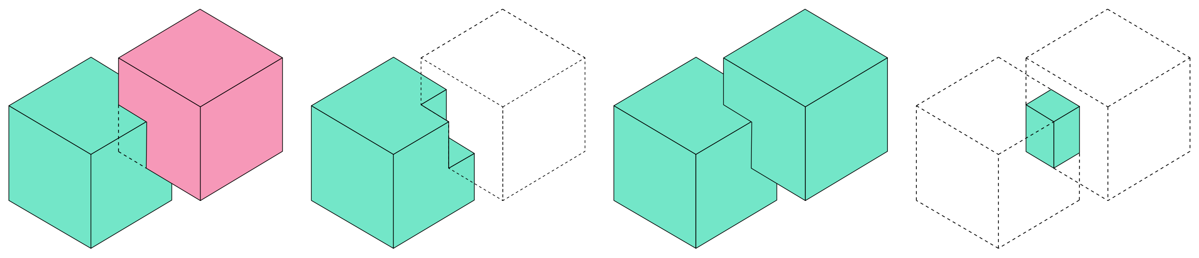
Another type of relationship, which can be easily queried thanks to the structure of the *Topologic* library, is the *lateral relationship* between individual elements (figure 2.24b).

[22] Aish et al. 2018, “Topologic: tools to explore architectural topology”.
[23] Jabi, Aish, et al. 2018, “Topologic: A toolkit for spatial and topological modelling”.



(a) Hierarchical (b) Lateral (c) Connection

Figure 2.24: Relation Query Methods



(a) Base Configuration (b) Cut (c) Union (d) Intersection

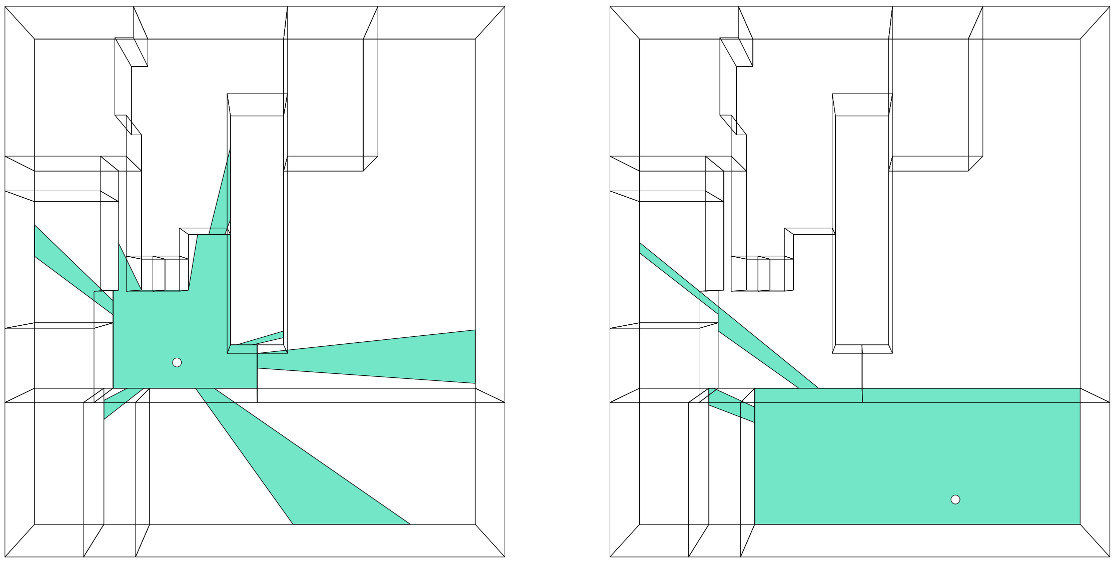
Figure 2.25: Boolean Operations

Here, the relation of two n-dimensional elements to each other is queried by looking at the subtopologies they share. For example, a three-dimensional apartment model with rooms and corridors as cellcomplex objects can be seen as a collection of spaces, each of which shares at least four edges and correspondingly at least four vertices at the one-dimensional level.

The third relational property is the *connectivity* of the individual elements (figure 2.24c), which can be queried through defined methods by describing the topological connection of element A to element B. The connection between the two elements is described in terms of the topological connection between the two elements and can pass along edges, through faces or volumes depending on the defined dimensionality. The connection created in this way can be represented by a graph structure, thus allowing the application of graphical analysis methods.

Another fundamental feature of the software described is the implementation of *Boolean operations* that can be applied to any n-dimensional pairs of elements, thus creating new topologies. The main Boolean operations are *Union* (figure 2.25c), *Difference* (figure 2.25b), *Intersection* (figure 2.25d), *Symmetric Difference*, *Merge*, *Slice*, *Impose* and *Imprint*, where the resulting objects are always non-manifold.

The main advantage of using non-manifold geometries is that elements can contain subelements that do not necessarily form a closed body. Thus it is possible for a face to contain an *aperture* which could represent a window or a door, so that the face becomes the domain of the aperture and can accordingly be examined by the topological methods mentioned.



(a) Isovist Point I (b) Isovist Point II

Figure 2.26: Isovist Analysis of Villa Savoye

2.2.2 Space Syntax

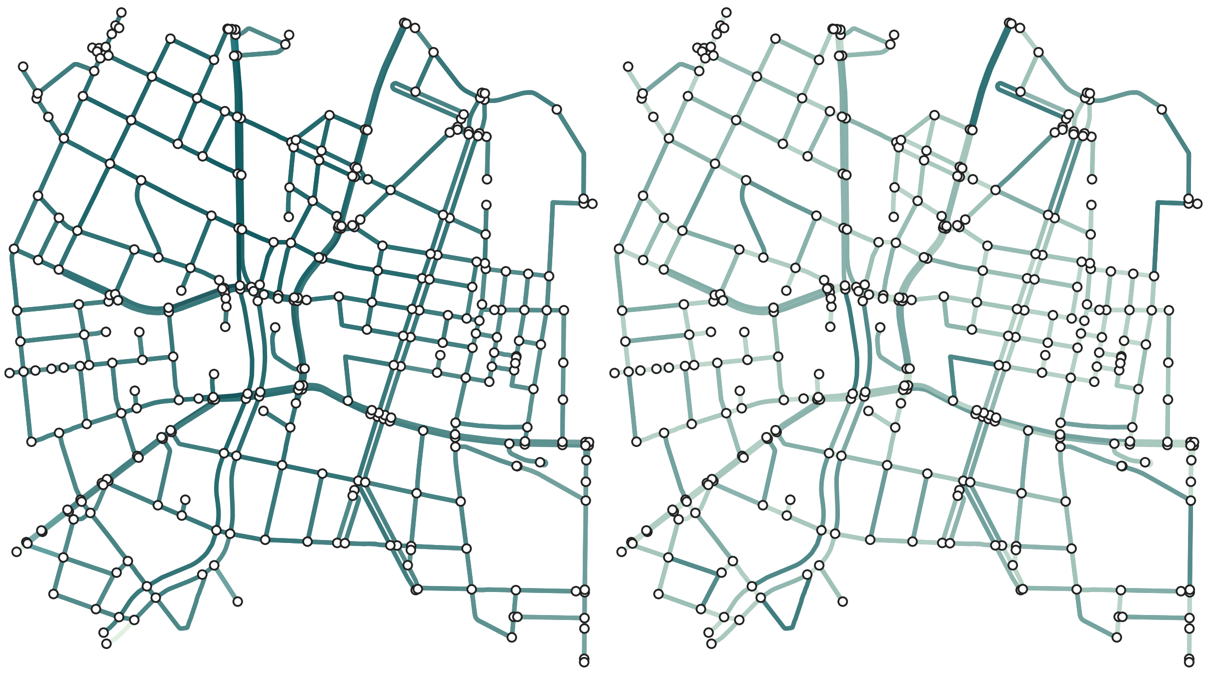
When first introduced[24], the study of *space syntax* was seen as a parallel current to the formal preoccupations of the architectural profession because, as a theory, it is primarily concerned with understanding spatial relationships and their social effects on people. It thus serves to metrically analyse the social performance of architectural objects[25] and is primarily concerned with the socio-spatial organisation of buildings. Particular attention is paid to the local status of spatiality and its organisational position between private and public space. However, the theory of spatial syntax is by no means limited to buildings, as it can develop its full potential in larger networks such as urban structures. Therefore, this theory is concerned with studying the interaction and behaviour of social and spatial structures in order to achieve the best possible combination and interaction[26]. As a discipline, it combines topological, geometric and social information to test specific spatial configurations through analysis and simulation to evaluate their performance before they are built. Three principal methods of analysis exist in spatial syntax theory, which can describe the relationships and interactions of spatialities at the urban or building scale through cartographic and graphical representation.

Isovist Analysis An *isovist analysis* (figure 2.26) describes a method that allows the visualisation of all elements of a spatial body that are visible from a given point in space. The creation of an isovist graph is based on the tracing of lines of sight and fields of vision, and makes it possible to understand and optimise the effect of light, space and materials on the perception of spaces.

[24]Hillier and Hanson 1989, *The social logic of space*.

[25]Nourian, Rezvani, and Sariyildiz 2013, "Designing with Space Syntax: A configurative approach to architectural layout, proposing a computational methodology".

[26]Franz, Mallot, and Wiener 2005, "Graph-based models of space in architecture and cognitive science: A comparative analysis".



(a) Closeness Centrality (b) Path Length

Figure 2.27: Axial Analysis of Street Network

Axial Analysis The creation of axial maps (figure 2.28) is another fundamental tool from the field of spatial syntax analysis and consists in representing spatial structures on an architectural or urban level through a network structure, thus making spatial connections clear. It forms the basis for graph-theoretic analysis methods that build on its framework, but certain conclusions can already be drawn without deeper analysis. For example, the spatial organisation of a building can be better understood and communicated by looking at the connections between rooms of an architectural object and the course of the generated axes.

Convex Space The *convex space* method (figure 2.29) considers architectural space as a collection of *convex* surfaces or volumes separated by walls or other solid elements. The individual convex bodies describe the number of respective point pairs that can be connected by an edge without intersecting or leaving the boundaries of the convex body. The analysis of convex space is therefore based on looking at the space created by the individual shapes and observing how these areas are connected. This method of analysis provides essential spatial information, as by placing an individual in such a convex space, the totality of the environment can be accessed by the observer, which strongly influences the cognitive perception of such spaces.

With the increased adaptation[27] of space syntax in architecture and urbanism, other concepts have emerged that allow for more concrete spatial analyses, a more accurate understanding of social structures and their interaction with the environment, and accessibility analyses of designed elements.

Depth Analysis *Depth analysis* (figure 2.30) in the context of spatial syntax theory is concerned with the topological depth of urban structures or architectural objects. The

[27]Y. Li et al. 2009, *Design with space syntax analysis based on building information model.*

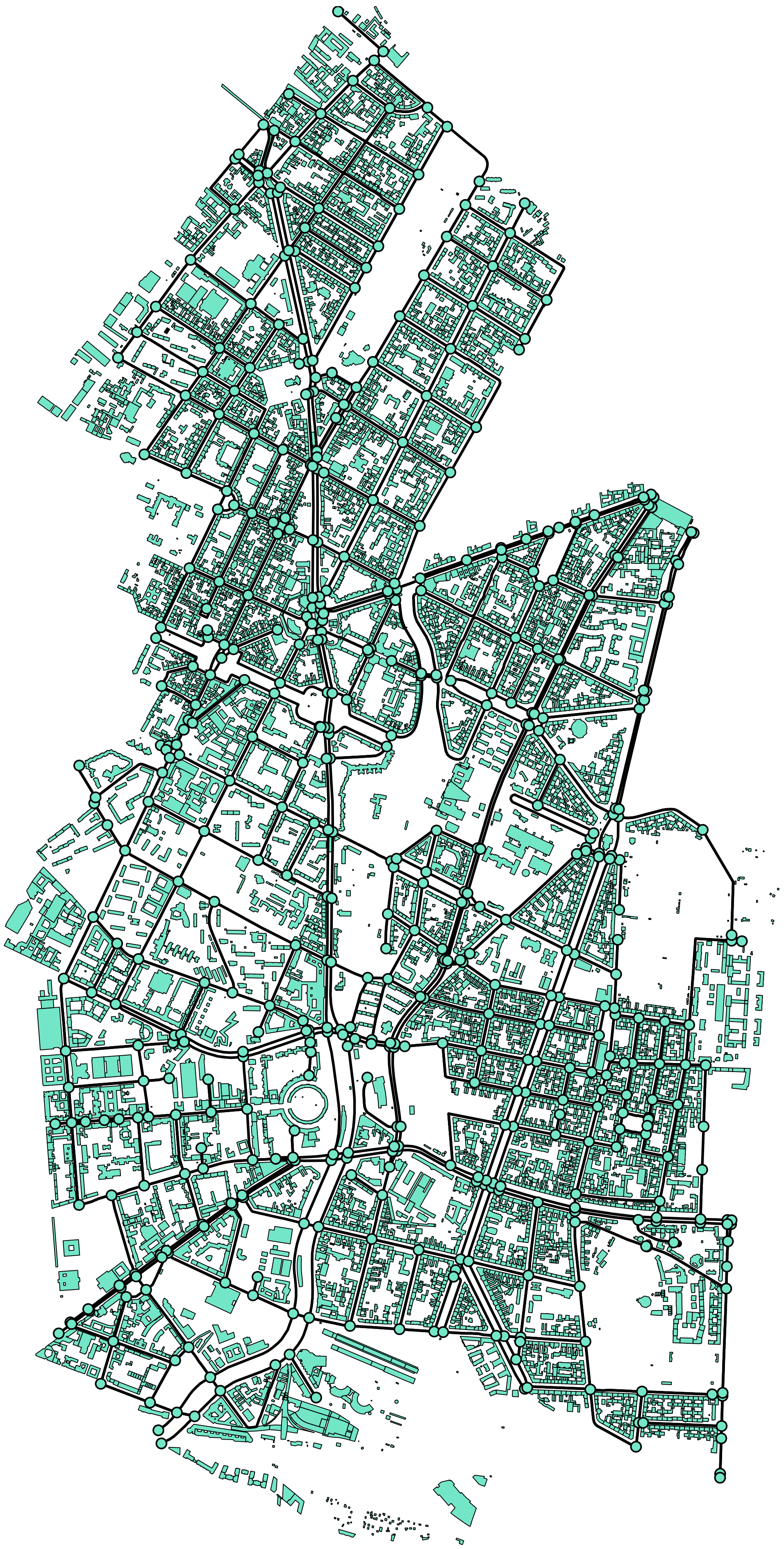


Figure 2.28: Axial Graph of the Kreuzberg District

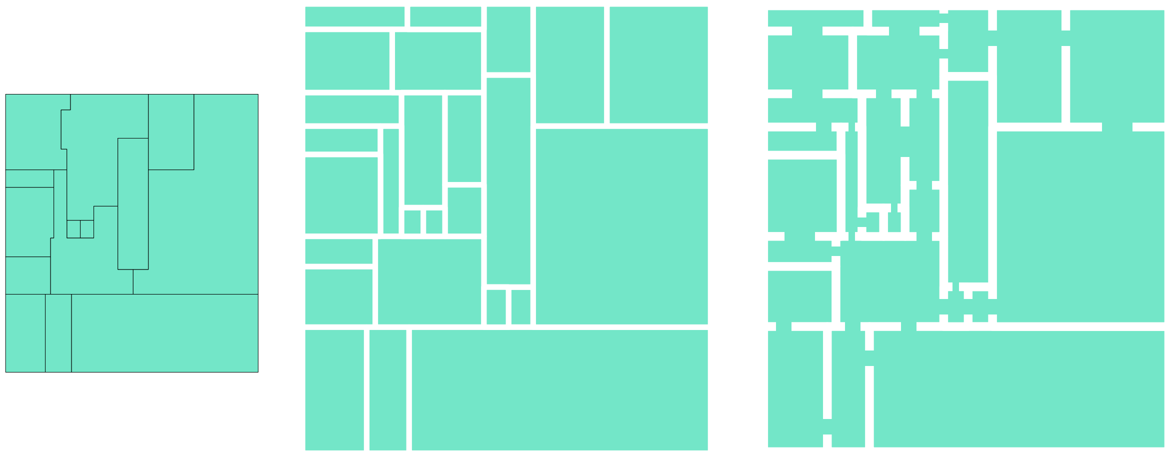
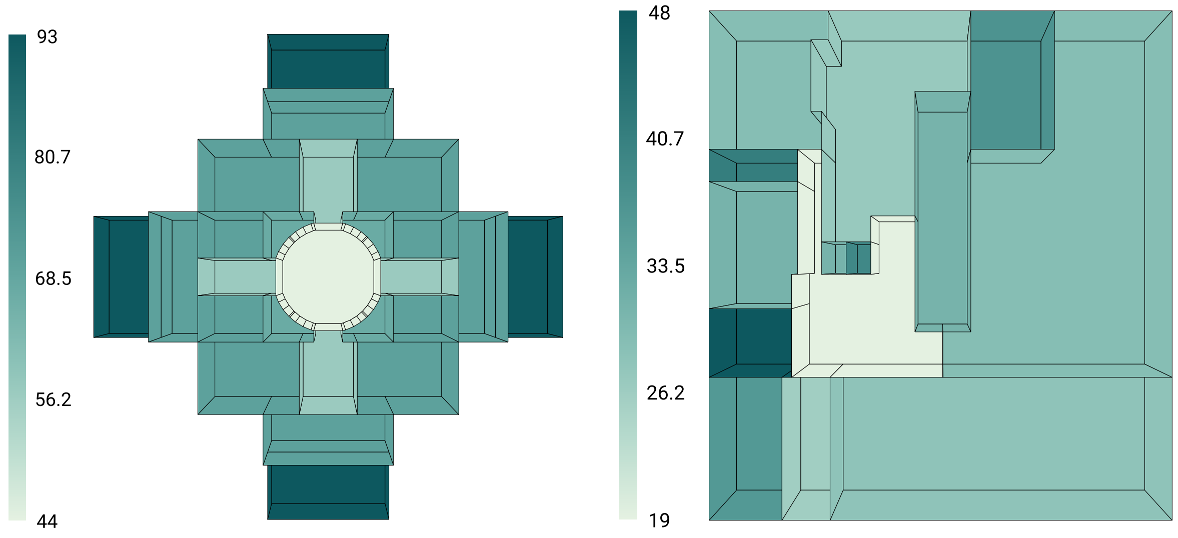

(a) Villa Savoye (b) Convex Space (c) Convex Connection

Figure 2.29: Convex Space Analysis



(a) La Rotonda (b) Villa Savoye

Figure 2.30: Depth Map Analysis

depth is described by the number of elements that must be traversed to reach a given point in the described system. In this framework, it is therefore a topological distance as opposed to a geometric one. The method is usually applied to topological graphs, but can also be used to create a general depth map of the body to be described, in which case it can also be used for geometric depth evaluations.

Space Integration Based on the creation of an axial map, the accessibility of certain paths or elements (figure 2.27a) can be detected, calculated and visualised by applying the concept of *space integration*. This method consists in calculating the degree of accessibility at each element of an axial graph by considering how many elements must be crossed to reach the analysed entity. Again, different units of distance can be considered, such as topological distance, metric distance (figure 2.27b), or even angular distance, which considers how many rotations an individual goes through to get from any point to the point or element being investigated.

Patterns The discussion of the role of recurring *patterns* in architecture, introduced by Christopher Alexander[28], has attracted considerable attention in architectural theory and has occurred concurrently with the development of spatial syntax theory. In this respect, it is hardly surprising that his description and meticulous definition of certain patterns, which follow clear and predictable rules, have also found their way into space syntax methods. In this context, these patterns can refer to the spatial structure, the social organisation or the functions of an environment. In particular, the so-called social patterns are considered, which refer to the patterns resulting from the social interaction and behaviour of people in certain spatial contexts. Each social pattern comes with its own characteristics and needs and, once identified, can help the designer to shape the space accordingly.

2.2.3 Shape Grammar

The so-called *shape grammar* (figure 2.31) is a formal methodology for the automated generation of architectural bodies in mainly two- and three-dimensional spaces, and has gained importance in architectural theory since the 1970s[29]. The basic operation of the method is based on the assumption that certain elements can be combined through established rules to generate various architectural objects and their derivatives (figure 2.31c). The theory thus assumes that the formal aspects of architecture follow certain regularities which are defined by a finite number of rules and thus become part of a formal design language. By applying this grammatical syntax to specific architectural forms, a multitude of different variants of a morphologically similar aggregate of bodies can be generated, all following the same generative rules, as long as these rules allow for a certain degree of variation.

As the terminology of the two concepts, shape grammar and space syntax, suggests, their combination is complementary at a theoretical level and, when skilfully applied, allows a qualitative architecture to emerge. The architectural rules needed to make the algorithm work can be quite diverse, but should always start with an initial geometric *ground rule* (figure 2.31a) and be terminated by a *final rule*. Typically, the initial rule is the definition of a simple geometric shape, and subsequent rules (figure 2.31b) are either *transformation rules* or *parametric rules*. Examples of transformation rules are rotation of certain added or initial bodies, translations, reflections, scaling or Boolean transformations. Parametric rules allow a greater relationship to the geometric context of the transformations, as parameters can be defined and varied depending on other subfigures or constellations.

2.2.4 Space Partitioning

For nearly a century, a subfield of architecture and design theory has been concerned with the research and development of algorithms that enable the generation of a two- or three-dimensional spatial layout by partitioning or accumulation that follows desired regularities. This area of research is often referred to as automated floor plan generation[30]. The shape grammar language mentioned above is part of this search for algorithm-based spatial partitioning. In the context of this work, a close look at each of the established and less common methods, as well as their combination, is essential, since a basic geo-

[28]Alexander 1977, *A pattern language: towns, buildings, construction.*

[29]Hong and Economou 2022, "Five criteria for shape grammar interpreters".

[30]Nisztuk and P. B. Myszkowski 2019, "Hybrid evolutionary algorithm applied to automated floor plan generation".

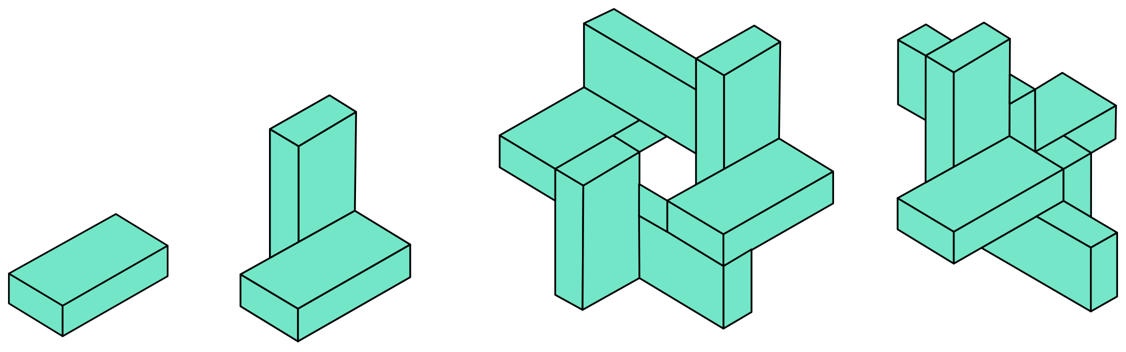

(a) Initial Shape (b) Rule (c) Variations

Figure 2.31: Shape Grammar Process

metric space partitioning is elementary for the generation of an architecturally qualitative dataset, as demonstrated in the chapter 4.

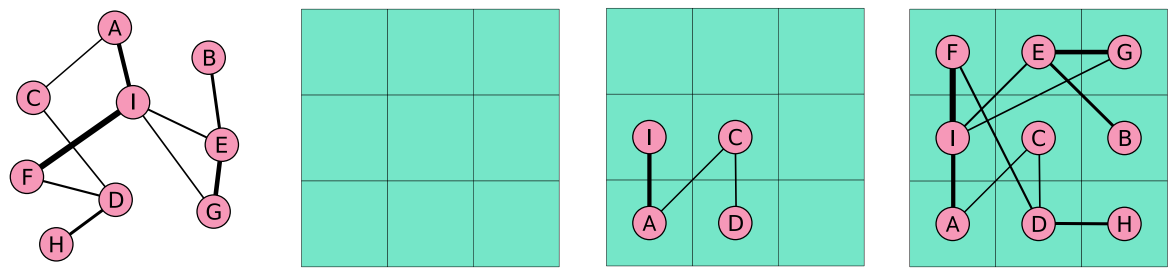
The initial conditions, the input, for the correct functioning of the various methods can be very different and are therefore an elementary part of the consideration, evaluation and comparison of the presented algorithms. The rule-based structure of the shape grammar differs significantly from the simpler geometric algorithms, which require as a starting point a defined two- to three-dimensional body, percentages of space or even a random seed. In addition, there are a number of methods that start with certain geometric bodies with given dimensions and then proceed to combine them in various ways to form an agglomerate.

Grid Planning A relatively simple way to partition a desired space under certain conditions is the grid-based planning method (*Grid Planning*). In this method, the initial surface or body to be partitioned is covered with a uniform or varying grid (figure 2.32b) that is more or less closely meshed with respect to the total area and the desired number of partitions. This method further requires information on the programmatic and topological relationships of the individual units in order to make compositional decisions and compare different arrangements.

In the case of automatic floor planning, the individual rooms to be arranged are assigned different values for their connections to each other, similar to a weighted graph, introduced in figure 2.32a. In order to place the individual spaces in the defined grid in the best possible way, the desired connection between them is respected as much as possible, be it Boolean values in the case of topological relationships or local distance values. This defines the basic framework of the grid-based planning method[31], to which different algorithms for optimised space placement can now be applied.

First, spaces can be placed through an iterative process starting with a randomly selected or predetermined space (figure 2.32c). In the following steps, the rooms are gradually placed according to their relationship priority. This continues until finally all rooms have been placed (figure 2.32d). Although this method has a higher probability of finding an optimal space allocation than a random placement method, there is a risk that certain decisions during the process may prevent an optimal solution from being found.

[31]Lopes et al. 2010, “A constrained growth method for procedural floor plan generation”.



(a) Weighted Graph (b) Grid (c) Iteration (d) Final Placement

Figure 2.32: Grid Planning

The *random swap* algorithm[32] has proven to be a better alternative to the *iterative assignment* method. Starting from a random placement of all rooms, this algorithm swaps two room positions in iterative steps and checks whether the general *fitness function*, which is the sum of all distances, is minimised. This method represents a *greedy algorithm*[33] because it only evaluates per iteration whether the action just performed optimises the overall result, but without having a general overview over several iterations. Using the random swap algorithm, optimised results can be obtained compared to the iterative assignment method. However, the latter is significantly more time-consuming and computationally expensive.

Subdivision While the grid-based planning method locks the final layout to the position and dimensions of the defined grid, a subdivision method can be used to iteratively structure a space by freely placing walls within a defined boundary, similar to the manual architecture method. This family of partitioning methods consists mainly of relatively simple algorithms that differ only in the way the walls are placed. However, they all have as a basic requirement the definition of an initial shape which, in this concrete context, represents the outline of the floor plan to be subdivided.

The first and most intuitive method is to divide the basic body (figure 2.33a) in an arbitrary or defined direction, creating two partial surfaces (figure 2.33b). This process is repeated until the desired number of faces is obtained (figure 2.33c). The basic concept of *recursive subdivision* is a binary tree which has a starting point, the base body, and two child leaves representing the two resulting faces. This is a fast and effective algorithm, taking only linearly more time as the number of spaces increases. However, this method of space division is very limited as it does not take into account any definition of room, size or percentage.

A slightly modified form of this recursive subdivision are the so-called *k-dimensional tree* algorithms[34] (K-D tree algorithms), which subdivide the space by iteratively placing walls in a similar way to the previous algorithm. However, the placement of the intersections is not arbitrary. First, points are placed on or in the two- or three-dimensional body representing the position of each room. These sets of points are then bisected by a wall along their larger extension in the middle, so that the same number of points are in both subspaces. This is repeated iteratively until there is only a single point in each subdivided space. The direction of the wall placed per iteration is variable and is determined by the

[32]Nagy 2021, “AI in space planning”.

[33]Nisztuk and P. B. Myszkowski 2019, “Hybrid evolutionary algorithm applied to automated floor plan generation”.

[34]Knecht and König 2010, “Generating floor plan layouts with kd trees and evolutionary algorithms”.

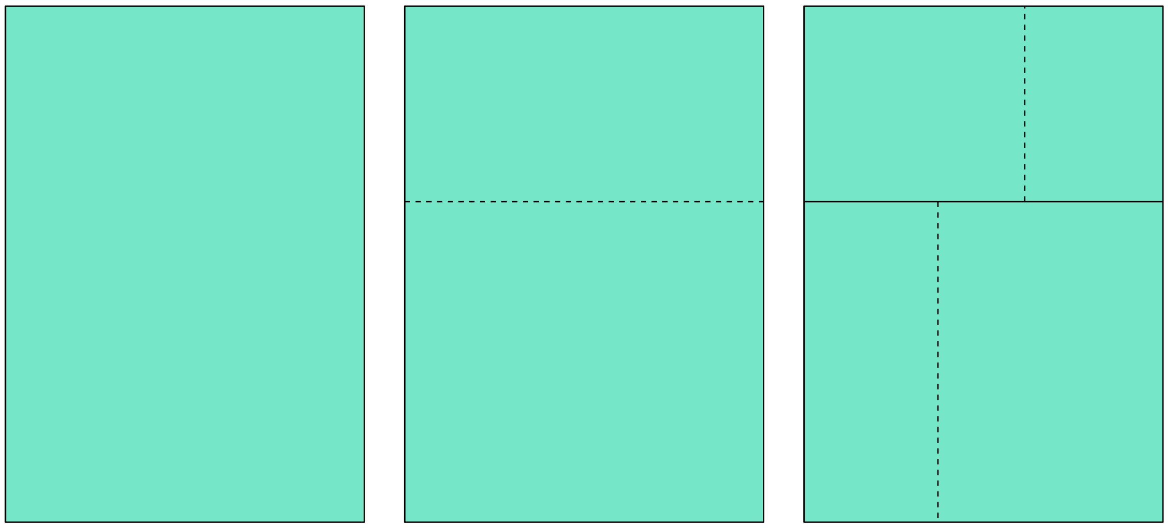


Figure 2.33: Recursive Subdivision of a Rectangle

geometric depth of the set of points. Due to its logic, this algorithm is only minimally slower than the previous one, requires only negligibly more computing power, and is more suitable for automated floor planning than the recursive subdivision method due to the possibility of controlling room sizes and placement through the position of the aforementioned points. Examples of the k-d tree method are given in the section 4.1.1.1.

Another method, also based on the binary tree construction, is the *squarified treemap* algorithm[35], which has the significant advantage of being able to define the percentages of the room sizes in advance. However, unlike the recursive subdivision and k-dimensional tree algorithms, this method is only suitable for rectangular initial shapes. Further explanations can be found in the section 4.1.1.1.

Methods that are not based on a binary tree structure include so-called *Voronoi diagrams*[36]. Similar to the k-dimensional tree algorithms, points are projected onto the surfaces to be subdivided, representing the position of each space. By creating a *Delaunay triangulation* of these points and then dividing the centers of the edges of the mesh by orthogonal lines, Voronoi cells can be constructed around the origin points. These denote the region that is geometrically closer to the Voronoi seed than to any other point on the shape. This method of subdivision has the significant advantage that the initial body can be of any shape, thus providing great variability in layout generation. This method is described in detail in section 4.1.1.1. However, a disadvantage of conventional Voronoi diagrams is the lack of control over the area of each Voronoi cell. To overcome this drawback, *weighted Voronoi diagrams*, also called *Laguerre Voronoi diagrams*[37], can be adopted. They allow a weight to be assigned to each Voronoi cell, thus influencing the size of the area of each cell in relation to the other cells.

[35]Marson and Musse 2010, “Automatic real-time generation of floor plans based on squarified treemaps algorithm”.

[36]Coates et al. 2005, “Generating architectural spatial configurations. Two approaches using Voronoi tessellations and particle systems”.

[37]Anuradha, Sabnis, and Thirumavalavn 2008, “Voronoi diagram voro: Application of interactive weighted Voronoi diagrams as an alternate master-planning framework for business parks.”

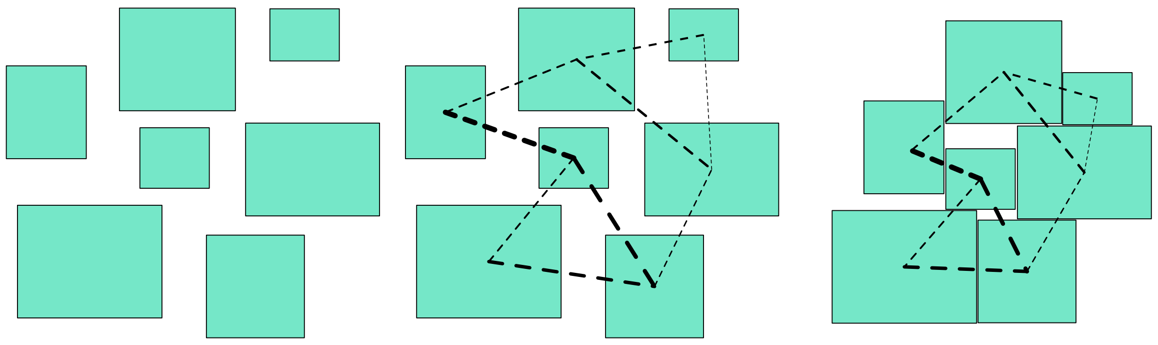

(a) Initial Configuration (b) Iteration (c) Final Aggregation

Figure 2.34: Aggregation Process

Aggregation A striking characteristic of both the grid-planning and the subdivision methods is that both require a predetermined initial shape. This is advantageous for certain automatic floor planning tasks, but it can also be limiting as it restricts design variation. The family of *aggregation methods* overcome this limitation by being structured in such a way that only the shapes of the individual rooms need to be defined in advance (figure 2.34a). As the name suggests, aggregation methods[38] attempt to create an agglomerate (figure 2.34c) of the predefined room shapes by applying different agglomeration functions.

To achieve the best possible result, evaluation formulas and conditions have to be created, which can be optimised by shifting the individual spaces (figure 2.34b) using various algorithms. For example, a fitness function can be created that tries to minimise the distance between the individual areas without overlapping them. This can be done by specifying different strengths of relationships between the rooms, if appropriate. This fitness function can then be used as a constraint for an *evolutionary algorithm*, which, similar to the biological concept, creates populations that influence different parameters and go through natural selection processes, mutations and crossovers per iteration to find the best possible position of each room, represented by the highest degree of optimisation of the fitness function.

Similar to the evolutionary algorithm, *simulated annealing* can also be used as an optimisation strategy, as it can likewise find an optimal arrangement by iteratively evaluating the initially randomly selected solution of the parameters of the function to be optimised. This process, like the evolutionary approach, does not guarantee that the optimal spatial arrangement will be found. However, both methods have a higher probability of finding the global maximum of the functions to be optimised.

Another method that allows for some autonomy in the room arrangement is the use of *agent-based optimisation*[39]. Here, the individual rooms are initialised as agents acting independently of each other, so that each individual room evaluates its own position and orientation in relation to its environment. This evaluation takes place iteratively and in parallel until the desired final condition is reached. More information on agent-based optimisation can be found in the section 4.1.1.3.

Despite their advantages, the methods mentioned so far all have the common drawback

[38]As and Basu 2021, *The Routledge companion to artificial intelligence in architecture.*

[39]Guo and B. Li 2017, “Evolutionary approach for spatial architecture layout design enhanced by an agent-based topology finding system”.

of being time-consuming and particularly computationally intensive due to their iterative computation structure. Another possibility for aggregation is the creation of a physical attraction model by *spring force*, where the rooms are connected to each other in a predetermined way by physical forces and thus a desired space aggregation is generated at the end of the simulation, as long as they were determined to be fixed, impenetrable rooms. However, the main disadvantage of this method remains the high requirement of computing power and that there is no guarantee that the final result will be free of gaps. It must therefore be complemented by a displacement algorithm that tries to minimise the gaps in subsequent steps.

Last but not least, methods from the family of *shape-packing* algorithms can also be used in the aggregation strategy. Here, the individual rooms are shifted from a randomly chosen initial arrangement in such a way that the area they occupy is minimised. However, even in this case, gapless layout generation is far from being guaranteed.

2.3 Simulation

During the design process of any architectural concept, the attentive designer is concerned with the interaction between inanimate matter and its environment. In order to evaluate this interaction in a sound and correct way, a number of different methods have been developed that allow the final or preliminary design to be tested in real scenarios in order to evaluate its performance. This discipline is summarised under the concept of *building simulation* in architectural design.

In terms of this definition, the best known simulation technique, which is inevitably familiar to any designer or observer, is the representation of architectural projects in perspective space. Here, the abstract constructive element adapts to the perspective perception conditioned by the human eye, thus attempting to represent a simulation of the physical situation. However, when talking about simulations in the architectural context, one usually refers to more complex methods that imply either physical simulations, numerical simulations, virtual, real-time or even agent-based simulations. In the context of this thesis, the focus lies on physical and agent-based simulations, where the results can be either stochastic or *deterministic*. A variety of various simulation techniques with different variables can be carried out to provide detailed insights into the respective environmental behaviour. A list and description of available open source simulation software can be found in section A.3.2.

Light Simulations This simulation method is primarily concerned with simulating the photons of natural light emitted by the sun, which manifest their interaction with our physical world as solar radiation and can be measured in *lux* (figure 2.35). A variety of different values and measurements can be calculated, such as the *Daylight Factor (DF)*, the *Useful Daylight Illuminance (UDI)*, the *Daylight Autonomy (DA)* or even the *Glare Factor (GF)*. In this way, the positioning and dimensioning of windows and the choice of solar shading systems can be optimised[40]. However, light simulations also allow the simulation of illumination and its interaction with matter from artificial light sources. In addition, the absence of light or the shadow cast by a particular building shape at a particular time and day of the year can be simulated in a similar way (figure 2.36).

[40]Aksin and Selçuk 2021, “Use of Simulation Techniques and Optimization Tools for Daylight, Energy and Thermal Performance: The case of office module (s) in different climates”.

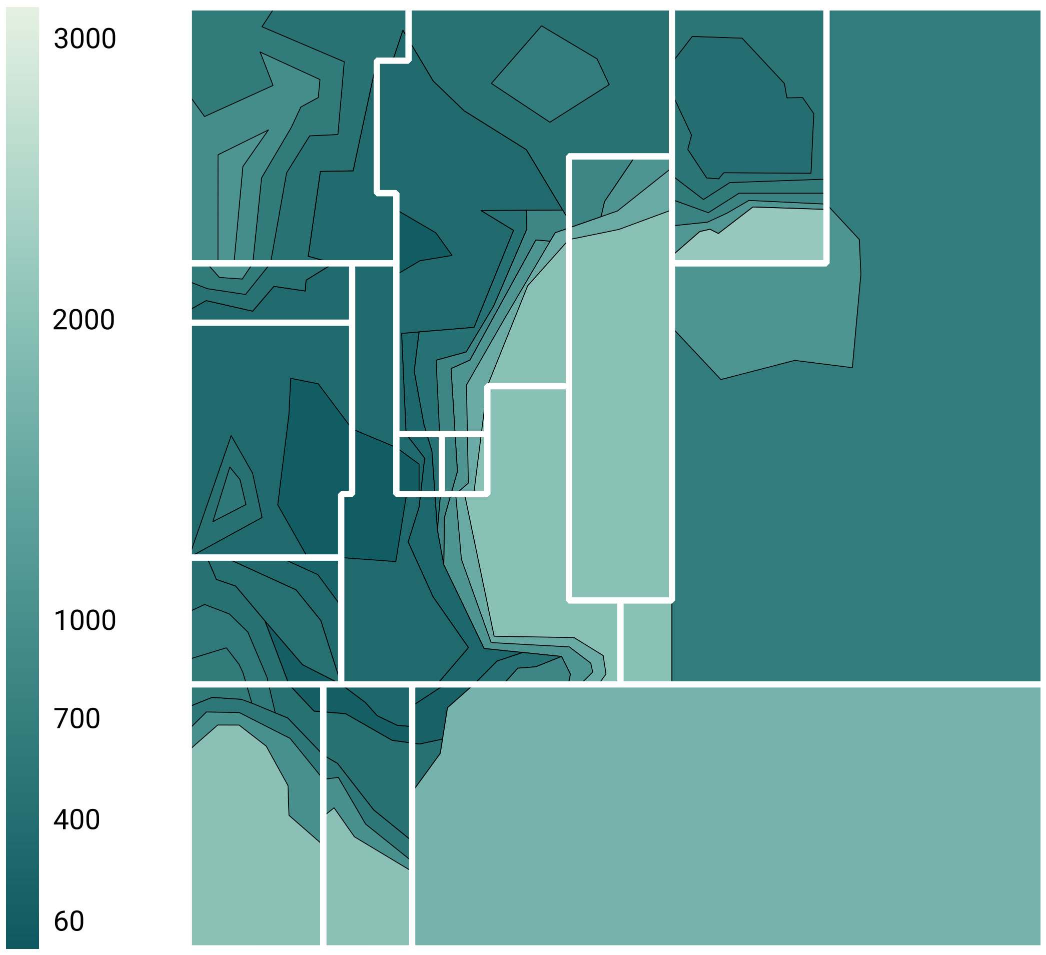
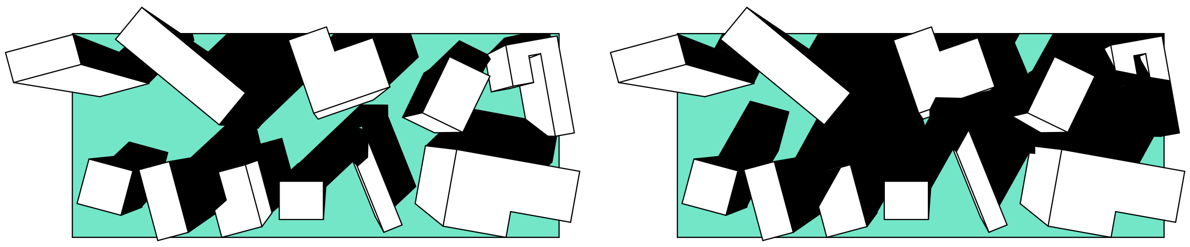


Figure 2.35: Light Simulation Villa Savoye in lux



(a) 1 May 13h30

(b) 1 November 13h30

Figure 2.36: Shadow Simulation

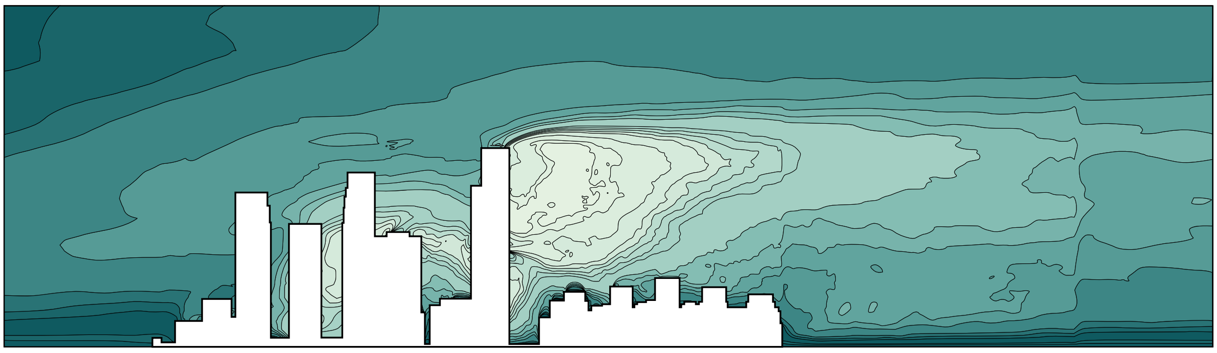


Figure 2.37: Wind Simulation

Acoustic Simulations This method of physically simulating the acoustic properties of defined spaces involves calculating the interaction of sound waves with the media in which they propagate. Either a sender object is defined in advance whose emitted sound waves are to be analysed, such as a professor in an auditorium, or artificial sender objects are created to simulate sound transmission through certain materials, such as a neighbour's footsteps and their sound transmission through the floor to the apartment below.

Vibration and Earthquake Simulations Material properties in the presence of vibration are not only of interest in the context of sound propagation, but can also provide information about general physical properties of the overall structure. Vibration and seismic simulations are an essential method of physical simulation for evaluating design decisions, as resistance to different levels of vibration can be calculated by changing the structure of the static frame or its material composition. For example, in geographical regions where earthquakes occur regularly, this type of simulation can provide answers to important questions about the stability and load-bearing capacity of the proposed structure.

Fire Simulations The propagation of fires and the changes in essential material properties under extreme heat are difficult to calculate mathematically, because the interaction of many different factors, such as the geometric structure of the building, the air density, the material composition and the connection of individual elements and their spatial proximity, significantly determine how the potential fire would affect the architectural object. Fire simulations are therefore the most reliable way of analysing a building from a fire safety point of view.

Rain and Wind Load Simulations As buildings are inevitably exposed to natural forces, thorough calculations of the interactions between these forces and architectural objects are required. Although rain and snow loads are often calculated using mathematical equations rather than complex simulations, wind forces are difficult to calculate in a simple mathematical way. For particularly high buildings, the horizontal wind load can even be considered a limiting factor and must therefore be simulated and analysed as accurately as possible according to the environment (figure 2.37). In almost all wind and rain simulations, it is necessary to take into account and model the immediate urban or natural environment, as certain wind corridors can form, which can cause unexpected horizontal forces.

Airflow Simulations In order to provide a comfortable microclimate within a designed building, or to prevent possible health risks due to reduced fresh air supply, general airflow simulations within the building need to be carried out. This involves simulating the medium of air and its movement through each room of the building, calculating the rate at which a certain percentage of existing air in a room is replaced by new incoming air, or even which method of ventilating the whole building requires the least amount of energy. Of course, the heating energy lost to incoming cold air also plays a role in the simulation, but it is primarily an important aspect of energy performance simulations.

Load-Bearing Simulations Almost every architectural creation is subject to forces based on Newton's laws of physics, essentially influenced by the earth's gravitational pull. In this respect, architecture, as an engineering profession, is more closely related to the constant consideration of the forces at work than any other activity. In this context, it is not surprising that methods for simulating the load-bearing capacity of individual materials, elements and even the entire structure are among the elementary tools of architectural practice.

Building Condition Simulations This family of physical simulation methods has a special place in simulation methodology because it is one of the few methods that takes a largely four-dimensional approach. Building condition simulations look at how individual materials change and possibly degrade over time to make predictions about the life cycle of individual components or even the entire structure. Furthermore, due to its temporal aspect, this simulation method can be integrated into the analysis of the material cycle, which provides information on the possibility of recycling certain materials once they have reached the end of their life.

This list of different physical simulations represents the principal methods for simulations in everyday architecture and engineering, but can be complemented by a variety of other simulations. Agent-based methods are less common simulation techniques, but can also provide valuable information about the performance of designed architectural objects.

Building Occupancy Simulations A specific simulation that can have significant application in architecture is the creation of agents that represent the occupants of a building or the visitors to a public building. The individual agents are given defined behavioural patterns and simulated using stochastic methods in an architectural context. The interesting aspect of such agent-based methods is that the individual actors can exhibit new, unexpected behaviours among themselves and in interaction with the architectural framework, which can indicate design shortcomings or confirm or refute certain spatial and programmatic design intentions.

Network Flow Simulations This method simulates the interaction of specific actors in a network in order to observe different patterns of behaviour in their interaction within the context of the given network topology. A widely used application in the urban context is traffic flow simulation, where the behaviour of vehicles and other road users is observed and analysed in the context of the given road infrastructure. In architectural applications, it can be the graphical modelling of public buildings, where certain agents go about their daily activities in and around the building, thus revealing possible bottlenecks in the spatial configuration.

Safety and Evacuation Simulations Another important application of agent-based simulations is to project the behaviour of crowds under specific circumstances. In this context, safety and evacuation simulations assign general behaviours to individuals that correspond to emergency situations in the simulated scenario. As those type of situations are influenced by a large number of different variables, the use of stochastic methods is necessary. The results of such simulations can therefore provide important insights into the optimal circulation within a building in evacuation situations.

Crowd Simulations The behaviour of large crowds differs in some respects from the way individuals respond, as some patterns of human behaviour are only exhibited when a certain number of people are present in and around the space. By defining the individuals of a crowd as distinct agents, the difficult-to-predict behaviour due to the interaction of the single agents can be represented and analysed. This provides formal and topological information about the performance of the designed architecture in relation to its use of space.

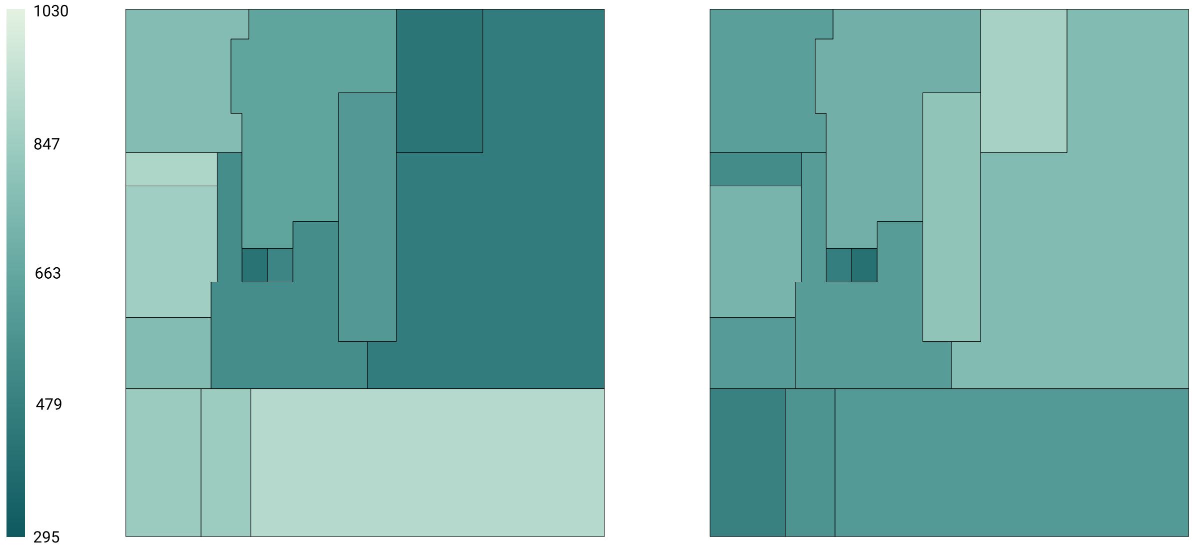
Typically, the aforementioned simulations are performed in advanced design phases of the project, or even after the design process has been completed, since in most cases they are elaborate processes which, in their conventional form, are not economically suitable for integration into an iterative process. This implies that, in reality, any shortcomings discovered during the simulation stages are often costly and difficult to correct in the architectural design[41].

2.3.1 Energy Performance

Energy simulation is a key tool in architecture for investigating and optimising the energy efficiency of buildings (figure 2.38). With growing global energy challenges due to climate change and limited resources, such simulations are becoming increasingly important and are commonly referred to as *Building Energy Modelling (BEM).* They use complex mathematical models and algorithms to analyse the energy performance of buildings, taking into account factors such as climate, geometry, material properties, use and energy transfer. The data collected is converted into *energy balances* to predict the behaviour of the building and identify weak points. By optimising the energy performance of buildings, significant energy savings can be achieved, bringing financial benefits and helping to reduce energy consumption and CO2 emissions. This type of simulation allows architects and designers to test different scenarios and optimise the energy performance of buildings before they are actually built. In practice, they are increasingly being used to make buildings more sustainable and energy efficient, both for new buildings and for the renovation or conversion of existing buildings.

Energy simulation in architecture can be performed in a variety of ways, including dynamic, thermal and structural simulations. Each type of simulation requires different input data to obtain accurate results. Typically, data such as building geometry, material properties, climate data and building occupancy data are required. The resulting values vary according to the type of simulation. For example, dynamic simulations can produce energy consumption projections for the building on a daily, weekly or annual basis. Thermal simulations, on the other hand, can map the temperature distribution in the building and the heat output from heating and air conditioning systems. Structural simulations can calculate loads on structural elements and the load-bearing capacity of materials. Despite the benefits of energy simulation, there are a number of drawbacks and limitations.

[41]Ali 2023, "Architectural Pipeline: An experiment into the role of topological graphs in the early stages of architectural design in the era of machine learning".



(a) Sensible Cooling (b) Sensible Heating

Figure 2.38: Energy Simulation

One of the biggest challenges is obtaining accurate input data[42], as material properties and actual building use are often unclear or difficult to obtain. In addition, simulations can be time-consuming and require powerful computers, specialised software and trained personnel to operate them[43].

The results of energy simulations are usually presented in numerical form, such as charts, graphs or tables and may have different units depending on the values simulated: for example, energy consumption projections are usually measured in kilowatt-hours (kWh) or even megajoules (MJ), while temperatures are expressed in degrees Celsius ($^\circ C$), Fahrenheit ($^\circ F$) or Kelvin (K).

2.3.2 Optimisation

Simulation-based optimisation is a relatively recent but growing area of architectural design tools[44], as it allows practitioners to test and manually or automatically optimise different design scenarios before they are realised. The aim of this simulation-based optimisation is to adjust and vary certain parameters, such as dimensions, materials, shapes, rotations and general transformations, so that the simulated evaluation function achieves its optimal performance[45]. An intuitive approach to optimisation through simulation in architecture is manual parametric testing, which allows architects to evaluate the overall design through simulations by varying specified parameters. Iterative processes and parameter adjustments are then used to achieve the best possible result.

The advantage of this optimisation strategy is that once the parametric model has been defined, little expertise is required on the part of the designer to optimise it. However, this

[42]Hauck 2017, “Energy Model Machine (EMM)”.

[43]Chatzivasileiadi et al. 2018, “The effect of reducing geometry complexity on energy simulation results”.

[44]Sebestyen and Tyc 2020, “Machine Learning Methods in Energy Simulations for Architects and Designers”.

[45]Aksin and Selçuk 2021, “Use of Simulation Techniques and Optimization Tools for Daylight, Energy and Thermal Performance: The case of office module (s) in different climates”.

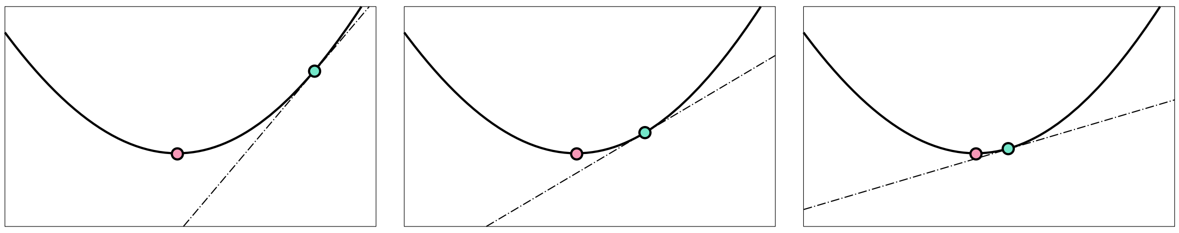


Figure 2.39: Gradient Descent Optimisation

approach is extremely time and computation intensive and does not guarantee optimal results due to its intuitive logic. As a result, over the years many automatic optimisation algorithms and methods, most of which have their origins in mathematics or computer science, have proved useful in architectural optimisation applications.

Deterministic Optimisation Methods Deterministic optimisation methods have become an indispensable part of today's scientific world. They are based on the application of mathematical models to find an optimal solution to a given problem (figure 2.39). These methods offer a variety of techniques, such as the *gradient descent method*, the *Newton-Raphson method*, the *conjugate gradient method* or even the *quasi-Newton method*. The gradient descent method is a well-known algorithm[46] used in many optimisation applications. It is particularly effective in finding a solution that minimises the value of a given function. It can converge quickly even in complex cases and is easy to implement. The Newton-Raphson and Conjugate Gradient methods are useful for large and complex problems. They can find solutions accurately in a short amount of time, using *local Hessian matrix* information. The Conjugate Gradient method is especially useful for *symmetric matrices*, while the Newton-Raphson method is good for non-linear functions. The quasi-Newton method is similarly an effective method for optimising non-linear functions, but is based on the approximation of the Hessian matrix by a *positive definite matrix*. The method is known for its fast convergence and ability to work well in practice.

Stochastic Optimisation Methods Stochastic optimisation methods provide an alternative approach to solving optimisation problems, and are particularly useful for solving complex tasks. Unlike deterministic methods, they use probabilistic models to find an optimal solution. Random processes are used to search for an optimal solution. There are several methods of stochastic optimisation, such as simulated annealing, *particle swarm optimisation*, genetic algorithms and *Bayesian optimisation*. Simulated annealing is a method that uses random search to find an optimal solution. In this process, an energy level is calculated and reduced as the optimal solution is gradually approached. Particle Swarm Optimisation and Genetic Algorithms use populations and evolution to find an optimal solution. Particle Swarm Optimisation simulates the behaviour of swarms or groups of particles to collectively find an optimal solution. The position of each particle is adjusted at each iteration to find the function maximizing variables. Genetic algorithms are based on the idea of natural selection and evolution. Solutions are represented in the form of *chromosomes* and evolve through *crossover* and *mutation*. Through continuous evolution of the population, a better solution is gradually found. Bayesian optimisation uses Bayesian statistics to find an optimal solution. It involves building a probability

[46]Zhang et al. 2021, "Dive into deep learning".

model that maps the relationship between the input and output variables. By combining *prior* and *likelihood functions*, the optimal solution is calculated.

Machine Learning Optimisation Methods Machine learning based optimisation methods represent another approach to optimisation and use algorithms and models from computer science to find an optimal solution for a defined or even undefined function[47]. These methods usually use the largest possible amount of data to train the model to make the best decisions. There are several machine learning methods for optimisation, such as *stochastic gradient descent*, *reinforcement learning* and *artificial neural networks.* Stochastic Gradient Descent is a method that uses gradient descent techniques in combination with random processes to find an optimal solution. In this process, a *cost function* is minimised step by step until the optimal solution is reached. Reinforcement learning is a learning technique in which the system learns how to optimise its actions by interacting with the environment. The system receives feedback in the form of rewards or punishments for its actions and adjusts its strategy accordingly. In this way, the system learns how to optimise its actions and achieve its goals. Artificial neural networks are a well-known method based on modelling the neuronal structure of the human brain, which makes them particularly suitable for processing large amounts of data. Like other methods, neural networks can be used for optimisation by creating a model of the function to be maximised or minimised. A particular feature of neural networks is the creation of a *predictive function* by training the neural network to find the optimal solution.

[47]Sebestyen and Tyc 2020, "Machine Learning Methods in Energy Simulations for Architects and Designers".

Chapter 3

State of the Art

The following overview of significant academic research, studies and projects related to the topic of this thesis is divided into four main areas: Graphs and Topology in Architecture, Feedback in Early Design Stages, Architecture and Machine Learning, and Synthetic Architecture Datasets. Each of these topics is further subdivided into several subtopics in order to simplify and structure the reading. It is important to note, however, that a large number of the references described do not easily fit into a single category, as most of the works listed demonstrate a methodology that combines several topics, such as machine learning, topology and automatic floor planning for instance. The literature listed below represents essential sources for the elaboration of the topics covered in this thesis. Further studies and research on the subjects covered are listed and evaluated in the section A.1 in the appendix.

3.1 Graphs and Topology in Architecture

A renewed interest in topological analysis approaches in architecture, as well as an intensive engagement and research in the field of graph-theoretic methodology in data and computer science in the last decades, has led to a variety of different research and projects at the interface of both disciplines. The application of topological graphs and their inherent methods to the AEC industry has led to new design approaches and analysis techniques that have the potential to significantly enrich architectural thinking. This section reviews the state of the art of graphs and topology in the architectural context by referring to the most relevant articles, books and experiments on and around these subjects.

3.1.1 Graph Theory in Architecture

The theoretical field of graph theory has found diverse applications in architecture, particularly in the analysis of spatial structure, layout and organisation. The beginning of the study of graphical relations and mathematics can be traced back to Euler in the year 1736. In the subsequent centuries multiple fundamental research has been conducted around the topic. One of these contributions is the book *Introduction to graph theory* by Wilson in which he introduces the concepts of graph theory through examples that define the structure and components of graph networks and explain special cases and methods such as paths, cycles, trees, planarity, digraphs and *colouring.* This lecture proved to be a valuable source of information for an introduction to theorems and methods of graph analysis. In the same year Earl and March in their book “Architectural applications of graph theory” deal with the application of graphs to two-dimensional architectural floor plans in order to analyse and evaluate the connection of spaces and their arrangement in the spatial layout design. Here, graphs are applied to the architectural example in two different ways: on the one hand, they take the form of programmatic-organisational information and, on the other hand, they represent the structure and connection of the wall elements represented by the edges of the graphs, whereby the empty area thus created within the graph describes the rooms of the architectural object.

On a technical level, starting from a BIM model, the study “Analysis of building structure and topology based on graph theory” by Van Treeck and Rank derives four different types of graphs: a structural element graph, a graph connecting room surfaces, a room adjacency graph and a graph representing object relations. The relationship between them is explained and the use of these networks as input for successful energy simulations is demonstrated. Due to the interest raised by the combination of graph theory and architectural practice, parallel research was carried out in several areas at the beginning of this century. In this context and with a more interdisciplinary approach, Franz, Mallot, and Wiener in “Graph-based models of space in architecture and cognitive science: A comparative analysis” connect the broad field of psychology with architecture by applying specific graph-based computational methods. The research includes spatial cognition-based graphs such as the *occupancy grid*, *place graph* and *view graph*, as well as architecture analysis-based graphs such as the *access graph*, axial maps, isovists and Visibility Graph Analysis. The aim of this work is to demonstrate the application of spatial cognition methods as design tools in architecture. Furthermore, the paper “Applications of graph theory in architectural analysis: past, present and future research.” by Dawes and Ostwald adds to this psychologically oriented approach by critically examining three essential methods of spatial syntax, namely axial line, convex space and isovists, through architectural examples. He suggests ways to extend the established, abstract and geometry-independent methods of analysis. The analysis tools developed aim to reconcile space syntax theory with geometric and formal information.

The technical approach to incorporating graph-theoretic concepts into AEC was further explored by Isaac, Sadeghpour, and Navon who, with “Analyzing building information using graph theory”, provided a basis for integrating graph-theoretic analysis methods into BIM models through a thorough study of IFC files and their ability to be synthesised into semantic graphs that could in turn describe the hierarchical relationships of architectural elements in the model. The authors further discuss significant advantages such as the representation and mathematical analysis of abstract information through computational graph algorithms. Another technical implementation of graph concepts is documented in “Two-graph building interior representation for emergency response applications” by Boguslawski et al. This paper develops BIM graph retrieval methods for the coordinated emergency response of rescue teams and calculates safe routes, shortest paths and fire spread within the model using graph analysis methods. The different types of graphs used in this study are summarised under the concept of '*variable density networks*'.

To conclude the general subject of the synergy between graph theory and architecture, Nourian provides a general research on the application of graph theoretical concepts in architectural design practice entitled “Configraphics: Graph theoretical methods for design and analysis of spatial configurations”. The motivation of this work was to provide designers with a spatial configuration-based design practice that complements traditional formal research. Within this framework, the configurative design toolkit '*Syntactic*' was developed for the application of graph analysis in the field of AEC.

3.1.2 Topology and Space Syntax

Similarly, topological analysis methods have found numerous applications in architecture over the last century, particularly in relation to form finding, design optimisation and spatial analysis. One of the fundamental literature in this field is the research of Kelley published in his book *General topology* of 1955. This document represents an essential resource in the field of mathematical topology as it thoroughly explains the fundamentals and advanced methods related to the concept of topology. Of particular importance were the explanations of elementary topics and algebraic concepts such as sets, relations and Boolean operations such as union and intersection. Thirty years later, Baglivo and Graver in their work, *Incidence and symmetry in design and architecture* extensively discuss the connection of graph-theoretical concepts in the design context through topological methods. The book begins by examining geometric symmetry through the application of graph-theoretic methods, then progressively delves into topological transformations, ultimately relating them back to graph-theoretic concepts.

Space significantly influences social structures and behavioural patterns, and society in turn has a considerable impact on its physical environment. This interaction has been intensively analysed in the foundational book *The social logic of space* by Hillier and Hanson, which explores the influence of buildings and urban structures on the formation of social patterns. It further develops tools that combine social and cultural approaches with constructive ones, such as the methods of analysis collectively known as *space syntax*. Following this publication, technical implementations of the theory introduced were carried out in 2009 by Y. Li et al., among others. The paper, *Design with space syntax analysis based on building information model*, presents and discusses the implementation of a space syntax analysis tool during the initial design phases. The developed feedback and analysis software, called '*Archispace*', can analyse and evaluate BIM models in terms of various space syntax methods such as connectivity, integration and *intelligibility*. During this study, the authors identify several challenges in applying space syntax concepts to BIM models, such as the lack of a unified topological syntax or interoperability with the IFC file format. The preoccupation with combining digital analysis tools and spatial syntax concepts was further explored by Nourian, Rezvani, and Sariyildiz in the study "Designing with Space Syntax: A configurative approach to architectural layout, proposing a computational methodology". In the course of this work, a complete design pipeline with integrated analysis metrics was developed. The user defines the program using a graph and has the additional possibility to impose certain geometric requirements. The algorithm then provides different floor plan suggestions with individual room syntax evaluations, thus enriching the design process.

In 2018, Aish et al. document the developed topological software toolkit called '*Topologic*', which establishes the connection between architecture and discrete mathematical operations by dividing the architectural model into elements of different hierarchies and dimensions. This ultimately enables topological analysis and Boolean operations, including neighbourhood and lateral relationship queries. In addition to presenting the functionality of the software toolkit, a subsequent publication in the same year, "Topologic: A toolkit for spatial and topological modelling", describes two applications of the software in the AEC domain. Firstly, the generation of non-manifold topological models used as input for energy simulations is considered, and secondly, the possibility of structural analysis and simulations through topological models is explained.

Finally, in the year 2021, a concrete analytical application of space syntax methods using three villas designed by Palladio is demonstrated in "Examining control, centrality and flexibility in Palladio's villa plans using space syntax measurements" by Dawes, Ostwald, and J. H. Lee. The authors focus in particular on evaluating the importance of the main salon and the overall flexibility of the spatial layout through the creation of so-called *justified plan graphs*.

The overview of the current state of research in graph theory and topological approaches in the context of architecture demonstrates the diversity and wide range of different research and application areas within the fields under consideration. However, it becomes clear that research into topological graphs in architecture is still in its infancy. This is partly due to the short time that has elapsed since the two subjects were brought together and applied in the architectural design process, and partly due to the lack of uniform design languages and processes in the AEC industry. It becomes thus apparent that, despite remarkable progress in the respective fields, the integration of graphs and topology in architecture is still an emerging area of research with room for further study and experimentation in order to apply the full potential of these mathematical concepts in the context of conventional architectural practice.

3.2 Feedback in Early Design-Stages

Early design stages are critical to the success of an architectural project, as they form the solid basis for later decisions and developments. Feedback about the design intentions and individual decisions during these preliminary steps is therefore essential to potentially optimise the design, identify problems and develop solutions. Likewise, performative feedback in early project phases can have significant economic and environmental benefits and is therefore of great value during the initial design process.

3.2.1 Decisionmaking and Feedback-Tools

In recent years, several techniques, tools and prototypes have been developed to provide feedback to designers during the initial design iterations. In particular, the application of computational methods has allowed a diverse exploration of different tools and algorithms to provide optimisation and support the decision-making process.

In this regard, the development and training of a neural network model that predicts the heating and electricity energy consumption of school buildings is described by Paterson et al. in their publication, "Real-time Environmental Feedback at the Early Design Stages". The resulting toolkit, *SEED*, could provide energy performance during the design process based on specific design parameters such as glazing ratio, floor area orientation and others. However, the achieved accuracy of 40% still leaves room for improvement. In order to successfully perform conventional energy simulations, BIM models often require substantial simplification at the geometry level. The research, "The effect of reducing geometry complexity on energy simulation results" by Chatzivasileiadi et al., investigates the extent to which this geometric abstraction affects the simulation results and to what point geometric simplification is possible without causing significant issues and loss of accuracy during the simulations.

Addressing the ambiguity between the need for early energy consumption simulations and the significant time and resource investment, Singh et al. apply in 2020, neural networks to predict energy consumption of buildings. The developed methodology allows the prediction of both embodied and operational energy demand during the initial design phases, providing a feedback function for the overall energy performance without any special effort or knowledge required from the designer. One year later, to provide continuous feedback on the performance of designed floor plans in terms of daylighting and energy performance, Yousif et al. similarly trains a neural network with labelled floor plans. The resulting modified *Pix2Pix* generative adversarial network (GAN) is capable of predicting heat maps for new floor plans with a high degree of accuracy. However, this method is limited to the second dimension and requires a pixel-based representation of the floor plan as input.

3.2.2 Optimisation in Early Design-Stages

A particular type of feedback function in design processes is the analysis of information about the performance of the designed layout, allowing the application of optimisation algorithms with the aim of automatically improving the design with respect to the chosen performance parameter at an early stage. In this context, a wide range of algorithms such as evolutionary approaches or simpler function optimisation computations have found their application in the architectural context. However, meta-heuristic or multivariable approaches have the significant disadvantage of being very time consuming and resource intensive. To address this issue, the study "Parallel planning-An experimental study in spectral graph matching" by Schaffranek introduces methods for optimising multiple parameters through *spectral graph matching* based on graphically encoded information. The main advantage of this method is that, unlike conventional methods, it does not rely on iterative processes, thus avoiding significant time overhead. However, the algorithm can only take into account information that can be represented in graphical form, which significantly limits its applicability. In addition to multivariable optimisation methods in the design process, the research carried out by Canestrino et al. in the year 2020 provides applications in constructive detail design using a complete BIM model. This study emphasises the importance of optimisation methods in the automatic exploration of a variety of different design and construction solutions.

The attempt to provide designers with a set of tools during the design process to achieve an optimised adjacency of programs and spatial functions through the application of graph centrality-based design heuristics is documented in "Space Matcher" by Fuchkina et al. Topological-graphic analysis methods such as *distance mapping* and shortest path calculation, in conjunction with *fuzzy logic evaluation*, are used in the course of this research to find the best-optimised adjacency variant of the rooms in the floor plan.

Finally, to achieve parallel optimisation of spatial configuration and function, Muslimin discusses in "Experience Grammar: Creative Space Planning with Generative Graph and Shape for Early Design Stage" his experimental combination of shape grammar and spatial syntax. The author was able to extend the syntax of the shape grammar with topological rules based on the computation and analysis of graph-theoretical methods in order to create an optimisation tool that can generate shapes and layouts according to space syntactic metrics.

3.2.3 Performance-Based Design

The designer is always concerned with performance in the broadest sense of the word in architecture, which means that automatic feedback on these very evaluation metrics can bring significant benefits. As an umbrella term, these approaches can be described as performance-based design, where the primary goal is to develop tools and methodologies for continuous performance evaluation that can complement traditional modelling and conceptualisation processes. As this is a relatively recent research topic, the progress made over the last decade will be presented in chronological order:

The *Generative design system*, developed by Caldas and Luìs Santos and documented in “Generation of Energy-Efficient Patio Houses With GENE-ARCH”, combines shape grammar with an energy simulation engine to optimise houses with a patio typology using evolutionary algorithms. The choice of energy performance as the optimisation metric allows for an automated performance-based design process. Furthermore, in order to clarify the requirements for performance analysis, Jabi in 2015 investigates the importance of non-manifold topology-based models for building performance simulation in the early stages of the design process. It is shown that this type of geometry representation, with its division into hierarchical subelements and high level of abstraction with limited loss of information, is well suited as input geometry for subsequent energy simulations. This implies the importance of a tool for extracting non-manifold geometry from existing BIM models at different levels of detail and project stages.

The meta-analysis of the application of evolutionary optimisation algorithms in the field of AEC, carried out in 2021 by Canestrino, demonstrates that there will always be limitations to the application of optimisation algorithms during the design process. Not every architectural criterion can be abstracted to act as an evaluation metric for optimisation methods. The most common application of such algorithms is therefore limited to clearly defined elements and fitness criteria, such as the performance of shading systems or the formal appearance of a building in terms of its Daylight Factor. In this context, Rogers et al. developed a simple performance-oriented simulation technique in 2022 that does not rely on such optimisation algorithms. Rather, it integrates graph-theoretic analysis with congestion path finding and building usage data such as schedules, walking speed, origin and destination, allowing for agent-based simulation along specific paths to identify potential bottlenecks in the overall circulation system.

The contributions presented show that feedback in early design stages can be of significant importance in optimising the design process and improving the architectural and technical quality of the outcome. In recent years, numerous methodologies and tools have been developed to assist architects, mainly through decision support systems such as computer-aided and algorithm-based technologies. However, this area of research is far from exhausted, and new studies and experiments are published on a weekly basis, with new results showing novel applications and potentials for the architectural design workflow. This is particularly relevant in the context of the recent resurgence of interest in machine learning and artificial intelligence.

3.3 Architecture and Machine Learning

As a discipline that has been increasingly applied to more and more different fields for some time now, machine learning methods have also taken on various roles in architecture. In most cases, however, these applications are primarily experimental or artistic, and are accompanied by a wide range of different research, studies and experiments. The following section presents research on the role of machine learning in the AEC industry, categorised into different subtopics, ranging from literature on the impact on the field to specific research on graph machine learning applications.

3.3.1 Impact of Machine Learning

To begin with, the general implications of integrating machine learning methods into architectural practice are considered, as well as the new questions and issues raised by this combination.

In the context of an experimental application of *Generative Adversarial Networks (GANs)* for architectural plan generation purposes, the work "How Machines Learn to Plan - A Critical Interrogation of Machine Vision Techniques in Architecture." by Campo discusses the potential role that machine learning can play in the conventional design process and to what extent this might already be possible. Important aspects such as creativity and stylistic aspects are examined and discussed in detail from an architectural perspective. On the other hand, a much more technical look at the application of computational methods in the AEC industry is provided by the anthology *The Routledge companion to artificial intelligence in architecture* in 2021. Structured around subthemes ranging from theoretical foundations to artificial intelligence in architectural practice, a selection of authors provide insights into the various benefits, challenges and implications of adopting machine learning concepts in the field of architecture and construction. Particularly helpful for this work was the section entitled "AI in space planning" by Nagy.

On a broader level, the publication "Limits to Applied ML in Planning and Architecture" by Joyce and Nazim presents a cautious and realistic meta-analysis of the limitations of integrating machine learning-based tools in the architectural domain. Various topics such as scale, representation, input data, resolvability, creativity, autonomy and design quality are identified and discussed. In strong contrast to this literature, Chaillou in *Artificial Intelligence and Architecture: From Research to Practice* offers an optimistic outlook on the future synergy between architecture and artificial intelligence. Current applications such as floor plan design, urban planning, automatic facade conception, structural design and predictive simulations through *DaylightGAN* and *ComfortGAN* are examined. Finally, potential application areas are presented which, according to the author, could benefit from the use of artificial intelligence.

3.3.2 Design and Optimisation Applications

Looking at some concrete implementations of machine learning in design practice, the primary use can be seen as a support and optimisation system during the design process in architecture. For example, in the study conducted by Hauck in 2017, a neural network is trained to directly predict the energy performance of a designed architectural object, avoiding the costly detour via thermodynamic and energy simulations. The input features, collected in tabular form for training, consist of essential characteristics related to geometric and energetic parameters of the designed three-dimensional layout. Another type of machine learning based optimisation of architectural objects is presented in the publication "The synergy of non-manifold topology and reinforcement learning for fire egress" by Jabi, Chatzivasileiadi, et al. This work combines topological building graphs with *reinforcement learning* methods to find optimal escape routes in the event of a fire in the building being studied. What makes this experiment interesting is that the simulated fire sources have both a three-dimensional spread and a temporal component, justifying the application of unsupervised reinforcement learning. The work demonstrates the potential of using abstract geometric representations through non-manifold topology and graphs in conjunction with advanced machine learning methods.

The term *surrogate model* refers to the method of replacing a variable that is difficult to compute with an approximated function. This concept is explored in "Deep learning surrogate models for spatial and visual connectivity" by Tarabishy et al. using a trained neural network to accelerate the prediction of spatial syntax metrics such as spatial and visual connectivity. The pixel-based generation and training pipeline is thoroughly documented and explained throughout the paper.

Finally, in the topic of design and optimisation applications of artificial intelligence in the context of architecture, Chaillou takes a creative and experimental approach to the application of GANs in the architectural floor plan generation process, demonstrating an interesting use of deep learning concepts in the form and style finding process. The technical documentation of the training and evaluation processes is complemented by extensive research on learning and combining different architectural styles from different periods, recently published in 2021.

3.3.3 Graph Machine Learning in Architecture

The concept of graph machine learning is relatively new in computer science, and thus has only been applied to architectural research and practice since around 2020. However, the fundamental importance of graph theory in the discipline of architecture makes it clear that there is great potential for such deep learning methods in this field. Some of the main contributions to this topic are listed below.

Semantic enrichment refers to the process of adding descriptive information to an existing model. In the study "Room Type Classification for Semantic Enrichment of Building Information Modeling Using Graph Neural Networks" by Z. Wang et al., Graph Neural Networks, specifically a *GraphSAGE* model, was trained to predict the function of individual rooms in a BIM model with high accuracy. This application could potentially be extended to other elements of the architectural model, enabling interoperability through augmented BIM data. Extending the research topic of room classification methods, the "Room-based energy demand classification of BIM data using graph supervised learning." experiment published in 2021 experimented with classifying the energy consumption of individual rooms in a BIM model using supervised graph deep learning algorithms based on the GraphSAGE method. The model is thus able to predict the labels of individual nodes in a graph object.

Using *transfer learning* methods, the publication "Autocompletion of Design Data in Semantic Building Models using Link Prediction and Graph Neural Networks" by Eisenstadt, Bielski, et al. developed and evaluated the prediction of connections between individual programs in an architectural project. The trained graph neural network can autocomplete partial layout graphs and predict the probability of edges between nodes with high accuracy. This approach can be combined with node or edge classification to predict room functions and the types of connections between them. In terms of graph-wide prediction, Alymani, Jabi, and Corcoran explored the application of a *Deep Graph Convolutional Neural Network* to synthetically generated building graph datasets with different ground relations in two publications between 2022 and 2023. The trained graph neural network is thus able to accurately predict the ground relation of new graph objects. The framework of this research included supervised learning methods such as graph classification and unsupervised learning methods such as clustering.

Taken as a whole, this research demonstrates the versatility of machine learning in the field of Architectural Engineering and Construction. It is possible to optimise topological arrangements, improve selected performance characteristics such as energy efficiency, and provide a detailed understanding of spatial structures. It should be noted, however, that the field of machine and deep learning in architecture is in constant evolution and there are still many open questions and challenges. In particular, questions remain about the interaction between human designers and machine learning systems, the *interpretability* of machine learning models, and the adaptability of these techniques to different contexts and requirements.

3.4 Synthetic Architecture-Datasets

With the increasing adoption of machine learning methods in the design process, there is a growing need for qualitative architectural datasets that represent specific aspects of the project as well as possible. Questions about the state of the art in the generation and development of these datasets, their different forms and their multiple applications are explored in the following sections.

3.4.1 Parametric Design and Algorithms

The creation of such data requires the application of parametric design principles as well as the definition and formulation of certain architectural rules and conditions. This has led to extensive research and experimentation on these concepts and their adaptation to allow the construction of appropriate and qualitative frameworks for architectural generation pipelines.

The algorithm developed in "An algorithm for exhaustive generation of building floor plans" is able to generate optimised rectangular floor plans based on variable input parameters such as total area, room areas, wall lengths, room adjacencies and room orientations. The research conducted by Galle in 1981 laid the groundwork for subsequent experiments by introducing geometric partitioning methods, iterative division and graph-theoretic adjacency calculations.

Regarding the purely algorithmic tools for automatic architecture generation, several different methods and approaches have been identified in the course of the research. *Treemaps*, which are primarily used to visualise related quantities and offer interesting properties for constraint-based space allocation, form the first advanced space partitioning algorithm. However, they are limited to dividing purely rectangular areas. In their study from 2005, Balzer and Deussen present a Voronoi diagram-based form of treemaps that can allocate desired percentages to a broader set of shapes, including irregular polygons.

An extension of the Voronoi diagram presented here is the weighted Voronoi diagram or Laguerre Voronoi diagram, which is explored in the context of urban planning by Anuradha, Sabnis, and Thirumavalavn in 2008. The regulation of Voronoi cell sizes represents a significant advantage in the field of automated space subdivision. The influence of individual Voronoi seeds could thus be determined by adding individual weight values and corresponding weight-distance functions to the diagram points. In addition to the weight function, the morphology of the computed regions can also be regulated by this approach. In this context, Y.-C. Wang, Lin, and Seah, with a publication entitled "Voronoi diagram voro: Application of interactive weighted Voronoi diagrams as an alternate master-planning framework for business parks.", introduces a new algorithm based on an *orthogonal Voronoi diagram* and an adapted distance and neighbourhood function, as well as a modified *sweep-line algorithm* for computing *orthogonal Voronoi treemaps*. This method combines the advantages of each technique, providing accurate percentage control through treemap calculation, flexibility through Voronoi diagram structure, and avoidance of irregular space generation through orthogonal shape generation.

3.4.2 Automatic Floor Plan Generation

When dealing with geometric floor plan data based on buildings or even apartment complexes, the application of automatic floor plan generation methods is almost inevitable. This topic comes with its own set of implications and challenges, partly caused by the high degree of freedom inherent in the conceptual architecture process. The main experiments of the last decade are listed and explained here in chronological order:

In 2010, Knecht and König discuss the application of K-dimensional trees using evolutionary optimisation algorithms for the automated generation of floor plan layouts. The binary tree structure of K-d trees proved advantageous in the context of regulated space allocation, as a certain degree of controllability is provided through the initial definition of points in two or three dimensional space. Another type of partitioning algorithm is the so-called squarified treemap, which constitutes a variant of the regular treemap algorithms with the advantage of generating regions with high aspect ratios, resulting in more square-like spaces. This algorithm is introduced in “Automatic real-time generation of floor plans based on squarified treemaps algorithm” by Marson and Musse, who explain the structure and functionality of such methods and test them on an architectural example. The published paper additionally discusses approaches to automatic corridor generation.

Staying within the topic of automatic floor plan generation, the paper “A graph theoretical approach for creating building floor plans”, by Shekhawat and Duarte explains the development of an algorithm that employs graph-theoretic methods to generate a *rectangular floor plan* (RFP) based on a given adjacency graph. If this is not possible, an *orthogonal floor plan* (OFP) is constructed. This research is essential as most automatic floor plan algorithms generate rectangular floor plans exclusively, thus limiting the variety and design possibilities.

After several years of research in this area, Egor et al. in the year 2020 provide a collection and analysis of existing room layout generators, combined with newly developed methods and extensions of existing research. This analysis led to the implementation of a so-called ’*Magnetizing Floor Plan Generator*’, which respects the desired adjacency of individual rooms and their functions through iterative placement. An interesting aspect of this methodology is the introduction of corridor elements to address the issue of gaps between the generated room geometries.

3.4.3 Architectural Datasets

What forms can these datasets take and what information do they contain? What are the most common datasets and what are potential issues in their application? To explore these questions in more detail, it is necessary to take a look at the main research around the topic of dataset generation and to examine some example datasets.

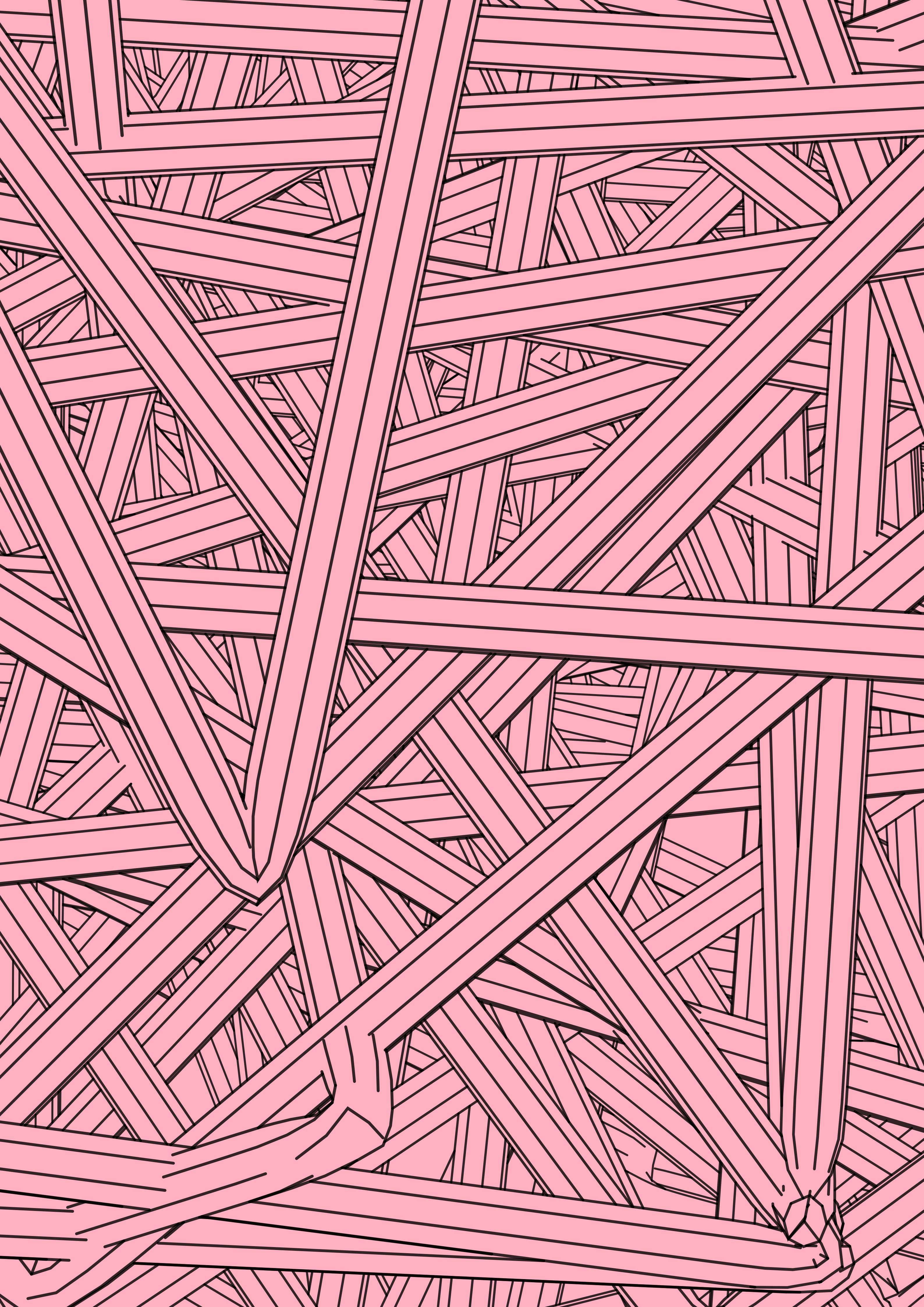
The research carried out by Kalervo et al. and published under the title “Cubicasa5k: A dataset and an improved multi-task model for floorplan image analysis” in 2019 resulted in the creation of a dataset containing 5 000 annotated, cleaned and vectorised floor plans. The dataset was generated using neural networks for architectural element recognition. It further provides an organised structure with polygonal room divisions and categorisation of individual elements and spaces, serving as a qualitative basis for training various deep learning methods in the architectural context. Building upon this vector-based *Cubicasa5k* dataset, two years later Lu et al. succeeded in synthesising a high-quality architectural graph dataset by applying relation detection methods to the Cubicasa5k data. A major advantage of this new set is the annotated graphs, which include node labels corresponding to room functions and edge labels describing the nature of room relationships. Furthermore, there is potential for combining the Cubicasa and the newly created *Cubigraph* datasets, as their datapoints correspond and their information is complementary.

Finally, the approach described in the work of Chen and Stouffs and published in a paper named "Robust Attributed Adjacency Graph Extraction Using Floor Plan Images" in 2022 demonstrates a method for the extraction of *attributed adjacency graphs* from pixel-based floor plan representations using image segmentation algorithms. The resulting graph dataset, generated using *ensemble learning methods*, includes node labels that provide information about the nature of the architectural element. The dataset has been released under an open source licence and is therefore well suited for further research on the topic.

From the literature cited, it is apparent that the creation of qualitative datasets is an important step in enabling the application of machine learning to the field of architecture. By employing Building Information Model and floor plan generation algorithms, as well as appropriate optimisation methods, researchers and architects can generate a variety of synthetic datasets that can be used to train and validate machine learning models. These synthetic datasets, in combination with data science methods, can help to improve the design process, optimise the performance of buildings or even develop new design concepts. The challenges and open questions identified during the literature review on synthetic architectural datasets mainly concern the quality and representativeness of the architectural data, as well as the origin bias that may be present in the information.

Part II

Contribution



The methodological part of this thesis serves to document the experiments carried out, which include the creation of a synthetic architectural graph dataset and the training and evaluation of different graph machine learning models. These experiments are the synthesis and application of the methods introduced in the theoretical framework, which come from different fields such as mathematics, geometry, urban studies, sociology and computer science, in order to propose answers to the stated research questions. To ensure a clear structure of the methodology, the essential steps and results of the experiments have been documented. However, it is important to mention in advance that the experiments presented are the result of a series of iterative processes, in which a multitude of different methods were tested in order to select the most appropriate approach.

The first part of the methodology deals with the application of different space partitioning algorithms that, integrated in a parametric framework and combined with architectural rules and conditions, initially allowed the creation of an architectural three-dimensional dataset. The resulting apartment geometries could be enriched with various information using topological methods to finally generate semantic topological graph networks corresponding to the individual apartment layouts. The added information includes categorical descriptions of each room, such as living room, bedroom, toilet, bathroom or utility room, but also typical descriptions such as apartment, window or room on a general level. Additionally, dimensional and geometric attributes have been included, providing information on the orientation and dimensions of each element.

Essential for the usefulness of the dataset is any information that allows statements to be made about the energy efficiency of the individual apartment layouts and is stored at the cellcomplex level. For this purpose it was necessary to subject each individual three-dimensional BIM model to accurate energy simulations and to evaluate the results. Accordingly, the resulting dataset contains 40 000 graph objects describing the topologies of the architectural objects, enriched with information about the dimensions and orientations of the individual elements. Each of these graph objects is assigned an energy efficiency value and a class, which are used as training data in the subsequent model training phase. The parametric, automatic generation of a synthetic dataset is in stark contrast to the conventional method, which is usually based on the accumulation, adjustment and annotation of reality-based architectural plans.

The second part of the experimental phase builds on the results of the first part and allows to evaluate the relevance of the dataset. This involves the application of graph machine learning to the generated knowledge graph objects in order to learn the relationship between the topology of the apartments and their corresponding energy efficiency. Two different supervised learning approaches are demonstrated, which differ to some extent in their possible applications.

The first step of the machine learning experiment is to design a classification model capable of categorising the unseen apartment graphs into their corresponding energy efficiency classes. In order to be able to define these energy classes, it is necessary to create a relative energy consumption scale due to the high level of abstraction in the structure of the energy simulation framework parameters as well as in the geometric composition of the individual apartment layouts. This measure takes into account the entire dataset and creates a new scale by dividing it into *quantiles* (table B.10). The accuracy of the trained model is then compared to other models, trained on tabular information rather than graphical data, in order to evaluate the benefits of graph machine learning methods in the context of energy efficiency classification compared to conventional machine learning algorithms.

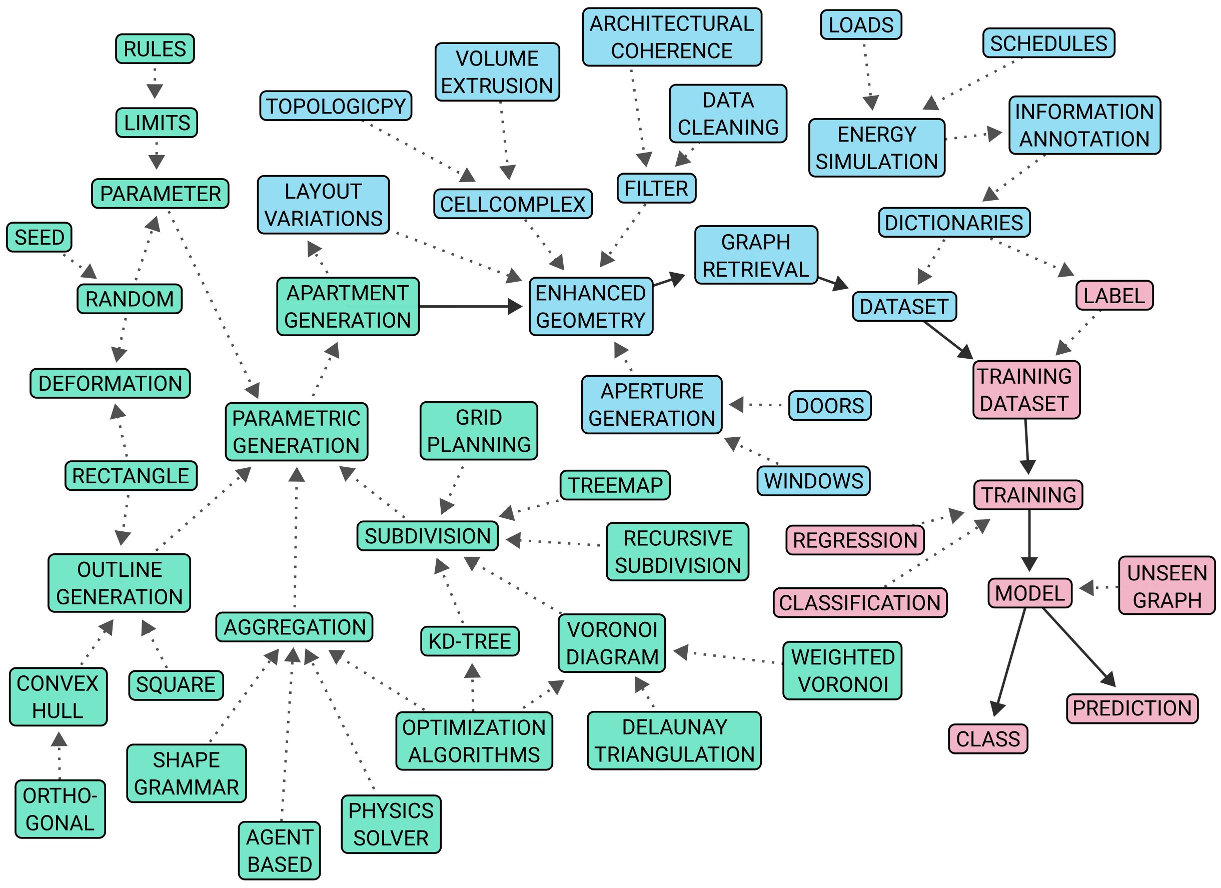


Figure 3.1: Experimentation Overview

The second deep learning method is to build a regression model that allows accurate prediction of energy consumption in MJ/m^2 from unseen topological knowledge graphs. This involved modifying and adapting the training setup of the *Deep Graph Convolutional Neural Network* and its model architecture to enable the regression task. The resulting model is then likewise compared with conventional regression models to make statements about its *precision*.

Figure 3.1 shows the overall process combining the two steps of dataset generation and graph machine learning. The continuous arrows show the principal workflow, while the dotted lines provide additional information about the steps required to accomplish the principal tasks. The diagram is divided into three colours, where green represents the dataset generation process and its subtasks, blue provides information about the composition of the data enhancement and the information generation through simulation, and red indicates the graph machine learning framework, which forms the final step of the experimentation.

Chapter 4

Synthetic Dataset Generation

Datasets form the elementary basis of data science methods such as machine learning, as this discipline is concerned with identifying regularities and correlations in the available information using large amounts of data. The patterns and trends in the dataset are recognised and used by a machine learning algorithm to perform the predefined task of the model. As the task of the experimentation involves supervised learning, it is essential that the dataset contains labels corresponding to the desired outcomes, in addition to the training information. This implies that the energy consumption value and energy class in the dataset must be assigned as a value to each corresponding datapoint.

One of the main requirements for the synthetic architectural dataset in this thesis is to have as much variation in the design layout as possible, so that a large number of different room arrangements and apartment morphologies can be associated as training data with their respective energy efficiencies. This allows for some generalisation in the application of the trained model. Typically, datasets in computer science have a tabular form and are represented by numerical values. Accordingly, architecture, as an artistic and geometric discipline, poses a challenge for the creation of datasets, because architectural geometric objects such as rooms, houses and apartment complexes, as well as their individual programs and structures, cannot be represented in an intuitive way as tabular, numerical data.

To solve this problem, several methods for representing architectural data in a machine-understandable form have emerged in recent years[1]. It is always important to distinguish what the desired output of the trained model is, as this has a significant impact on the requirements on the training dataset. The most common data representation is the one that has been used in architectural practice since the beginning of the profession, namely the representation by pictorial cross-sections of the architectural object, such as plans and sections.

The advantage of an image-based data representation is that it forms an integral part of the architectural representation, and thus a wide range of real data derived from scanned or digitally created plans is available. Furthermore, plan representation is deeply rooted in architectural practice, with many design processes traditionally focused on the two-dimensional plan level. As a result, plan representation has become a synthetic form of presentation that provides insight into unmapped elements of the architectural object by

[1]Kalervo et al. 2019, “Cubicasa5k: A dataset and an improved multi-task model for floorplan image analysis”.

adding semantic and descriptive information. In addition, there are usually several levels of detail in the two-dimensional plan representation of a project, as different instances in everyday architectural practice have different requirements for architectural plans. This enables the aggregation of multiple layers of information, all relating to the same object.

As with conventional machine learning based on image data, the bitmap pixels are tabulated with their colour values, resulting in a machine-readable table of values as a dataset. However, a major drawback of this conventional way of creating architectural datasets is the significant loss of information at multiple levels due to the pixel representation of architectural objects. Conceptual and programmatic information, such as the use of individual spaces, their topological relationship to each other, their extension into the third dimension, as well as information on materials and morphological properties, are difficult and, above all, inconsistent to represent in architectural plans. This means that this information cannot be taken into account during the training phases, even though it may represent elementary information.

Furthermore, pictorial datasets in architecture are associated with high memory and computational requirements, as they usually contain a large amount of non-informative areas, such as the white framing around the project. Architectural plans are often annotated with handwritten or computer generated annotations, which do not follow any uniform rules and therefore have to be removed in several successive filtering processes in order not to contribute to unwanted noise in the dataset[2]. In this work, an attempt is made to counteract the aforementioned drawbacks of traditional datasets by representing architectural objects not in pictorial form, but through topological graphs enriched with additional information, which can thus be called a knowledge graph dataset.

4.1 Space Partitioning

Since the synthetic data generation approach requires an automated algorithm, it is first necessary to consider the appropriate application of a geometry generation method. Basically, the objective is to develop a pipeline that takes specified architectural guidelines as input and generates a floor plan as output, which can then be automatically transformed into a full three-dimensional BIM model in subsequent steps.

It is essential that these methods have a degree of randomness to allow for the greatest possible morphological diversity in geometry generation. The process of choosing the appropriate space partitioning method is intended to compare as many different algorithms as possible in order to make a selection of suitable methods that could be integrated into the generation pipeline. As the aim was to develop as large a dataset as possible, properties such as the speed of the methods and the computational requirements were considered more carefully. However, the quality and especially the variety of the generated geometries was deemed to be the most important property for the selection of the algorithms.

[2]Newton 2019, "Deep generative learning for the generation and analysis of architectural plans with small datasets".

4.1.1 Algorithms

As presented in the theoretical framework, a variety of different automatic floor plan generation methods exist[3], some of which can be combined with different optimisation algorithms, and can be roughly divided into grid-based, subdivision or aggregation methods. In general, aggregation methods are much more computationally intensive and complex than subdivision or grid methods. However, this increased complexity also leads to increased variety between each generated layout, which is a significant added value in the demonstrated methodology. Finally, due to resource constraints, it was decided to include only three algorithms, coming from the subdivision family, in the final generation pipeline. However, in order to present a comprehensive picture of the experiments and to provide a basis for further research, methods that were ultimately deemed unsuitable for the generative pipeline for various reasons are equally presented.

4.1.1.1 Subdivision

The simplest methods for automatic space segmentation come from the family of subdivision algorithms, which, as the name suggests, subdivide a previously defined shape into a desired number of polygons according to defined rules. As they are mostly simple algorithms, they all share the advantage of being computationally inexpensive and therefore relatively fast in their execution. However, in order to integrate any method from this family of algorithms into an automatic pipeline, an parametric generation of the initial shape to be divided must first be considered.

Initial Shape A common method of automatically generating irregular convex polygons is to apply a *convex hull algorithm*[4] to a two-dimensional set of points. This involves first randomly placing a defined or arbitrary number of points in two-dimensional space. The amount of points is not important as long as there are more than two; however, a larger number of points increases the probability of generating a polygon with an equally larger number of sides. Once these points have been placed, the convex hull can be calculated and thus create the initial shape for the generation pipeline. However, a major drawback of this method is that it is only suitable for generating convex figures, making concave plans impossible.

In order to remain true to architectural reality, it is equally important to generate rectangular plans as well as their derivatives, which can be generated by shifting individual vertices of the rectangles. This is possible through the application of simple algebraic operations and does not require the use of established algorithms. Furthermore, by using *orthogonal convex hull algorithms*[5] (figure 4.1), the variety of basic shapes to be generated can be increased, since this is a method that generates rectangular polygons, analogous to ordinary convex hull methods, based on a randomly placed set of points in the two-dimensional space.

[3]Egor et al. 2020, "Computer-aided approach to public buildings floor plan generation. Magnetizing Floor Plan Generator"; Lopes et al. 2010, "A constrained growth method for procedural floor plan generation"; Nisztuk and P. B. Myszkowski 2019, "Hybrid evolutionary algorithm applied to automated floor plan generation".

[4]Anuradha, Sabnis, and Thirumavalavn 2008, "Voronoi diagram voro: Application of interactive weighted Voronoi diagrams as an alternate master-planning framework for business parks."

[5]An, Huyen, and Le 2021, "A modified Grahams convex hull algorithm for finding the connected orthogonal convex hull of a finite planar point set".

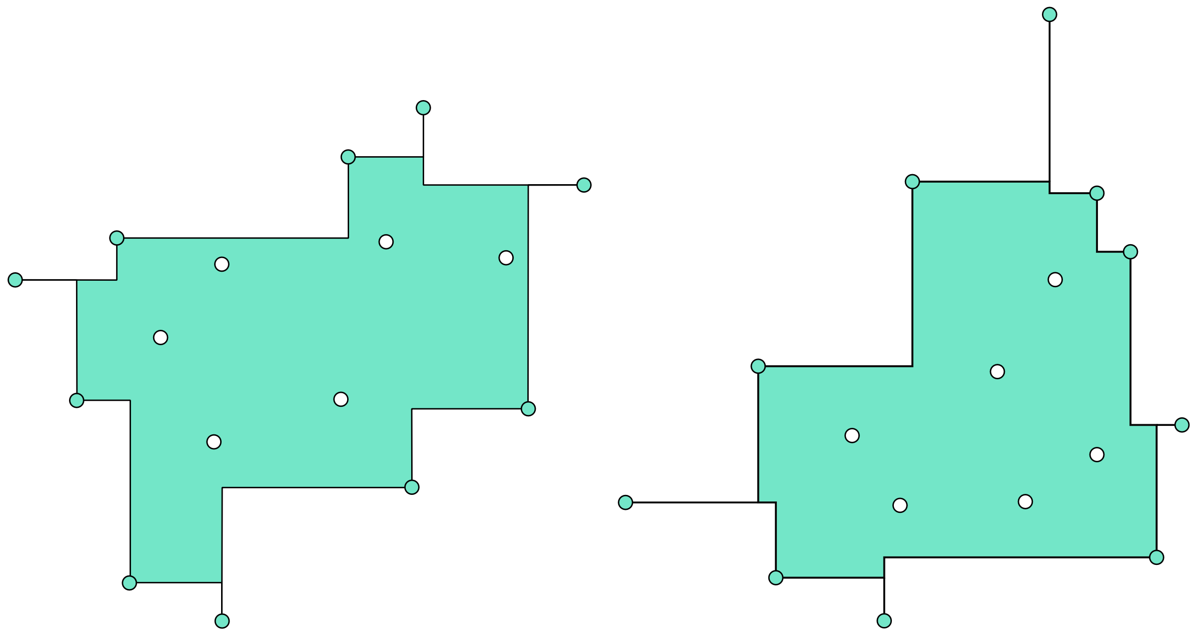


Figure 4.1: Orthogonal Convex Hulls

Recursive Subdivision Using a simple recursive subdivision method, the initial shape is iteratively sliced so that the parent shape is split into two equally sized child shapes. This method works for rectangular bases, but can cause problems when convex or concave polygons are involved. One way to have more influence on the position of the cutting planes is to define percentages for the positioning of the slicing faces in relation to the two-dimensional depth of the shape being divided. This can be re-evaluated from iteration to iteration (figure 4.2), allowing it to adapt to irregular shapes.

Another derivative of the recursive subdivision method can be *recursive quad-division*, which, unlike the conventional method, does not divide the parent surface by a straight line in the X or Y direction, but instead performs a division from the *centroid* of the geometric shape to the center of the edges of the polygon, resulting in at least three quad-edged child polygons. This subdivision method is particularly suitable for irregular convex shapes, as it avoids unwanted intersections. Also of interest is the *recursive bent bisection method*, which instead of straight cutting surfaces introduces a bend in the cutting surface at a random or defined point. This simple modification can lead to a great variety in the resulting design, but does not solve the problems encountered with the simple recursive subdivision method.

Squarified Treemap The squarified treemap algorithm[6] originally comes from the family of data visualisation algorithms. It is based on the creation and ordering of the spatial quantities to be created in a binary tree structure, where the values are ordered in advance to produce the most optimal subdivision. The cutting method is not much different from a conventional recursive subdivision, but in a squarified treemap the resulting spaces in the generated layout are calculated to be as quadratic as possible, which means that the quotient of two adjacent edges is as close to 1 as possible. A major advantage of this method is that the percentage size of the spaces can be defined in advance (figure 4.3),

[6]Marson and Musse 2010, "Automatic real-time generation of floor plans based on squarified treemaps algorithm".

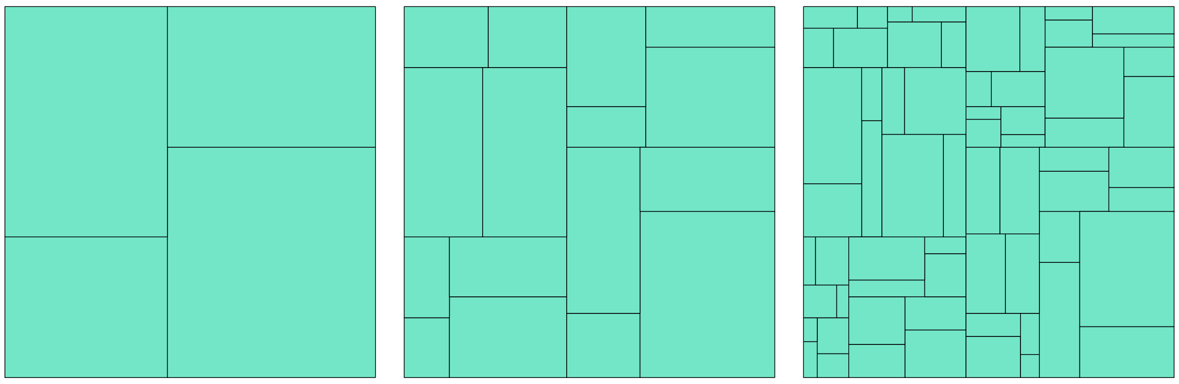
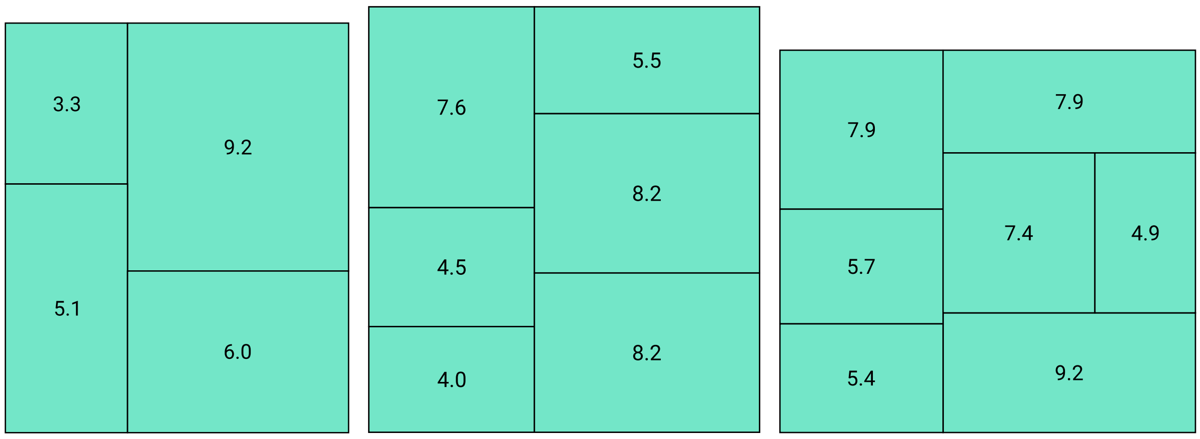

(a) Second Iteration (b) 4th Iteration (c) 6th Iteration

Figure 4.2: Recursive Subdivision



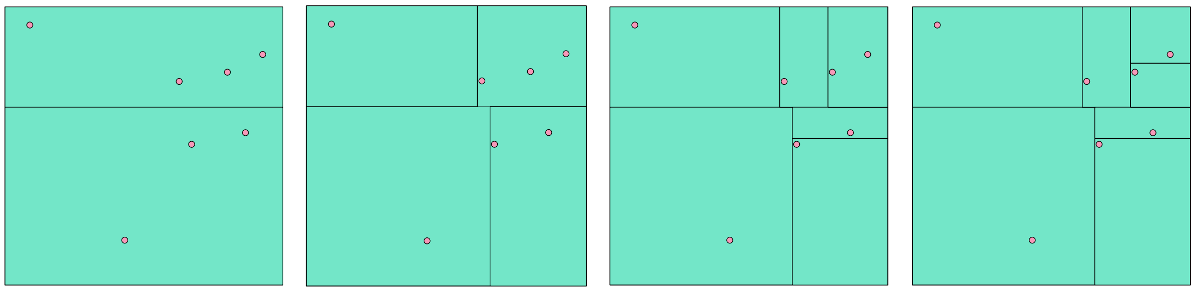
(a) 4 Spaces (b) 6 Spaces (c) 7 Spaces

Figure 4.3: Treemap Subdivision

allowing finer architectural control over the generation process. However, this method only works for rectangular solids and is therefore not suitable for a generation pipeline where polygons are also desired as initial shapes.

K-Dimensional Tree When using a k-dimensional tree[7] for space partitioning, new requirements are placed on the input. In order to perform a successful space subdivision with a satisfactory level of control, a two-dimensional set of points is required to be located on the initial shape. These points will be inside the respective spaces of the fully partitioned surfaces and can thus serve as a basis for a rough local determination of the position of the individual spaces in advance (figure 4.4). This ability to predetermine the approximate division of space by placing points on the two-dimensional body is an advantage over other methods. This can be achieved by simple random or rule-based point placement methods. Furthermore, the algorithm works not just on rectangular solids but also on convex polygons (figure 4.6). This algorithm allows seamless subdivision of rectangles with a particularly low edge ratio (figure 4.5) but does however not allow to define the desired percentage areas of the individual spaces in advance, as in the example of the squarified treemap algorithm.

[7]Knecht and König 2010, "Generating floor plan layouts with kd trees and evolutionary algorithms"; Das et al. 2016, "Space plan generator: Rapid generation and evaluation of floor plan design options to inform decision making".



(a) Iteration 1 (b) Iteration 2 (c) Iteration 3 (d) Iteration 4

Figure 4.4: K-d Tree Subdivision

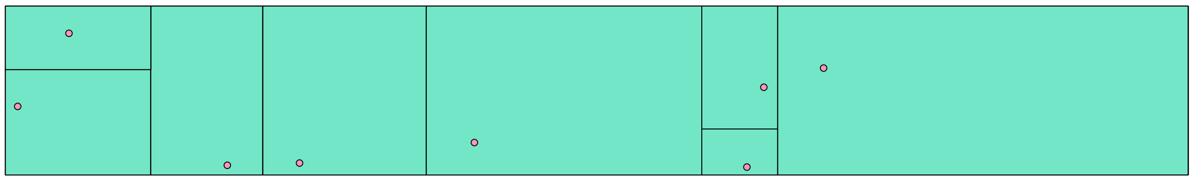


Figure 4.5: K-d Tree Subdivision of Stretched Rectangle

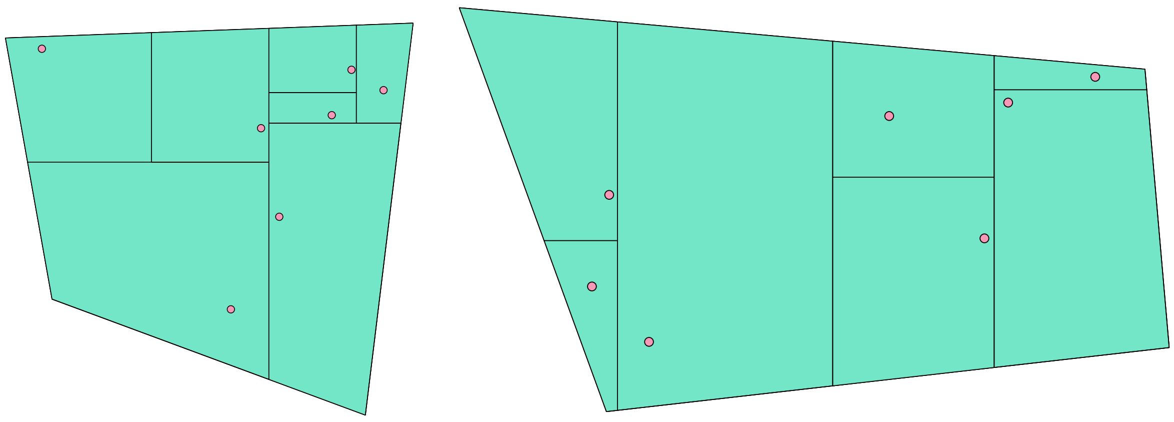
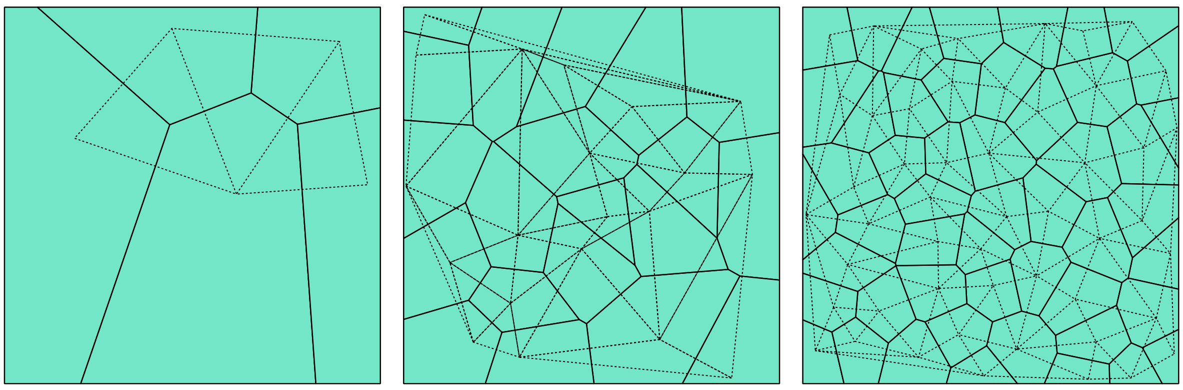


Figure 4.6: K-d Tree Subdivision of Irregular Polygons



(a) 5 Cells (b) 20 Cells (c) 50 Cells

Figure 4.7: Voronoi Subdivision

Voronoi Diagram The initial requirements for Voronoi diagram subdivision methods are similar to those for k-dimensional trees, since they also involve defining points on the initial shapes in advance. However, the body to be subdivided can be any convex as well as concave polygon, allowing for greater diversity in layout generation. The points on the surface are first connected to each other and to the vertices of the base body by a Delaunay triangulation, so that each point is connected to another by an edge. Then the Voronoi diagram is computed so that the points become Voronoi seeds and all points that are closer to one seed than to any other point of the body are bounded by a *Voronoi cell* (figure 4.7). These cells are irregular polygons whose shape is defined by the distance of the triangulated point mesh only. This means that the spaces generated in this way tend to have irregular shapes, which is an architectural challenge, since in construction reality, although it is not impossible to build irregular spaces, a general convention is to design spaces that are as orthogonal as possible.

In order to deform the irregular Voronoi cells into polygons that are more regular, various post-processing algorithms can be applied to the generated design. For example, a *Lloyd algorithm* can iteratively shift the seed points to relax the Voronoi mesh, allowing initially highly deformed cells to approximate more compact geometric shapes. This works by iteratively computing a new Voronoi diagram based on the continuous shift of the Voronoi seed points towards the centroid of each Voronoi cell. It should be noted, however, that this step-based algorithm is more resource intensive in terms of time and computing power. Another way to generate more regular Voronoi cells is to use a *semi-orthogonal Voronoi diagram*, where each cell has at least one orthogonal edge or completely orthogonal cells computed by the *orthogonal Voronoi treemap algorithm*[8]. Another drawback of the Voronoi method is the lack of control over the resulting area dimensions of each cell, as the method is point-based like the k-dimensional tree method. To overcome this problem, weights can be assigned to each point in advance, so that by varying them the radius of each cell can be precisely controlled; these diagrams are then called power diagrams or Laguerre-Voronoi[9] (figure 4.9).

[8]Y.-C. Wang, Lin, and Seah 2019, "Orthogonal voronoi diagram and treemap"; Chatzikonstantinou 2014, "A 3-dimensional architectural layout generation procedure for optimization applications: DC-RVD".

[9]Anuradha, Sabnis, and Thirumavalavn 2008, "Voronoi diagram voro: Application of interactive weighted Voronoi diagrams as an alternate master-planning framework for business parks."

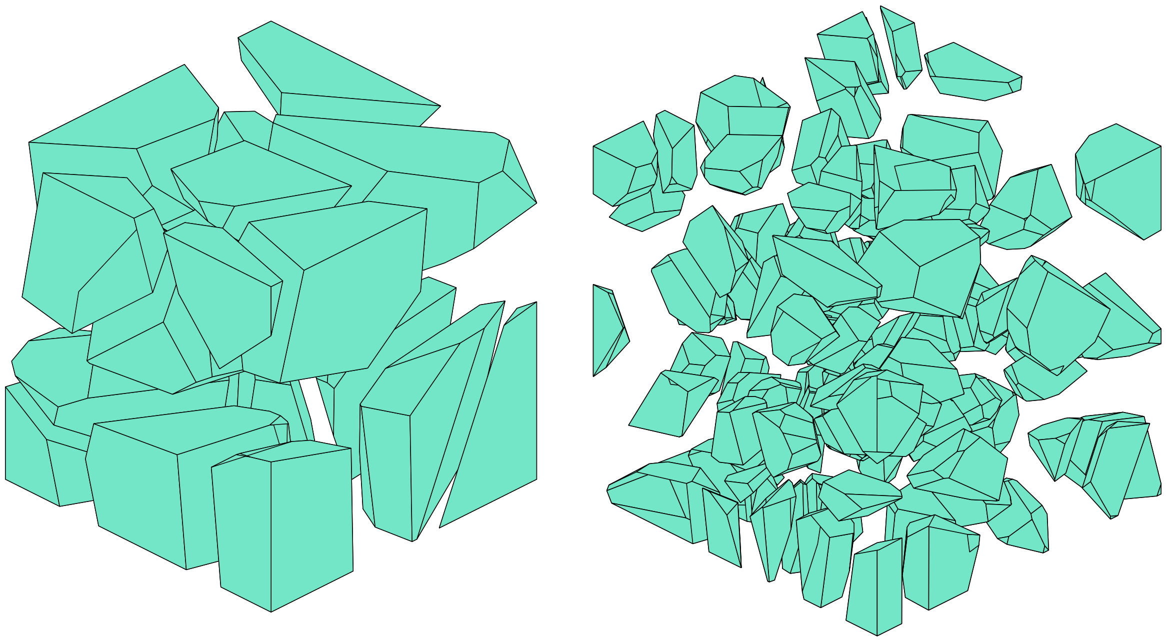


Figure 4.8: Three-Dimensional Voronoi Division

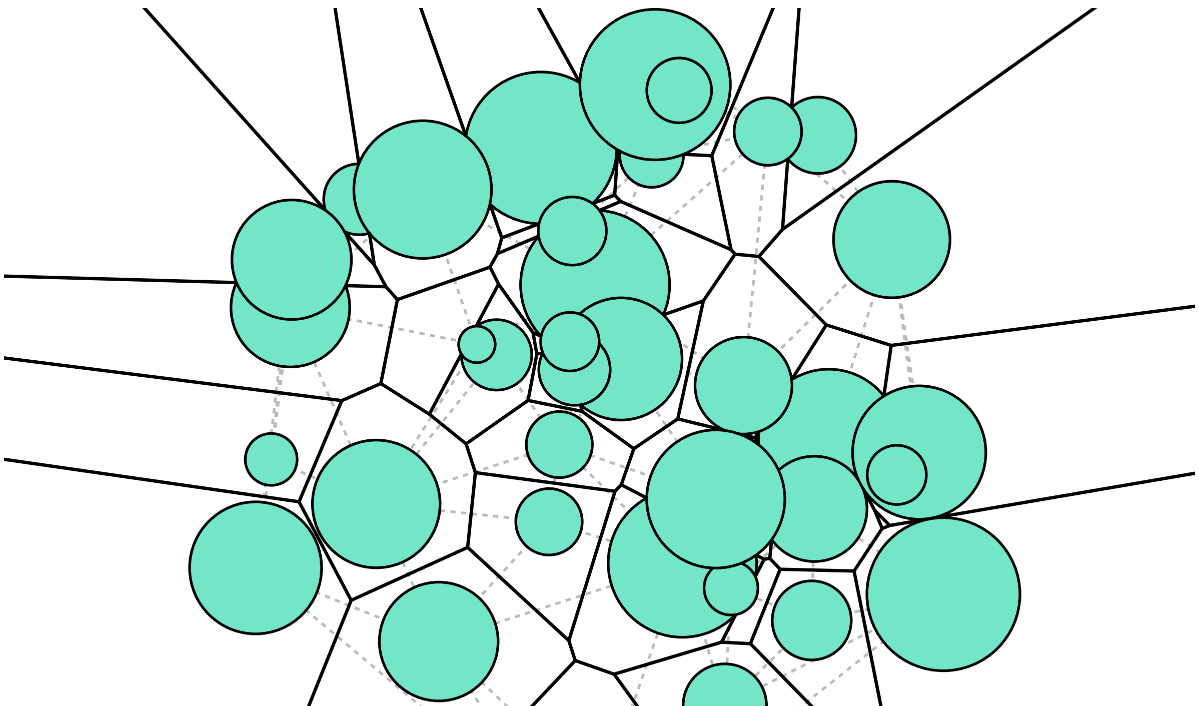
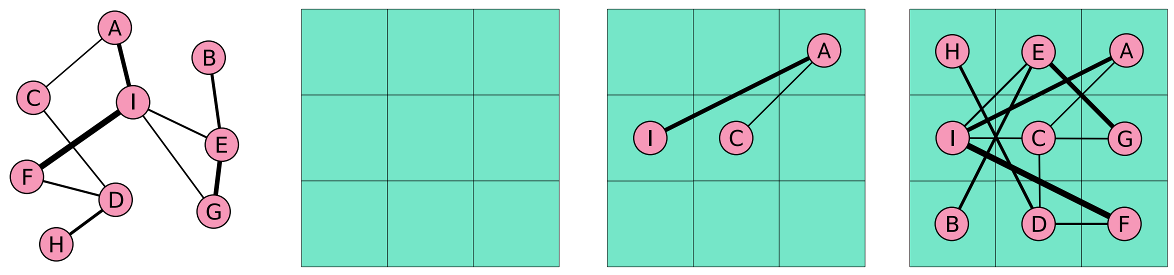


Figure 4.9: Laguerre-Voronoi Diagram



(a) Weighted Graph (b) Grid (c) Iteration (d) Final Placement

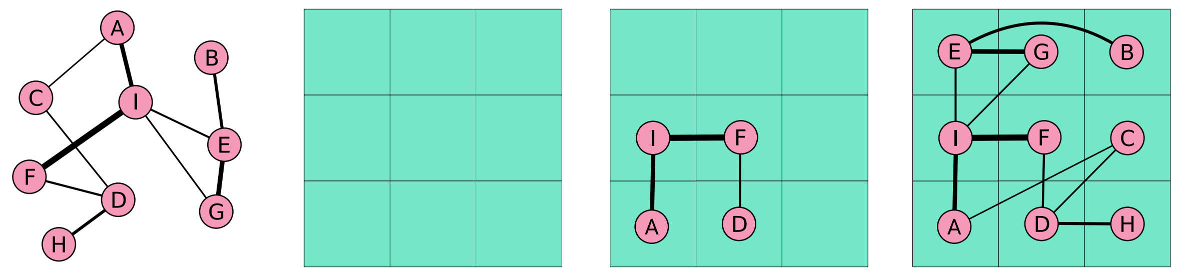
Figure 4.10: Random Grid Assignment

4.1.1.2 Grid Planning

The basis of grid planning algorithms is the placement of a grid on the surface to be subdivided. This basic requirement in itself introduces some issues and difficulties, since the act of grid placement is in itself a subdivision of space. The simplest way of placing a grid is to project a pre-established mesh onto the base to be subdivided, matching the dimensions of the grid to the dimensions of the surface in two dimensions. This only gives satisfactory results if the base is orthogonal and leads to unwanted subdivisions as soon as convex or concave polygons are involved. To satisfactorily subdivide irregular polygons by grids, it is necessary to apply the subdivision methods described above. In particular, the quad-subdivision method (section 4.1.1.1) and the Voronoi subdivision (section 4.1.1.1) are suitable for applying a grid to the desired shape, since both have a high adaptability to the morphology of the basic shape. Thus, it is easily possible to divide a convex irregular polygon by a grid; however, the generated meshes of the grid have an increased probability of inconsistency if the irregularity of the basic shape is higher. This problem can be partially remedied by applying relaxation methods such as the Lloyd algorithm, but complete regularity of the grid cannot be guaranteed.

Random Assignment The basic principle of the grid planning method is to place programs connected by a certain weighted graph (figure 4.10a) in the given grid in such a way that the most optimal arrangement of their spaces is guaranteed according to the programs. Therefore, this is an abstract rather than a geometric partitioning of space, since the basic shape as an initial condition has already been divided into subspaces through the placement of the grid, and it is only a matter of assigning the programs to the individual spaces in such a way that the weighting of their interconnection is respected in the best possible way.

The simplest implementation of this method involves randomly assigning the programs (figure 4.10c) to each of the spaces defined by the grid (figure 4.10b). This method can work iteratively, since after each random assignment an evaluation result can be computed by solving the function to be optimised. Thus, the randomisation is repeated until a defined satisfaction threshold (figure 4.10d) or even the maximum number of iterations has been exceeded. Obviously, this method is purely stochastic and thus represents the least optimised method of grid planning theory.



(a) Weighted Graph (b) Grid (c) Iteration (d) Final Placement

Figure 4.11: Iterative Grid Assignment

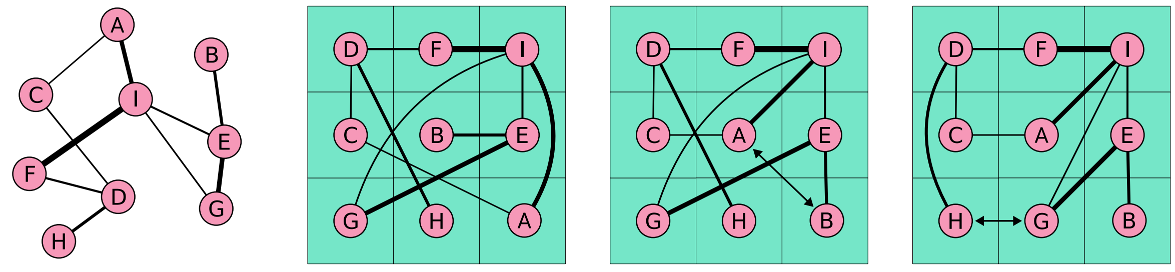
Iterative Assignment A more optimised approach than the random assignment method is the iterative placement of programs[10]. Here, a random program is first assigned to an equally randomly selected slot. Then the program with the greatest weight associated with the placed element is selected. This is then positioned as close as possible to the already placed program in the grid (figure 4.11c). The procedure is deterministic because as long as the same basic conditions are given, such as the same programs, their connection weights and the initial program, the same program order will always be generated. Although this method achieves more optimised results than random positioning (figure 4.11d), its deterministic, iterative structure carries a high risk of generating suboptimal arrangements, since the algorithm does not use methods that consider a variation of all optimisation function variables at the same time.

Random Swap The best method to solve the arrangement problem without using established optimisation algorithms is the random swap method. Its operation is relatively simple and starts with a random assignment (figure 4.12b) of all programs to the respective spaces defined by the grid. Then the optimisation function is solved and the result is stored. Then the position of two randomly chosen programs is swapped (figure 4.12c) and the new solution of the optimisation function is compared with the stored result of the previous constellation. If this result is an improvement with respect to the defined target value, the current constellation is kept and the process starts over again (figure 4.12d). These steps are repeated until the previously defined target threshold is exceeded and the final layout is generated. This method achieves more optimised results than the previously mentioned methods; however, finding an optimal solution is still not guaranteed and, due to its stochastic nature, the computational and time requirements of the algorithm cannot be predicted. Therefore, the use of state of the art optimisation methods (sections 2.3.2 and 4.1.1.3) is recommended for a concrete application of the grid planning method.

4.1.1.3 Aggregation

Another method that differs significantly from the previous ones is the family of aggregation methods in automatic floor plan design. A key feature of these methods is the ability to generate floor plans of irregular shape (figure 4.13), such as balconies that extend beyond the rectangular boundary of the building's cross-section, or courtyards that split the floor plan by creating a void in the center of the layout. Aggregation methods do not require predefined outlines to initiate the generative process, but rely on the formal definition of individual spaces in advance, which are then combined into an overall plan

[10]Egor et al. 2020, "Computer-aided approach to public buildings floor plan generation. Magnetizing Floor Plan Generator".



(a) Weighted Graph (b) Assignment (c) Swap 1 (d) Swap 2

Figure 4.12: Random Swap Grid Assignment

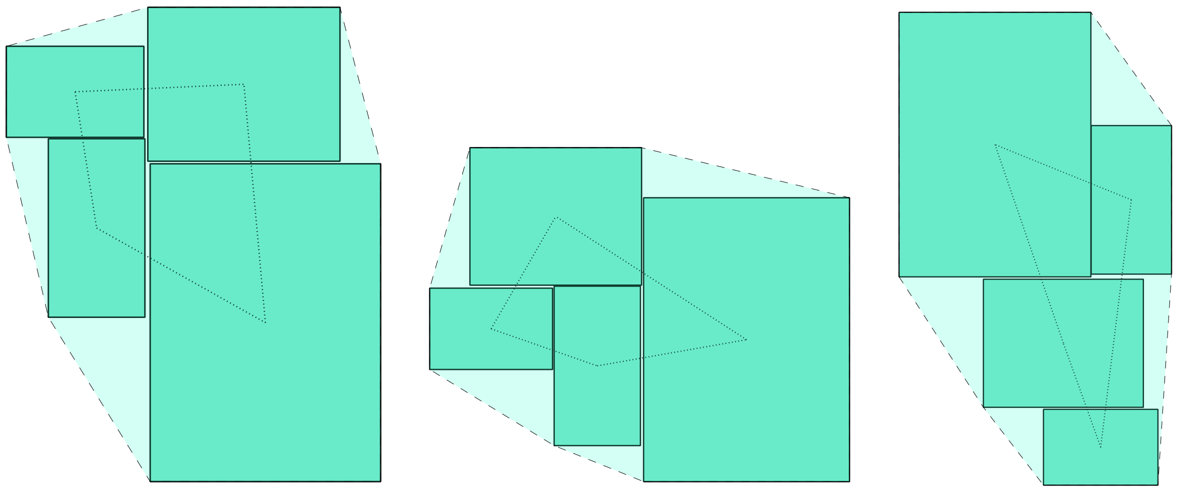


Figure 4.13: Aggregation Results

through various methods. Apart from the increased formal variety that this method allows, the possibility of defining the formal aspects of the spaces to be placed in advance is also a significant advantage of aggregation methods. However, due to their specific logic, it is always necessary to evaluate a function to be optimised in order to aggregate the spaces according to certain conditions. This means that in most practical applications of these methods it becomes necessary to apply optimisation algorithms[11]. In general, the use of such algorithms inevitably means an increased demand for computing power and thus an increased runtime, which can lead to complications when trying to integrate this type of automatic space planning methods into a parametric dataset generation pipeline.

Optimisation Algorithms In order to successfully perform an aggregation of the initially defined bodies, the individual space contours must be shifted on the same two-dimensional plane in such a way that they are arranged in the most optimal manner possible. But what does optimal mean and how can it be evaluated? Since rational mathematical methods are involved, it is essential to create an evaluation scale in order to be able to compare one generated layout with another in terms of its degree of optimisation. In other words, it is necessary to establish a universal optimisation evaluation formula that assigns a value to each individual layout which must be maximised or minimised in

[11]Nisztuk and P. Myszkowski 2019, “Tool for evolutionary aided architectural design. Hybrid Evolutionary Algorithm applied to Multi-Objective Automated Floor Plan Generation”.

order to achieve an optimum in the design evaluation.

This in itself can be a great challenge, as some of the criteria are highly subjective, purely architectural and extremely variable both locally and over time. This difficulty in defining an optimal design in architecture is the core problem of automatic floor plan generation[12]. However, since the present experiment is not concerned with the generation of individually optimised plans, but rather with the generation of the broadest possible architectural dataset, it is possible to include mainly objective design criteria as variables in the function to be optimised. Specifically, this means that the formula evaluates how close the individual rooms are to each other, but without overlapping, since unwanted gaps in an architectural floor plan are just as much a design flaw as an overlap of defined rooms. Furthermore, weighted graphs, which formulate the desired local and topological relationships between spaces as a formula to be optimised, can be used to evaluate the implementation of programmatic requirements in the final design. The parameters to be varied are thus represented by the two-dimensional coordinates of each space in the function, and the evaluation of the layout is solved by computing the weighted or unweighted graph.

In order to find the optimum of the formula defined in this way, optimisation algorithms and methods are applied, which exist in a great variety[13] in mathematical, physical and computer science contexts. Among the better known methods for the successful optimisation of multivariable non-linear functions are the evolutionary algorithms[14], which, based on the natural model, rely on concepts such as populations, natural selection, crossover and mutation. Or the Simulated Annealing Algorithm, which, as a probabilistic method, also provides strategies to avoid local maxima by continuously reducing the probability weighting to find a global maximum of the function to be optimised. Of course, machine learning methods such as reinforcement learning can also be used for function optimisation.

Agent Based A slightly different approach to the definition of a function to be globally optimised is the establishment of independently acting agents. Each of the predefined spatial elements (figure 4.14a) is given a certain autonomy as well as an evaluation capacity, so that the individual agents can perform different local transformations (figure 4.14b) on a two-dimensional level without deforming the space and find their optimised position in the overall layout by solving an optimisation function corresponding to the individual variables. Possible transformations include variation of the X and Y coordinates of the shape-defining points and rotation of the whole space to match the rotation of another body by calculating their degree of alignment.

Physics Solver Furthermore, the optimisation formula can also follow physical laws, so that the problems to be solved can be defined and calculated as physical constraints. In the concrete application for the arrangement of defined spatial bodies to an optimal floor plan layout, this means that the weighting of the individual graph edges of the topological graph, which represents the relationship of the individual spaces to each other,

[12]Nisztuk and P. B. Myszkowski 2019, "Hybrid evolutionary algorithm applied to automated floor plan generation".

[13]Pena et al. 2021, "Artificial intelligence applied to conceptual design. A review of its use in architecture".

[14]Caetano, Luís Santos, and Leitao 2020, "Computational design in architecture: Defining parametric, generative, and algorithmic design"; Grzesiak-Kope, Strug, and lusarczyk 2021, "Evolutionary methods in house floor plan design".

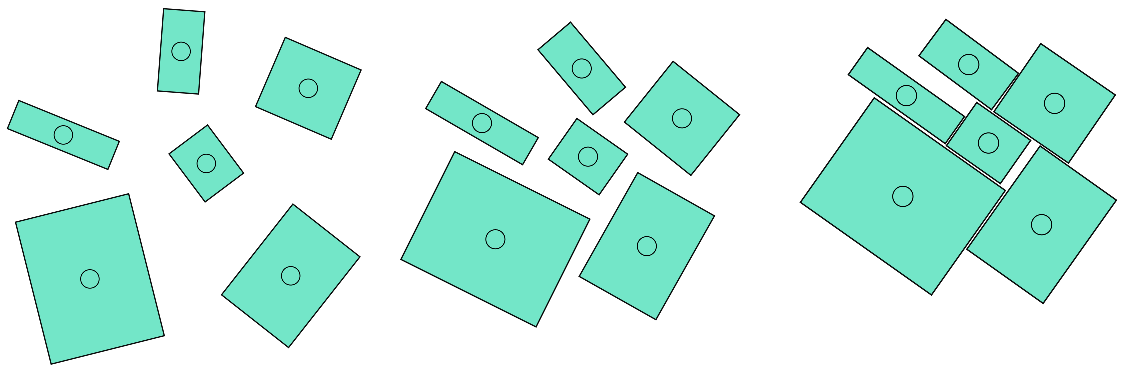

(a) Initial Position (b) Iteration (c) Final Layout

Figure 4.14: Agent Based Aggregation

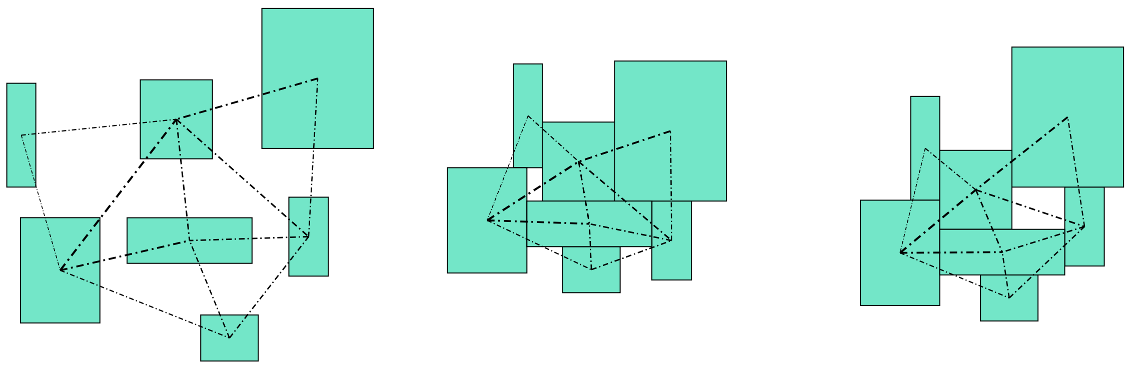

(a) Initial Position (b) Final Layout I (c) Final Layout II

Figure 4.15: Spring Force Aggregation

is simulated by physical spring forces corresponding to the weighting of the edges (figure 4.15). The individual spaces in this physical simulation are defined as solid bodies, which are therefore not penetrable and whose motion is limited to two dimensions. To add some variety to the generation process, the deterministic nature of such a simulation needs to be complemented by randomness. This is achieved by arbitrarily positioning the individual rooms in two-dimensional space as the starting position of the simulation or regulating the strength and intensity of the simulated forces.

Refinement Strategies The different aggregation strategies share similar methodological advantages and disadvantages. As mentioned above, the arrangement of the individual subbodies, by virtue of their formal definition in advance, allows for a high degree of design diversity and a relatively accurate representation of architectural design practice. However, the fundamental drawbacks of this logic should also be noted. Due to the necessary application of optimisation algorithms or force simulation in the case of the physical solver method, any aggregation strategy requires a large number of iterative, relatively complex calculations, which are reflected in a high computing power requirement and consequently in a considerable time consumption.

However, the most serious drawback of such methods is the lack of a possibility to guarantee a gapless generation of the final floor plan. Since all methods consider the distance function as part of the formulas to be optimised, the distances will tend towards zero, but without a certainty of reaching exactly zero. This makes the application of *gap-removal*

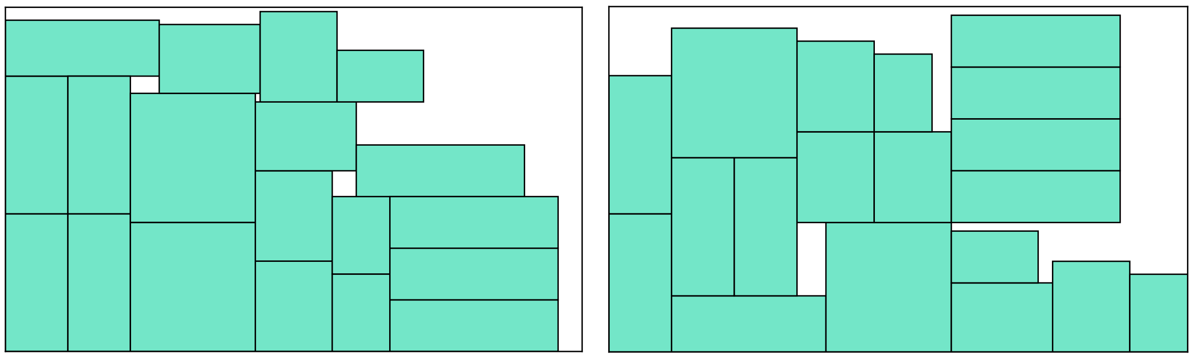


Figure 4.16: Shape Packing Algorithm

algorithms after the generation process inevitable. A slightly modified version of the shape-packing algorithms[15] (figure 4.16) can be applied here, which consists of moving the individual spaces in such a way that these gaps are reduced. However, this is only a minimisation of the individual gaps and not a complete resolution of them.

In a final step, it is therefore common to assign the individual gaps to the areas of the different rooms in a random or regularised manner. This allows for the complete elimination of free spaces, but has the consequence that the morphology of the spaces no longer exactly matches the previously defined shape, and thus problems such as lack of orthogonality may arise.

4.1.1.4 Shape Grammar

In contrast to subdivision or aggregation methods, the so-called shape grammar[16] (section 2.2.3), which, by defining generic geometric or topological rules, can generate a large number of different design layouts that always follow the same rules. In the shape-grammar approach, the initial element can be a single room or the floor plan to be subdivided, providing a greater variety of starting conditions than the previously described methods.

Since the rules of this method can be geometric transformations, the positioning of new shapes or even the generation of new shapes by intersecting given bodies, both subdivision and aggregation processes can take place within the framework of shape grammar. It thus represents a promising methodology for the automatic generation of floor plans[17]. However, it can be observed that its application in architectural practice is limited to a few project prototypes and has not found a place in everyday design. The reason for this could be the high complexity of its structure, since a clear, precise and unambiguous formulation of the different rules requires a fundamental understanding of basic geometric operations and patterns. Therefore, no meaningful application of shape grammar could be found in this research, as scientific or documentary sources on this method are rare and of limited quality and depth.

The various implementations of shape grammar processing tools all require a digital component that translates the formulated rules into machine processes, the so-called *interpreter*, which is thus an essential component of the method. There are several widely differing implementations of this form grammar interpreter, and relatively little insight

[15] Jabi, Grochal, and Richardson 2013, "The potential of evolutionary methods in architectural design".
[16] Hong and Economou 2022, "Five criteria for shape grammar interpreters".
[17] Ruiz-Montiel et al. 2013, "Design with shape grammars and reinforcement learning".

into its function, making it difficult to use legitimately.

However, a renewed interest in shape grammar methods in recent years shows promise in new interpreter implementations[18] which, in addition to the basic geometric rules, can potentially respect topological relations and thus certain programmatic requirements of the designer.

4.2 Parametric Framework

The advantages and drawbacks of the individual methods and algorithms were compared using various criteria in order to make a final selection of the methods to be integrated into the synthetic dataset generation process. It was decided to use a modified recursive subdivision and a Voronoi-based subdivision due to their comparatively low complexity and computational requirements. Furthermore, an intermediate method of the two algorithms mentioned above is used, where the layout generated by the recursive subdivision is used as input for a Voronoi diagram subdivision[19], so that the centers of the existing spaces become the Voronoi seed points. During the research process it was not possible to find aggregation methods that did not exceed the computational and thus the temporal capacities. For the same reasons, it was not possible to implement a reliable method based on shape grammar rules.

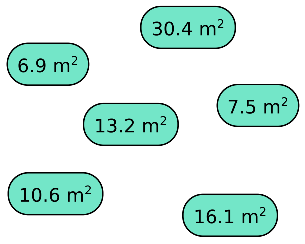
Now that the three methods belonging to the family of division algorithms had been selected, the data required for the correct execution of the algorithms could be clearly defined. First, a basic geometric shape was needed to represent the outline of the floor plan to be divided. The convex hull method for generating irregular polygons and an algebraic method for generating randomly sized rectangles and squares were combined with a method for generating slightly deformed rectangles in an outline generation method. Since the goal is to automatically generate as diverse a dataset as possible, the individual outline morphologies were assigned different probabilities of being selected as the initial shape generation method for the current generation step. In addition, the dimensions of the rectangles and squares, the placement of the convex hull points and the deformation of the sides of the rectangles were carefully given a certain degree of randomness, but without running the risk of taking architecturally unrealistic forms, such as very low aspect ratios of the rectangles or very sharp angles.

Another important input to the floor plan generation methods was the pre-specification of the size ranges of the respective rooms in relation to the total number of rooms selected in the current step of the generation process (section B.1.1). Thus, by randomly selecting items from this list, the generation algorithm was allowed to obtain information about the program of desired rooms and their respective maximum and minimum sizes (figure 4.17a). The number of rooms to be placed is determined by the total area of the arbitrary initial shape (figure 4.17c) and is equally communicated to the generation algorithm.

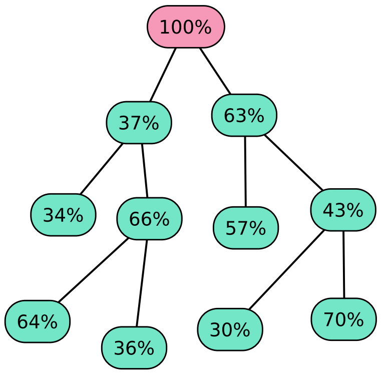
One challenge was to create an iterative subdivision strategy that would allow the walls to be placed in such a way that, in the final layout, the individual rooms would have their exact percentage size corresponding to the generated outline. To achieve this, it was first necessary to create binary trees (figure 4.17b), each of which contained the desired size percentage of the parent rooms as child leaves. A *downhill simplex algorithm* was

[18]Muslimin 2023, "Experience Grammar: Creative Space Planning with Generative Graph and Shape for Early Design Stage".

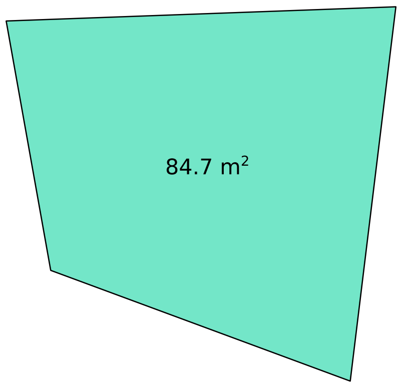
[19]Balzer and Deussen 2005, "Voronoi treemaps".



(a) Amount and Room Sizes

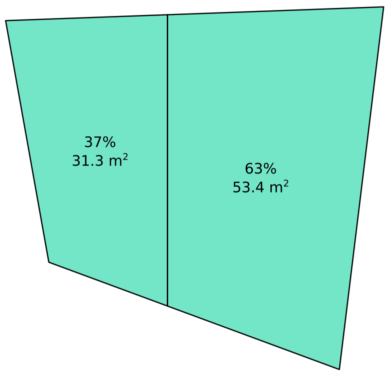


(b) Constructed Binary Tree

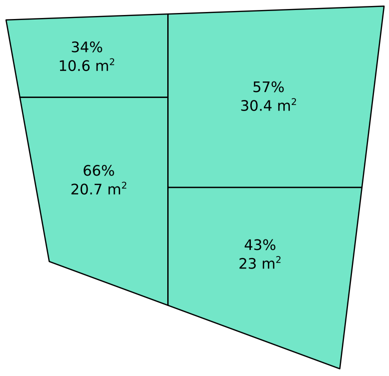


(c) Area of Outline

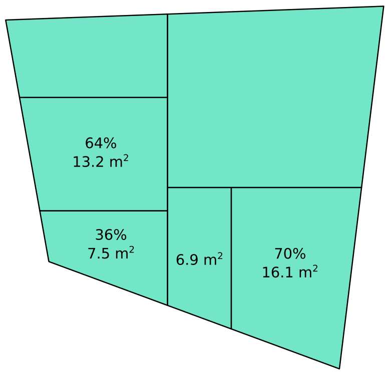
Figure 4.17: Binary Tree Computation with Percentages



(a) First Iteration



(b) Second Iteration



(c) Third Iteration

Figure 4.18: Space Division by Percentages

then used to optimise the placement of the cutting planes and the weighting of each Voronoi cell so that the percentage subdivision (figure 4.18) specified by the binary tree was maintained as closely as possible.

The final step in the geometry generation process was the transformation of the previously purely two-dimensional apartment floor plans into three-dimensional cell elements by topological operations through the extrusion of the individual surface elements along the Z axis, in order to generate a three-dimensional BIM model. This whole process was implemented in the programming language Python and was essentially enabled by the use of certain libraries such as *TopologicPy*, *NumPy*[20], *SciPy*[21] and *PyVoro*[22].

A particularly common method of parametric geometry generation in architecture is the use of visual scripting languages (figure 4.19), which allow direct visualisation of the results of the individual substeps of a parametric generation pipeline and are thus more intuitive to use than abstract programming languages. In this thesis a large amount of experimentation has been done in the visual scripting interface *Sverchok*[23]. This is an add-on written in Python for the 3D manipulation software *Blender*[24]. It quickly became apparent that such an interface has strong advantages during the experimental phase, but

[20]Harris et al. 2020, "Array programming with NumPy".
[21]Virtanen et al. 2020, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python".
[22]Rycroft 2007, "Multiscale modeling in granular flow".
[23]Nortikin 2013, *Sverchok*.
[24]Community 2018, *Blender - a 3D modelling and rendering package*.

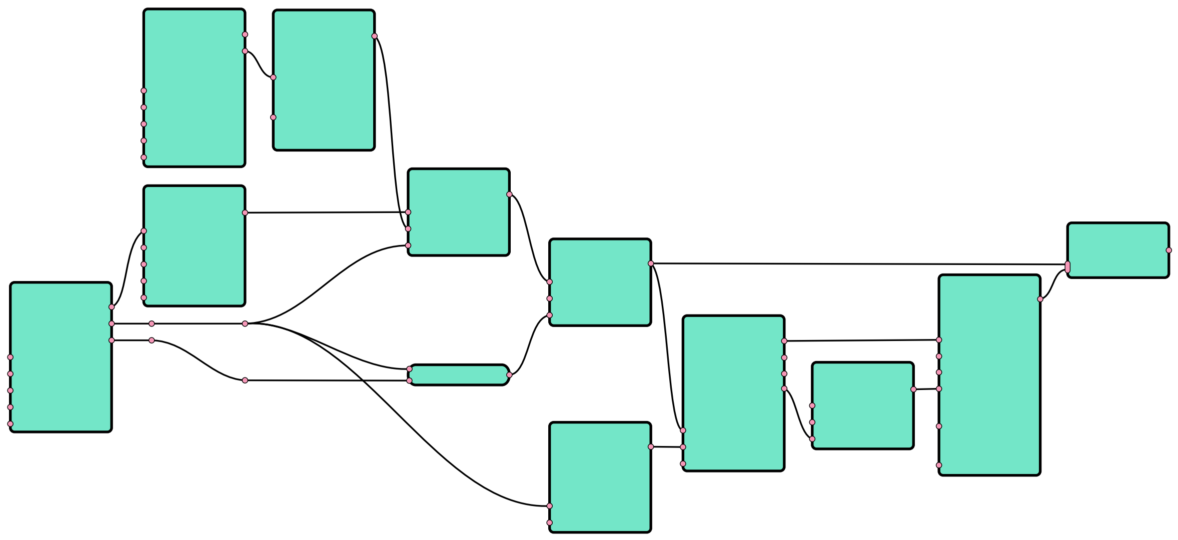


Figure 4.19: Visual Scripting Language

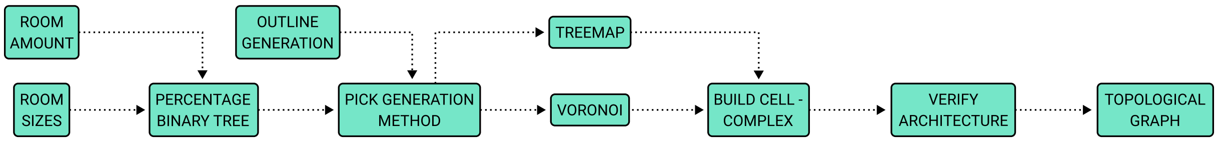


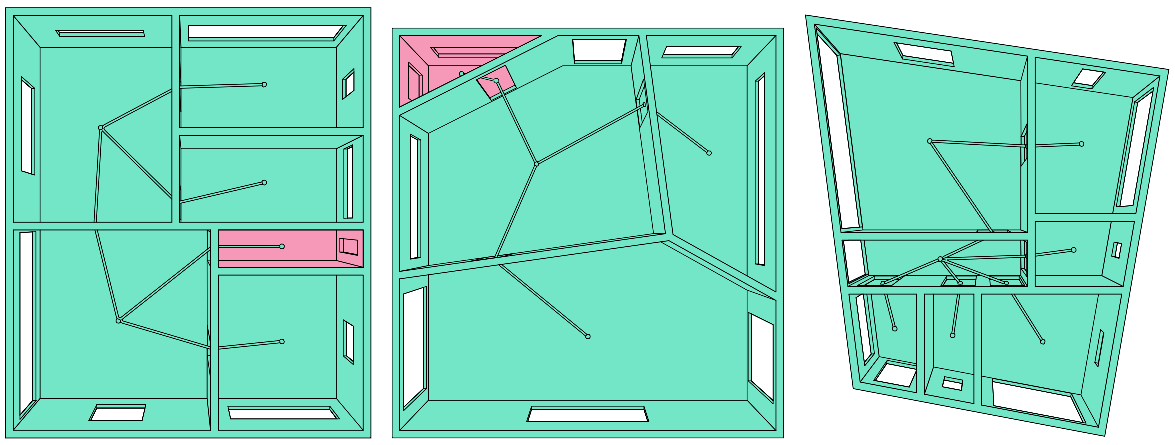
Figure 4.20: Structure of the Framework

is not suitable for the generation of large datasets due to a high memory load and a lower portability as well as stability. Therefore, a pure Python implementation was preferred for the final generation. An overview of the whole dataset generation pipeline is shown in figure 4.20.

4.3 Architectural Rules

Throughout the construction of the parametric structure, a great deal of attention has been paid to the constant verification and guarantee of the architectural feasibility of each floor plan, since the ambition was to create a dataset that claims to reflect, to a certain extent, the state of architectural reality. The first architecturally regulated instance consisted in defining the various architectural programs, such as living room, bathroom, bedroom, toilet and utility rooms, and assigning them to the corresponding apartment types (table B.1 and table B.2). For example, a ten-room apartment has four bedrooms, a living room with a kitchen and two bathrooms, a toilet and two utility rooms. A three-room flat, on the other hand, only has a living room with a kitchen, a bedroom and a bathroom. This list was then used to determine the maximum and minimum room sizes for each individual apartment constellation and room.

In addition, architectural control functions were implemented in the process of generating the apartment outline to be subdivided, which made it possible to eliminate unrealistic or particularly atypical plan morphologies in advance. These functions are also used to control the maximum and minimum size of the total area of the residential object to be generated. Here the values $27.5m^2$ have been set as the minimum acceptable size and $136.5m^2$ as the maximum acceptable size. The minimum acceptable aspect ratio of these plans was set to 0.7. For the non-orthogonal plans, a topological compactness measure is



(a) Low Aspect Ratio (b) Sharp Angles (c) Good layout

Figure 4.21: Layout Evaluation

applied to make an appropriate selection.

The next architectural control instance is represented by a function that analyses the generated apartment plan in terms of architectural criteria and decides whether it is an acceptable result (figure 4.21c) or whether the generation process needs to be repeated for this iteration. For example, this function verifies whether there are exterior or interior walls in the design that are unrealistically short in relation to the size of the apartment, or whether there are rooms whose *degree of compactness* is below an acceptable value, namely too long or too deformed (figure 4.21a and 4.21b). If, despite the implemented optimisation function, a floor plan generated per iteration produces rooms that are too small or too large in relation to the other rooms or even the size of the apartment, these plans are equally classified as architecturally unacceptable and sorted out. This architectural control, both before and after the application of the algorithms, makes it therefore possible to synthetically generate BIM models that represent the architectural reality in the best possible way.

4.4 Post-Processing

The three-dimensional apartment models, which constitute the final result of the previous steps, consist of rooms with their respective programs and their arrangement in an apartment constellation. However, in order to be able to speak of true BIM models, important architectural elements are missing, such as doors, which represent the topological connection between the individual rooms, and also windows, which represent the connection to the surrounding medium. These elements can be summarised under the term apertures and are represented and stored in the geometry engine used as subsurfaces associated with their respective walls. In order to be able to generate them automatically, certain topological tools have to be applied so that a logical placement of the individual elements becomes possible.

Firstly, in order to place the door elements between individual rooms, a graph object is created which connects the individual rooms in such a way that all topologically adjacent rooms are connected by edges (figure 4.22a). The vertices of this network represent the individual spaces. Once this graph has been determined, a Minimum spanning tree of the respective network can be computed, which represents the subgraph of the original graph connecting all vertices without forming any cycles (figure 4.22b). Furthermore, this tree

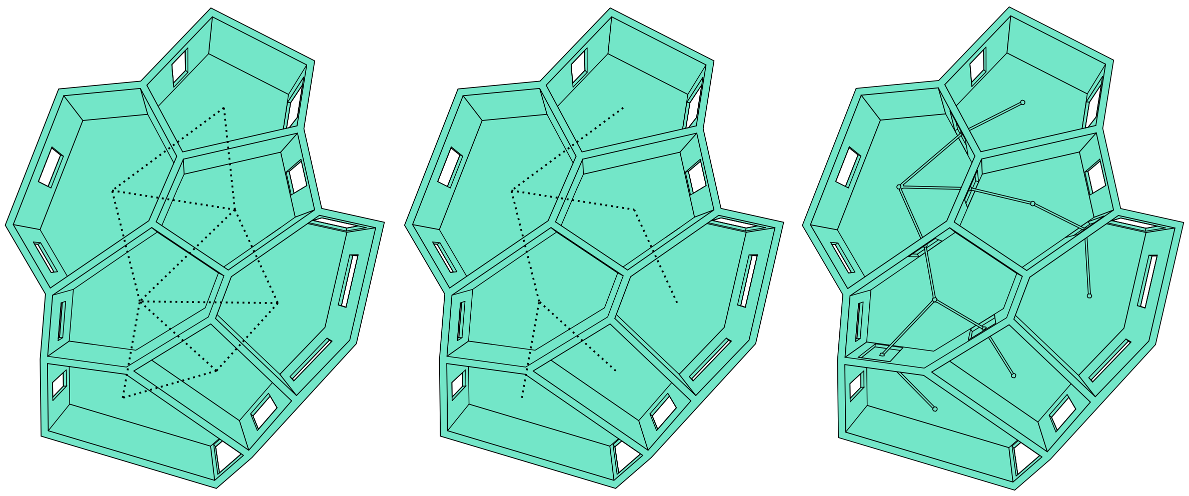

(a) Delaunay Triangulation (b) MST Computation (c) Door Placement

Figure 4.22: Door Generation Process

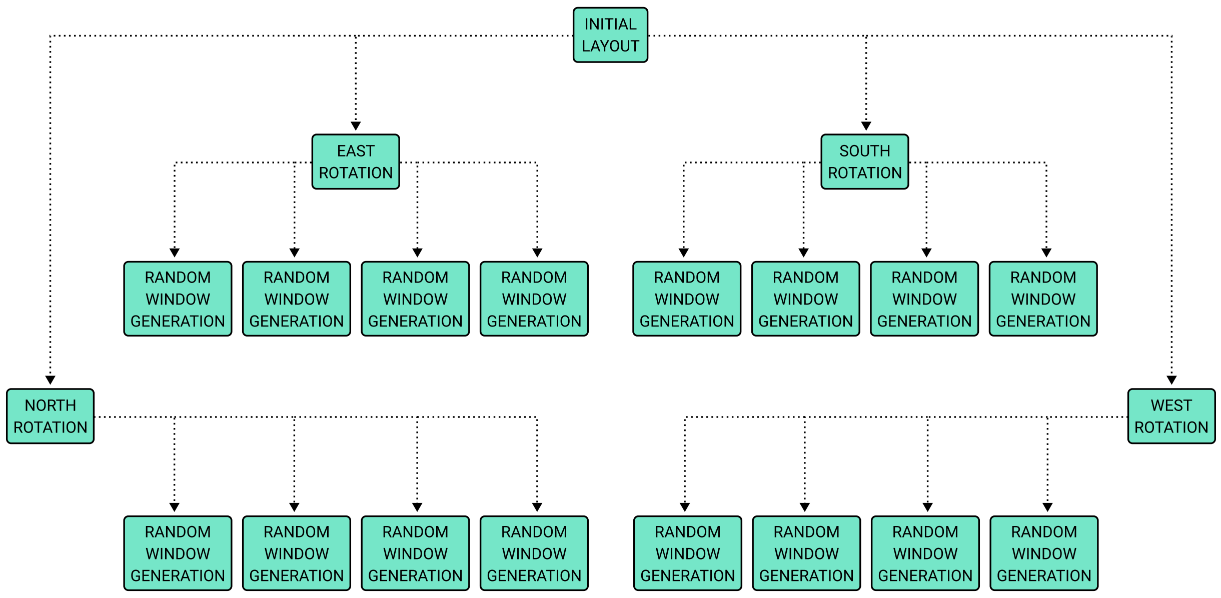


Figure 4.23: Layout Variation Process

is computed in such a way that the sum of its edge weights is as small as possible.

The next step is to relate this graph to the basic geometry by topologically identifying the walls that connect two spaces in the Minimum spanning tree. Finally, each of the wall objects found in this way is assigned a door aperture in order to create a more architecturally realistic three-dimensional model (figure 4.22).

Once the doors have been placed, the window aperture generation can begin. First, however, to provide greater variety in the dataset, each generated apartment model is duplicated and rotated along the Z-axis in 90 degree increments to create four new models that differ only in their orientation. This allows the same apartment structure to be stored in the training dataset with an orientation in all four cardinal directions. Once this duplication and rotation operation is complete, four different window arrangements are generated for each rotated apartment model, which are then stored as final geometries in the dataset (figure 4.23).

The method of generating the individual window apertures is relatively simple, as it is basically a matter of randomly calculating the *glazing percentages* per wall. Since this is the preparation of models for an energy simulation, the shape of the windows is not critical; only their orientation and size play a significant role in the energy efficiency calculation. Similarly, each exterior wall is given a probability of not containing a window aperture in order to remain faithful to architectural reality, since of course not every exterior wall of a building necessarily carries a window.

4.4.1 Information Annotation

The final geometry model has thus been successfully generated, but it still requires the annotation of certain information in order to represent a valid BIM model and start the next iteration. This information is fundamental to the creation of this dataset, since the aim is the computation of knowledge graphs that contain certain dimensional, positional and evaluative data. To understand how such information can be stored in the model, it is necessary to look at the structure of the geometry files. In the generation pipeline presented here, the data is stored using *JavaScript Object Notation (JSON)* file formats as the frame structure (section B.3.1) and the geometry information is stored using *Boundary Representation Strings* (Brep-Strings) (section B.1.2).

The JSON structure is a text-based file format which, like the IFC file format mentioned earlier (section 2.1.2.1), allows information to be related in a hierarchical and structured way, so that specific key-value pairs can be assigned to individual geometry elements and stored and read in a structured way. First, the Brep string of the geometry is stored as a value of the 'brep' key at the top level of the structure, and an associated *dictionary element* is added containing information about the type of element, the number of rooms, the different room types and the identification number of the geometry. The individual rooms are then also assigned dictionary elements by referencing selectors containing information about the element type, the total area in m^2, the respective program of the room and the identification number of the room. This structured form of data representation also makes it possible to store aperture elements such as doors and windows as separate geometries, and to associate individual dictionary objects with them, containing information about their element type, their area in m^2, their orientation, and their architectural role.

4.4.2 Energy Performance Simulation

The generated and annotated BIM model now contains information about its geometry, topology, program and dimensions, but essential information about its energy efficiency is still missing. To obtain this data, careful energy simulations have to be performed for each individual apartment model. Furthermore, as the aim is to develop an automated data generation process, these simulations need to be parameterised and integrated into the general pipeline in order to automatically add the calculated energy-related information to the JSON file of the BIM model. For this purpose, the *Openstudio*[25] software tool is integrated into the parametric framework, providing, among other functionalities, simplified access to the *EnergyPlus*[26] energy simulation software.

In order to carry out a thorough energy simulation in *EnergyPlus*, a variety of information is required, presented in different file formats. First, a so-called *EnergyPlus Weather*

[25]Guglielmetti, Macumber, and Long 2011, "OpenStudio: an open source integrated analysis platform".

[26]Crawley et al. 2001, "EnergyPlus: creating a new-generation building energy simulation program".

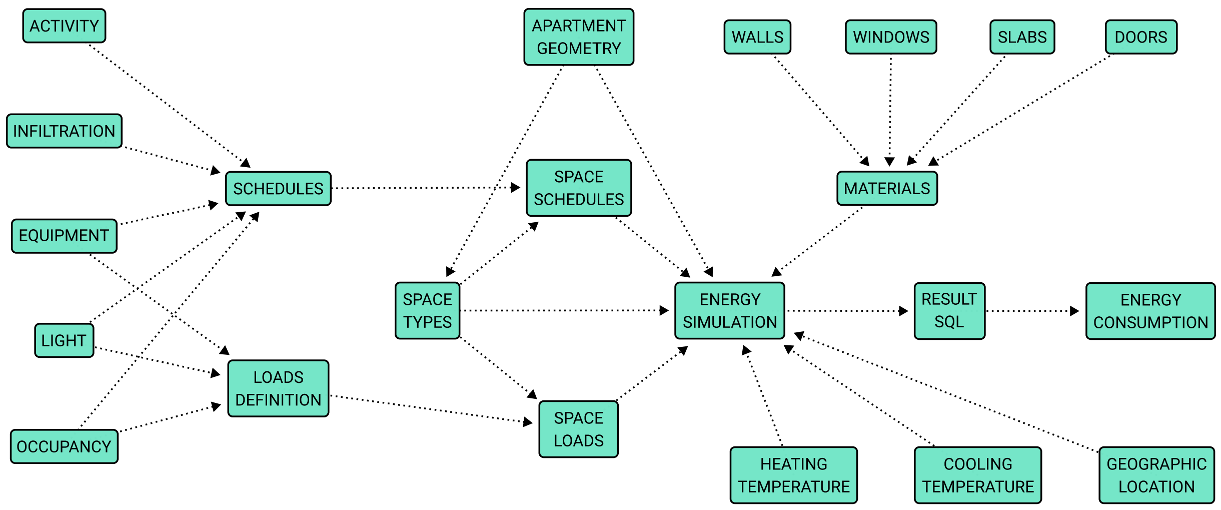


Figure 4.24: Energy Simulation Process

Format (EPW) file is required, which provides general, historical and detailed information about the climatic conditions of the chosen environment for the simulation. A design day file is also required, which provides information about the specific *American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE)* design conditions. However, the most important component of the energy simulation is the creation of a *OpenStudio Model*, stored in a human readable text format called OSM. In the case of this work, this is a partial simulation model that contains basic information about the materials and composition of each element of the residential model (section B.1.3), as well as individually created operating and loading schedules that comply with the ASHRAE standards. These schedules provide a variety of different information for each individual room program, namely the number of people over time, the technical equipment of the room, the heat emitting light sources and much more.

Finally, all these simulation conditions are merged with the generated BIM models and their stored information to initiate the energy simulations. The result of this simulation is presented in a common database form, *Structured Query Language (SQL)*, and can be easily and accurately queried. This now allows energy efficiency information to be added to the original BIM model. Many different energy simulation values can be included, providing information on a variety of different parameters and performances[27]. However, in this work the initial focus was on the total value of energy consumption per area in MJ/m^2, which was then automatically added to each of the BIM models individually. The described energy simulation process is shown in figure 4.24.

4.4.3 Data Mapping

Although the individual BIM models thus generated are now complete, the stored information still needs to be partially processed and manipulated to produce a qualitative dataset. Each room and window element already carries numerical values about their respective programs, dimensions and orientations, corresponding to the cardinal points, but these must be converted into distinguishable classes, due to the way graph machine learning methods work. For this purpose, 91 classes are created, which describe the type

[27]Aksin and Selçuk 2021, "Use of Simulation Techniques and Optimization Tools for Daylight, Energy and Thermal Performance: The case of office module (s) in different climates".

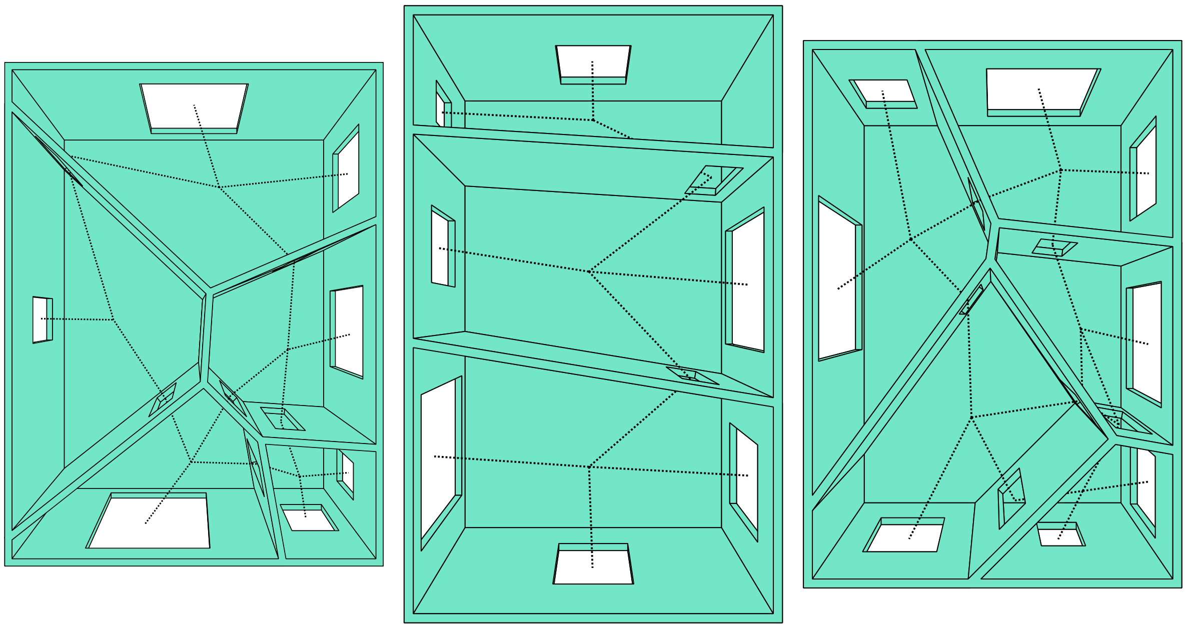


Figure 4.25: Final Layouts with Corresponding Graphs

of window or the respective program of the rooms, a relative size ranging from XXS to XXL and the respective cardinal direction of the windows from north to northwest in clockwise direction (table B.9). The cardinal direction and the type description are already available in the saved format, but the relative size calculation requires an additional step.

In order to classify the individual elements into the seven relative size categories, the quantiles of the total representative size information of all generated geometries must be calculated (section B.3.2). This means that, for example, a living room classified as XL is relatively large compared to the average living room size generated by the algorithm and controlled by the architectural function. It is also necessary to calculate relative energy classes per apartment layout based on energy consumption values. This is similarly done by considering the full range of simulated energy values and dividing them into the desired number of classes, five in the case of this experiment, by quantiles representing the minimum and maximum values of each energy class (section B.3.3).

This means that, after successful execution of the data mapping process, the dictionary objects of the individual rooms and windows have now been extended by a 'label' key, which carries a category from 0 to 91 as a value and describes the essential relative information of the individual element, such as: window - medium - south-east or bathroom - extra small.

4.4.4 Graph Retrieval

The basis for generating synthetic graph datasets is thus complete and functional; the only thing missing is the automatic translation from the geometric information model to the topological knowledge graph (figure 4.25). To achieve this as smoothly as possible, the capabilities of the Python library *TopologicPy* were used, as it offers the possibility to generate different topological graphs based on geometric objects and to automatically assign information to each node and edge.

The particularity of the generated graphs is that the individual nodes not only represent the individual spaces and their topological relationship, but also, starting from the individual spaces, edges are drawn to the window apertures associated with the spaces. This means that the resulting knowledge graph objects represent a topological description of the room constellation and window arrangement for the entire apartment. Furthermore, the topological graph generation method allows an automated and regularised information transfer between the dictionary objects of the geometry and the elements of the graph. Thus, the individual nodes receive the previously generated label keys with the corresponding categories of the individual rooms and windows (table B.5). Similarly, the individual graph objects receive general dictionaries, which in this case contain the *site energy consumption per surface* value in MJ/m^2 and the respective energy class as well as an individual reference number (table B.4).

The complete knowledge graph objects are then stored as binary objects using the Python *Deep Graph Library (DGL)*[28] to build the final dataset.

4.5 Outcome

4.5.1 Results

Through the described construction of the generation pipeline, it was possible to generate a large dataset of 40 000 synthetic housing graphs in a relatively short time. Key aspects such as the selection of the best geometry partitioning methods, the construction of the parametric framework, the architectural control function, and the information retrieval and annotation were satisfactorily handled and allowed a broad insight into the respective tools and algorithms. An essential requirement of the dataset was the morphological and architectural versatility of the individual BIM models, which was made possible by the application of different partitioning methods, namely Voronoi diagram, optimised recursive subdivision and the combination of both.

The final dataset consists of two complementary parts, which are stored separately. The first dataset represents the totality of the generated housing geometries including the generated apertures and the stored information about the individual elements (figure 4.26 and section B.3.1). The second part consists of the corresponding knowledge graphs of the individual apartment geometries, which contain essential information such as the energy efficiency and topology of the individual apartment layouts (figure 4.27 and tables B.4, B.5 and B.6).

Several methods were used to verify the quality of the generated dataset. Firstly, an arbitrary visualisation of the geometries and corresponding graphs was performed, with a focus on controlling for unwanted regularities. This revealed a high variance in the layout design, indicating a satisfactory result. However, due to the large amount of data involved, a statistical analysis of the datapoints and associated information was required.

Accordingly, the correlations of the three variables: opening area in m^2, total surface in m^2 and energy consumption in MJ/m^2 were first plotted in order to detect any unexpected patterns or relationships (figure 4.31). This and the analysis of the correlations between the energy consumption in MJ/m^2, the id number as $\in N$ and the total surface in m^2 (figure 4.32) showed a satisfactory distribution of the individual information variables. Some logical relationships could also be verified for the legitimacy of the dataset, such as the linear relationship between total surface and total window opening area, or the

[28]M. Wang et al. 2019, “Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks”.

increased energy consumption for larger apartments.

The distribution of the values for total surface (figure 4.28), energy consumption (figure 4.29) and total window opening area (figure 4.30) was also examined to confirm the variance of the dataset. A satisfactory distribution of the total surface was found within the defined maximum and minimum values, which was likewise the case for the distribution of the total window opening area. The distribution of energy consumption values showed a concentration around the value of 660 MJ/m^2 with a relatively strong increase on the left side of the peak. Whereas the decrease on the right side of the graph shows a much less steep gradient and extends up to the values at 1100 MJ/m^2. Thus indicating that the energy consumption values in this dataset are non-symmetrically distributed.

4.5.2 Conclusion

In this thesis, the primary reason for the creation of the graph dataset is for its use in machine learning. However, the generation pipeline provides a basis for automatic floor plan generation that can also be used in the context of optimised floor plan generation processes. In this case, however, a modification of the described processes is necessary, as the ambition is then to generate an optimal design solution and not a search for the greatest possible variance in the design.

The whole generation workflow implemented in Python shows a generally satisfactory harmony of the different stages. However, some could benefit from more customised logic and functionality. For example, for maintenance reasons, it would be advisable to replace the Voronoi generation with *SciPy's* spatial Voronoi instead of the Python library *PyVoro*. It would also be advantageous to speed up certain aggregation methods by parallelising processes, thus allowing their integration into the pipeline. In fact, the variance of the dataset would benefit from such integration, as in the current implementation there is no way to generate floor plans with a central patio or external balconies.

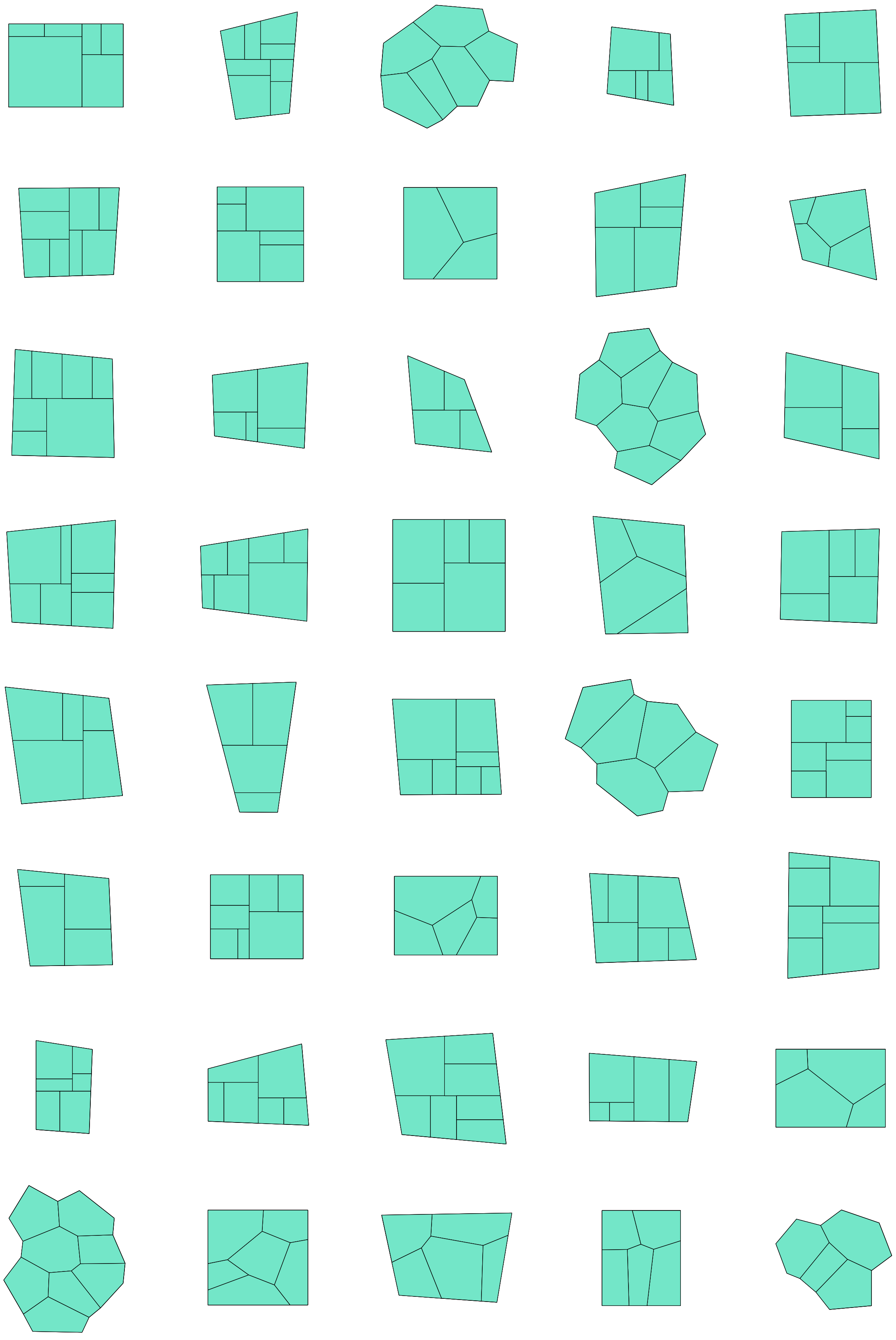


Figure 4.26: Geometry Dataset

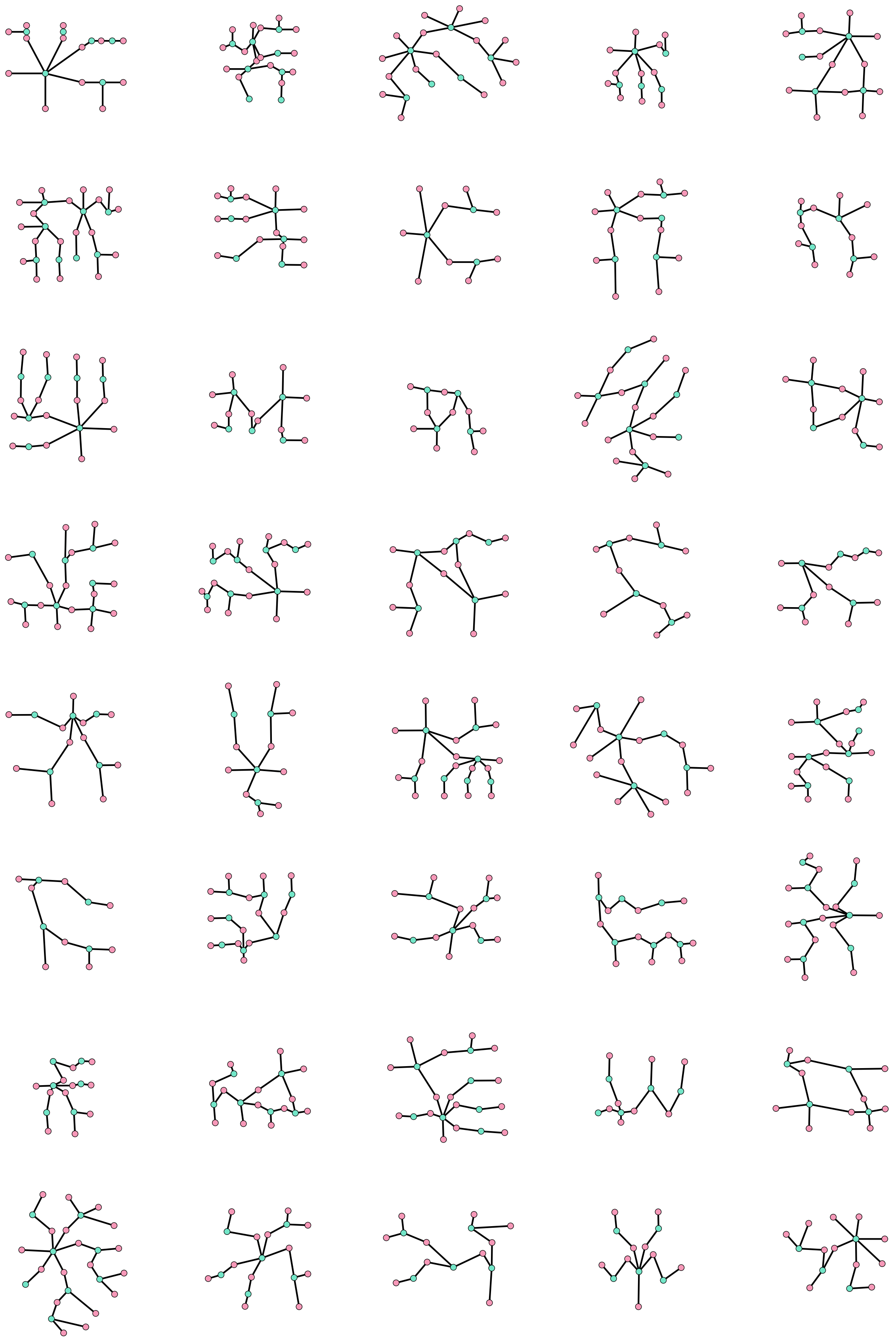


Figure 4.27: Corresponding Graph Dataset

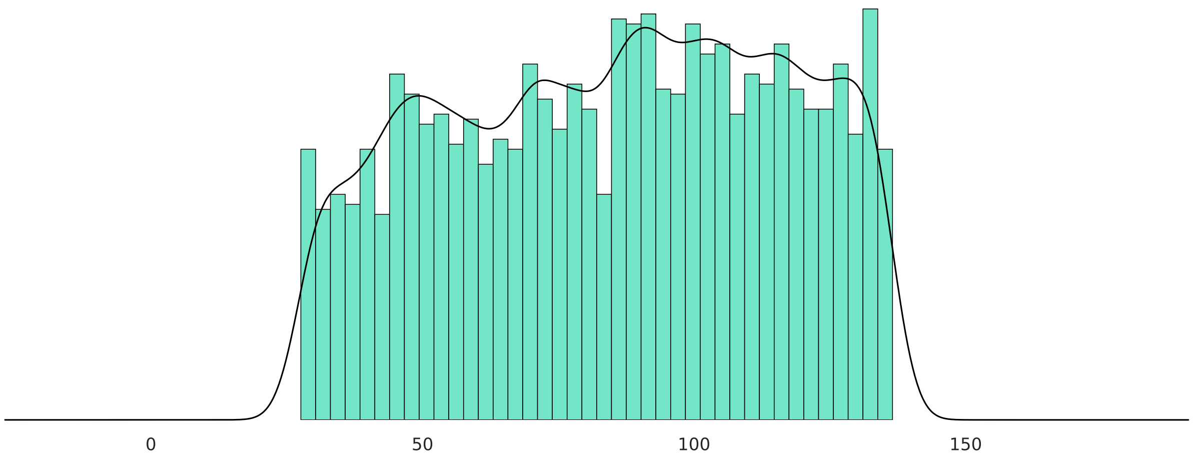


Figure 4.28: Distribution of Surface Value

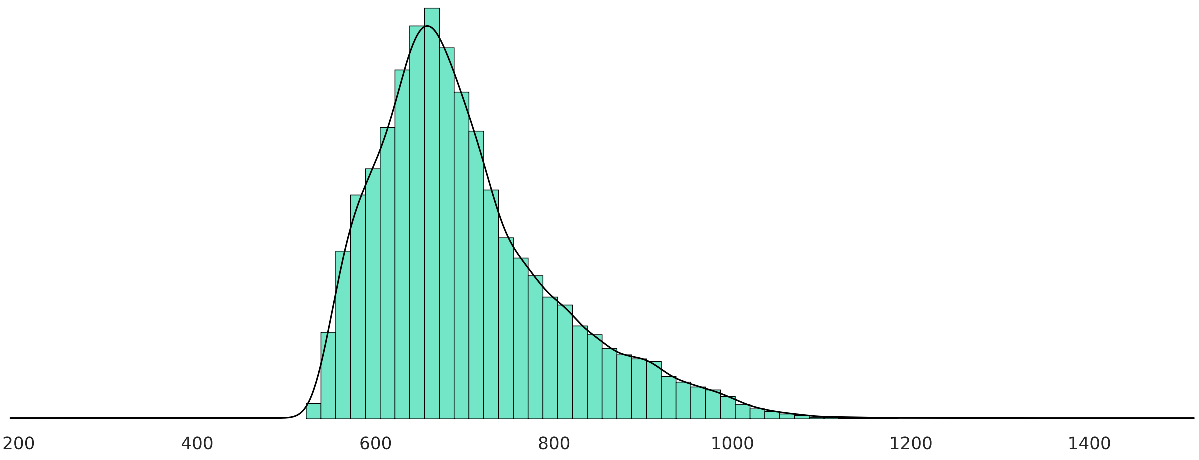


Figure 4.29: Distribution of Energy Consumption Value

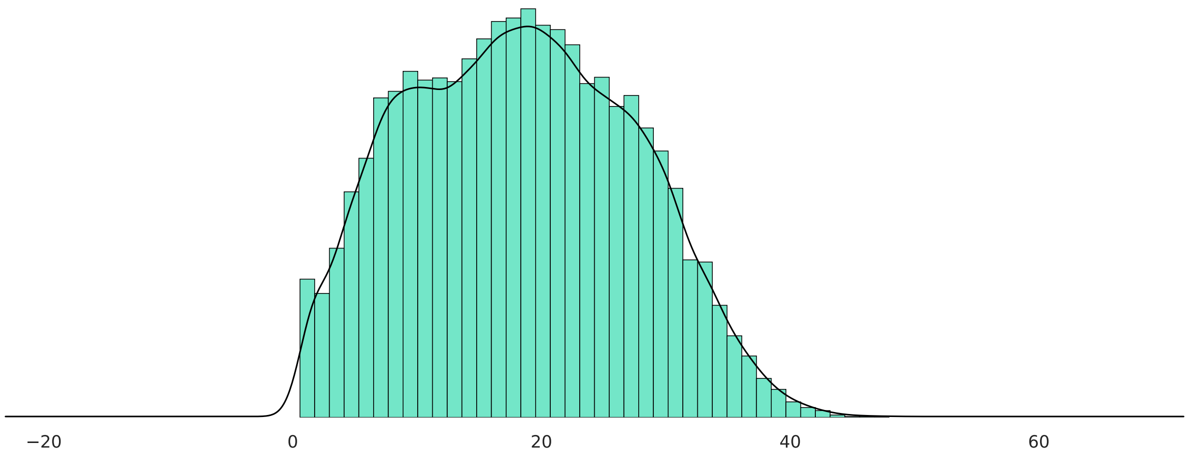


Figure 4.30: Distribution of Opening Area Value

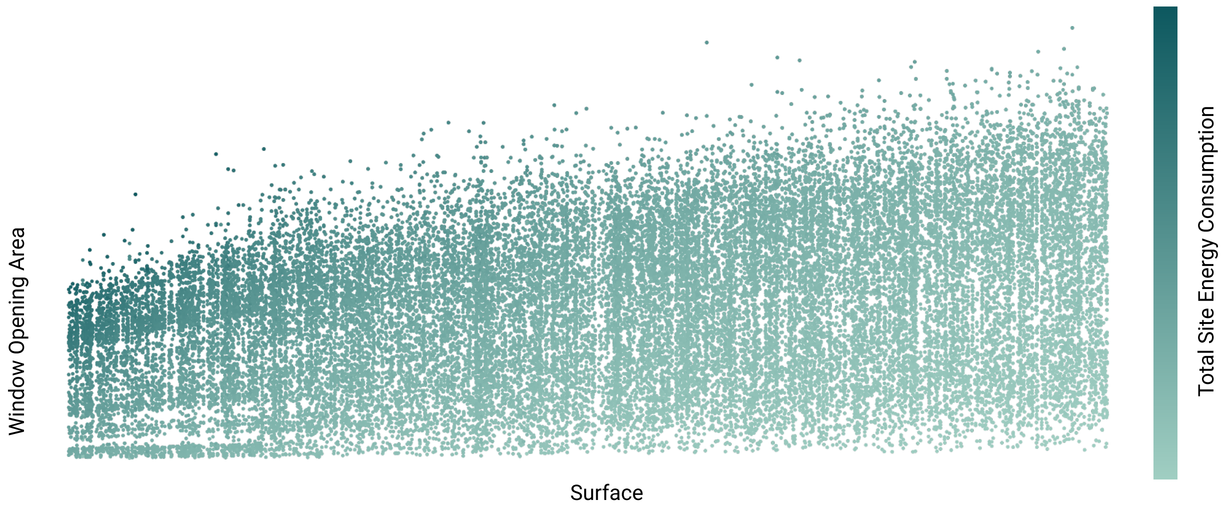


Figure 4.31: Relation Between Opening Area, Surface and Energy Consumption

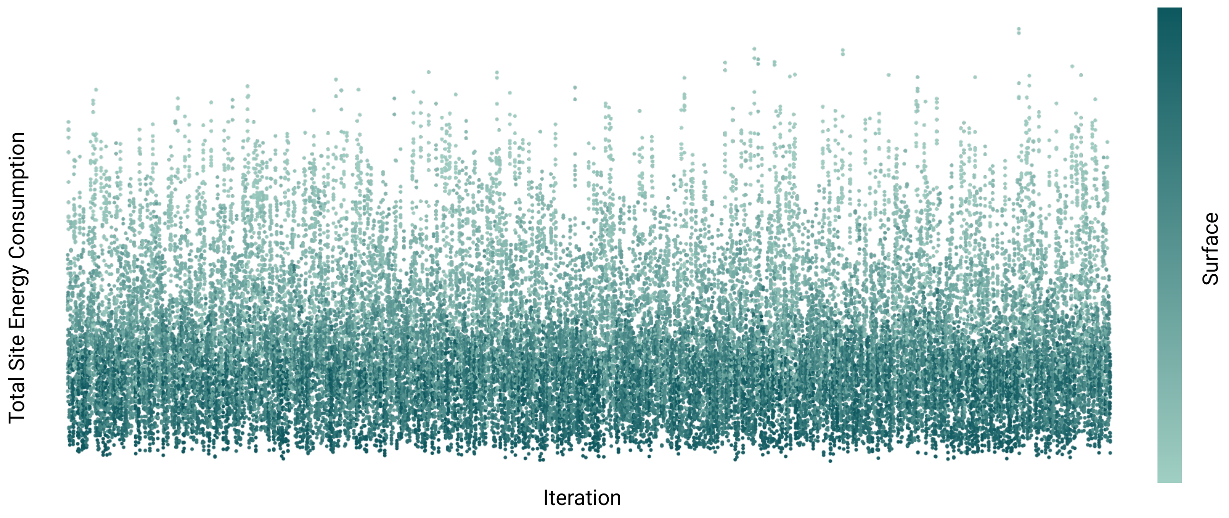


Figure 4.32: Relation Between Energy Consumption, Layout ID and Surface

Chapter 5

Graph Machine Learning

The second part of the methodology deals with the construction, training and evaluation of different graph machine learning models. The focus is on two special variants of graph deep learning. First, a model is developed that is trained by graph classification methods and thus represents a classification model. More specifically, this model is trained on the labelled graph objects and their energy class to be able to classify new, unseen and annotated graph structures into one of the five energy classes. The second model is also based on graph-wide prediction, but uses continuous energy consumption values instead of classes, and is therefore a regression model. In other words, for any architecture-based graph structure, this regression model allows the site energy consumption per surface in MJ/m^2 of the underlying architectural objects to be predicted as accurately as possible.

The two models created belong to the family of Deep Graph Convolutional Neural Network*s* (DGCNN*s*)[1] and, as the name suggests, are based on a *convolutional neural network structure*. Due to their special architecture, they allow the use of graphical datasets as a basis for the supervised learning process instead of conventional pictorial or tabular training datasets. Since DGCNN*s* are composed of several layers, each of which performs convolutional operations followed by *activation functions* and *pooling operations*, information can be reduced and abstracted so that classes or values corresponding to the entire graph can be predicted[2]. Another key feature of DGCNN*s* is their ability to handle graphs of variable size and structure. This means that these networks are able to process graphs with different numbers of nodes and edges without changing the network structure or training method, which has proven to be an essential feature in their application.

The entire training, testing and *validation* process, as well as the evaluation of the results, was implemented in Python, as was the geometric generation pipeline, and is essentially based on several different Python libraries such as *TopologicPy*, DGL, *PyTorch*[3], *Scikit-learn*[4], *NumPy* and *Pandas*[5]. The DGL library is by no means the only Python library that allows the creation and training of DGCNN*s*; However, it was preferred over other libraries, such as *PyTorch Geometric*, because DGL is an excellently documented library that can be easily combined with the *TopologicPy* toolkit. In addition, DGL offers interesting functions such as the mean node calculation or a wide range of *sampling methods*.

[1]Velikovi 2023, "Everything is Connected: Graph Neural Networks".

[2]M. Wang et al. 2019, "Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks".

[3]Paszke et al. 2019, "PyTorch: An Imperative Style, High-Performance Deep Learning Library".

[4]Pedregosa et al. 2011, "Scikit-learn: Machine Learning in Python".

[5]McKinney 2010, "Data Structures for Statistical Computing in Python".

5.1 Structure

In order to successfully train the respective DGCNN models, several steps must be taken, beginning with the data preparation. With the help of the DGL library, a *DGL.Dataset* object is created, which allows the individual stored and annotated graph objects to be read in, converted into adjacency matrices and the respective training information to be stored at graph level. The thus created graph dataset object is perfectly suitable for continuing the training preparation.

The next step is to divide the dataset for the *holdout* training method into a training, validation and test dataset, with the test and validation datasets each representing ten percent of the total. The resulting validation dataset is stored and used only after successful training. Because of the large amount of data involved, it is also necessary to create a so-called *data loader*, which allows the graphs to be loaded in a bundled form per learning step. This is based on the creation of so-called *graph batches*. A graph batch consists of several graph objects that are combined into a larger bundled graph without changing their structure or information. In this sense, a graph batch is a supergraph object to the individual subgraphs from the dataset. The data loader function also needs information about how to read the graphs from the dataset. In the present project, a *subset random sampler* was chosen to allow a random composition of the graph batches.

At this point it is advisable to verify the balance of the created dataset, which in concrete application means comparing the number of energy classes occurring in the dataset. If these percentages are not sufficiently balanced, methods such as *over-* or *undersampling* should be used to rebalance the dataset.

The data preparation and reading is now complete, and all that remains to be done is to define the model and its individual structure, the hyperparameters that influence training, and the choice of optimisation methods. Then the training cycle can begin, where *epoch* by epoch the weights of each layer of the model are optimised by *backpropagation* to improve the prediction accuracy on the training data. Finally, the trained model is tested on the test dataset to evaluate its prediction accuracy.

This framework is the same for both classification and regression models. The two methods differ only in their individual structure, input data and the way *loss* is calculated during the training process.

The graph classification model (figure 5.1) is based on the energy classes contained in the dataset at the graph level, ranging from zero to four. Its layers consist of the *input layer* with a dimension corresponding to the number of node labels, a variable number of graph convolutional layers with different dimensions and activation functions, a pooling layer used to reduce the size of the graph and at the same time the information loss, and finally the *output layer* corresponding to the dimension of the number of classes to be predicted and providing the final prediction about the class membership of the graph. The loss calculation for the classification model can be computed by either *cross-entropy* or *negative log-likelihood*.

The structure of the regression model (figure 5.2) is similar to that of the classifier; however, the final layer is a *linear layer*, which is one-dimensional because the prediction of the regression model is, by definition, almost always a single value. Another difference is the loss calculation, which is performed in the regression model by calculating the *Mean Squared Error*.

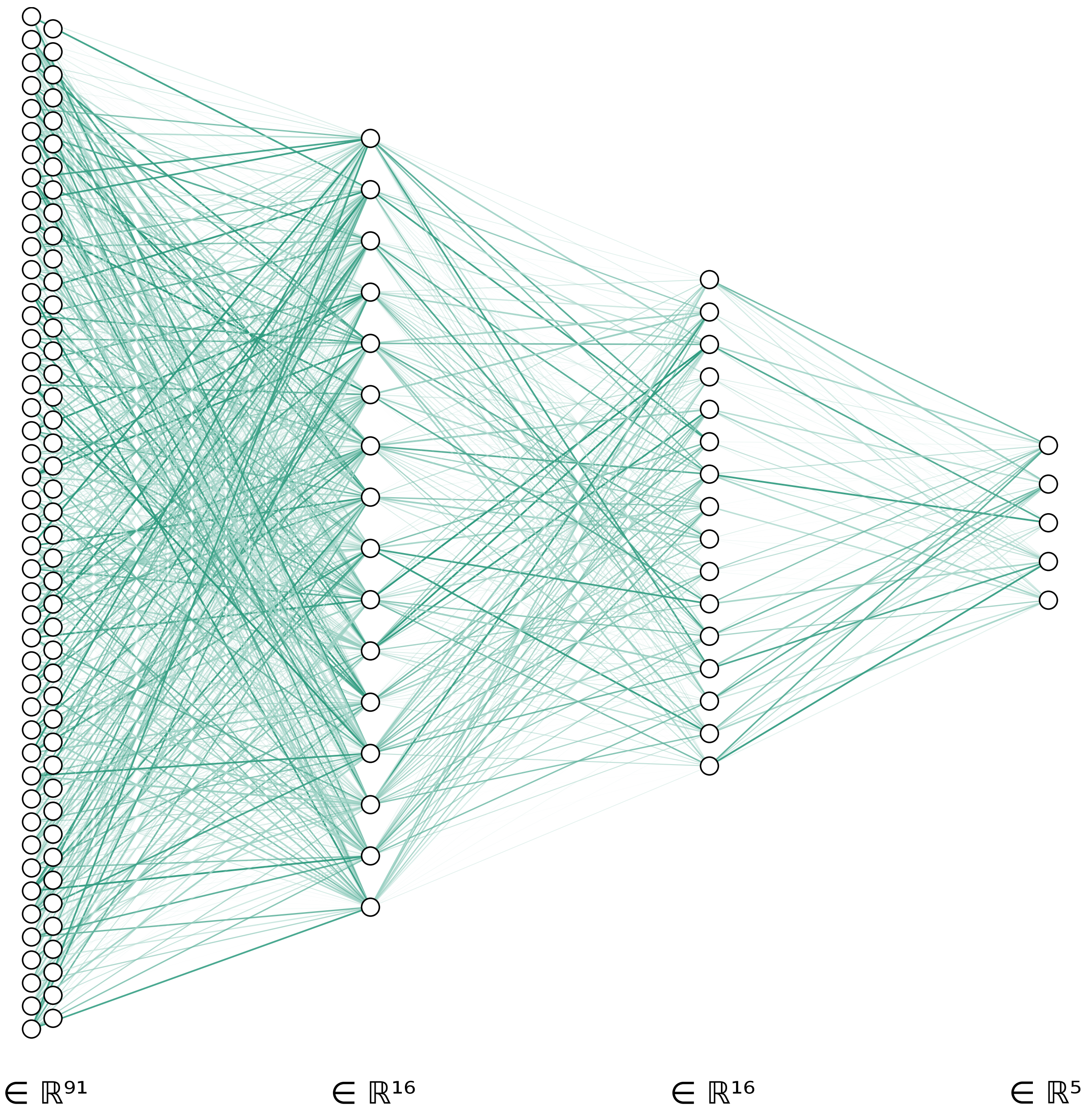


Figure 5.1: Classification Model Structure

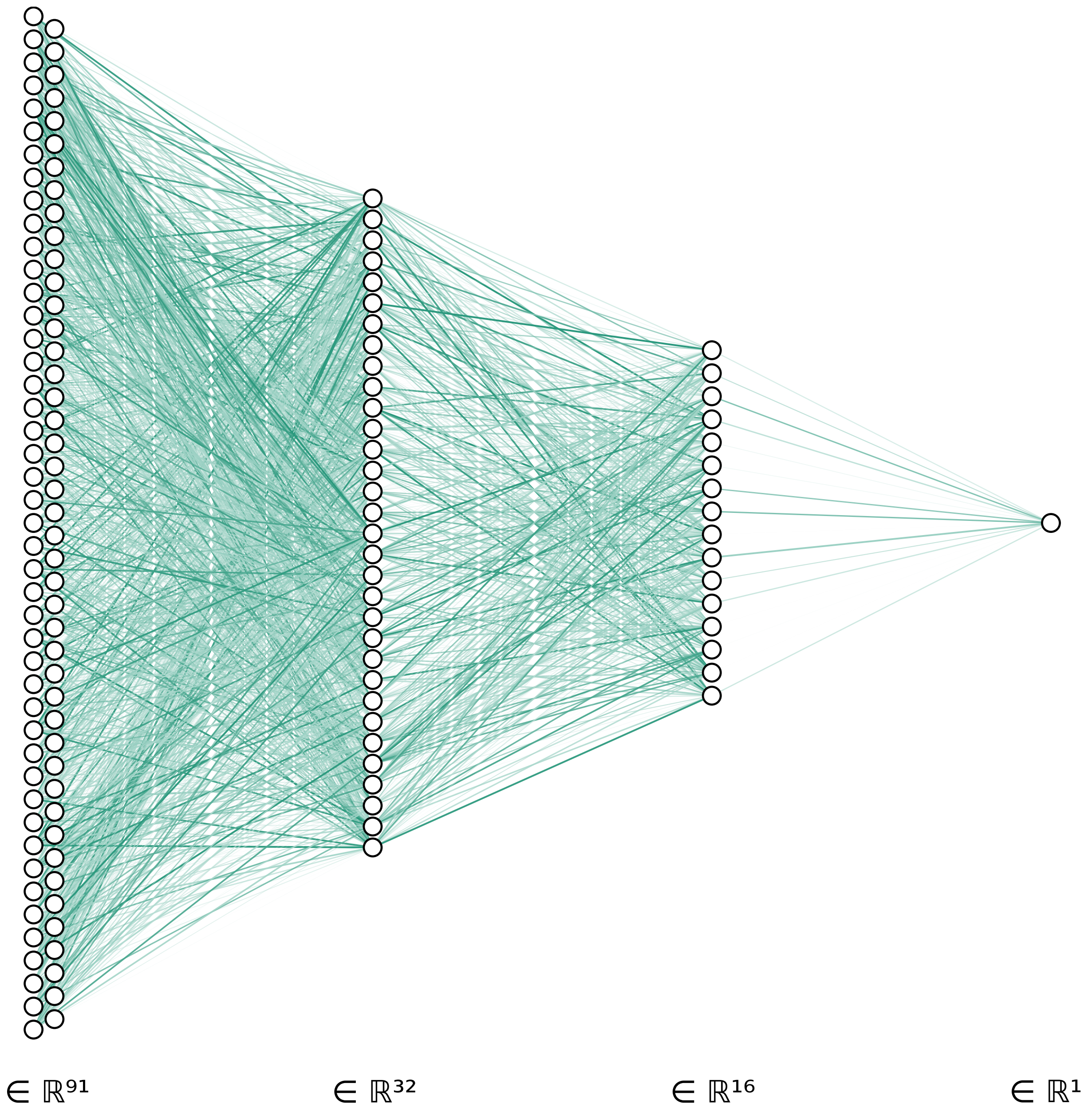


Figure 5.2: Regression Model Structure

Epochs	Batch Size	Learning Rate	Pooling	Hidden Layer	Layer Type	Optimiser	Loss Function
300	32	0.01	Average	32	TagConv	Adam	MSE

Table 5.1: Regressor Hyperparameter

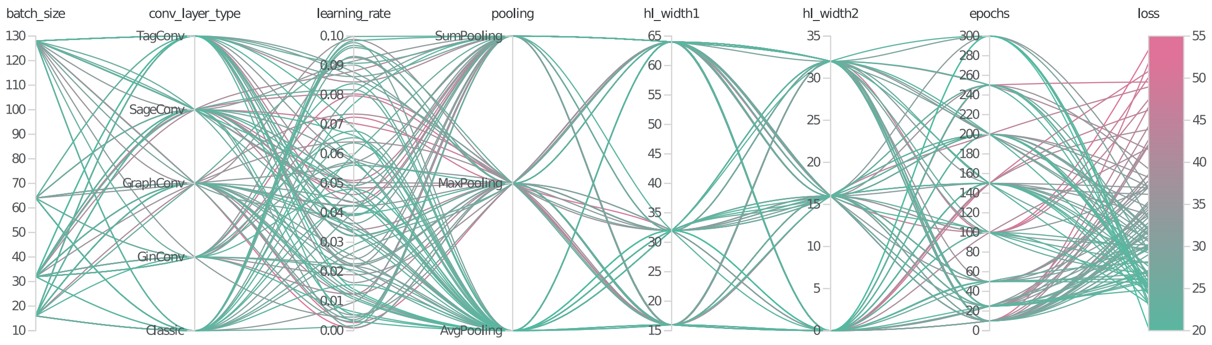


Figure 5.3: Regressor Hyperparameter Tuning

5.2 Hyperparameter

Hyperparameters are parameters that control the behaviour of the model, but are usually not chosen by the model itself. Instead, they have to be set manually by the developer. In this work, the main hyperparameters are the *learning rate*, the number and dimension of *hidden layers*, the number of epochs, the *batch size*, the pooling strategy, the loss function, the *optimisation method* and the convolution layer type. Since there are many different and interdependent parameters, certain methods such as *random search* or Bayesian optimisation can be used to find the optimal hyperparameters. This process is called *hyperparameter tuning* in computer science.

The basic principle is to perform the training process with the parameters determined by the automatic tuning algorithms and then test the performance of the model on the validation dataset. This process is repeated until the most optimal combination of each hyperparameter is found. The results of the random search tuning as well as the Bayesian optimisation method (figure 5.3 and 5.4) show that for the classification model and the regression model, the hyperparameters shown in the tables 5.1 and 5.2 lead to an optimisation of the prediction and classification accuracy.

5.3 Evaluation

There are several evaluation metrics that can be used to evaluate the trained DGCNN models. The most common are *accuracy* (table 5.3), precision, *recall*, *f1-score* and Mean Squared Error or *Root Mean Squared Error* (table 5.4), although only the last two can be used as a strategy for evaluating regressors. These metrics measure different aspects of model performance, such as the overall precision of the model, or even the ability to distinguish positive examples from negative ones in the case of classifications.

To ensure that the evaluation of the DGCNN model is robust and reliable, a *10-fold*

Epochs	Batch Size	Learning Rate	Pooling	Hidden Layer	Layer Type	Optimiser	Loss Function
50	64	0.01	Average	16 | 16	GraphConv	Adam	NLL

Table 5.2: Classifier Hyperparameter

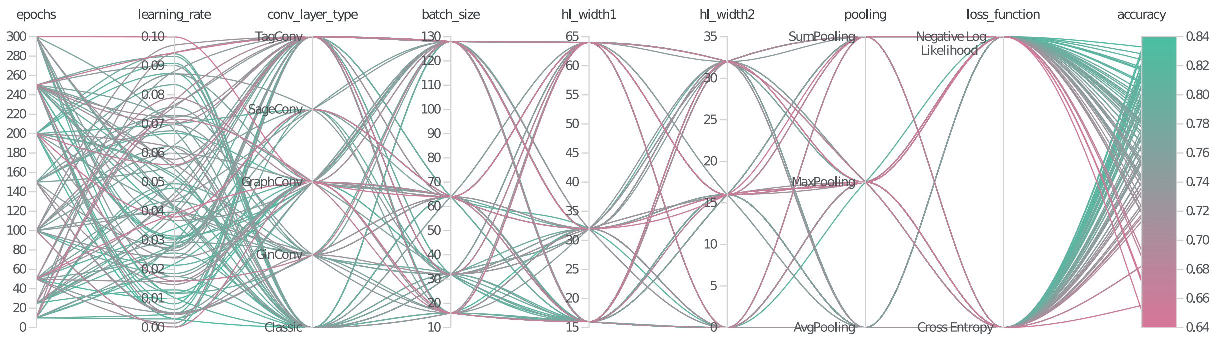


Figure 5.4: Classifier Hyperparameter Tuning

cross-validation is preferable to simple holdout training. In this method, the dataset is divided into 10 subsets and the model is trained and tested accordingly 10 times, using each subset once as a test dataset. The results are then averaged to obtain a robust evaluation score of the two models.

5.3.1 Classification Metrics

Graph classification models are typically evaluated based on their accuracy in predicting graph classes. To do this, graphs are divided into training and test datasets. The training dataset is used to train the model, while the test dataset is used to evaluate the performance of the model. The accuracy of the model is then measured by the number of correct predictions on the test dataset. A commonly used metric for evaluating graph classification models is the accuracy metric, which is calculated as the ratio of correct predictions to the total number of predictions.

In addition to accuracy, other metrics can be used to evaluate the performance of graph classification models, such as precision. This is the number of times that the model correctly predicts that a datapoint belongs to a particular class, as measured by all the predictions that the model has assigned to that class. It is calculated as the ratio of *True Positive* predictions (TP) to the sum of TP and *False Positive* predictions (FP).

In addition, the recall score can be used for evaluation, which indicates how often the model correctly predicts that a datapoint belongs to a particular class, measured against all actual datapoints of that class. Recall is calculated as the ratio of true TP predictions to the sum of TP and *False Negative* predictions (FN).

Finally, it is also common in research to calculate the f1-score of a classification model, which is a combination of precision and recall. It indicates how well the model is both precise and comprehensive. This score represents the *harmonic mean* of precision and recall.

It is also important to note that the performance of graph classification models is highly dependent on the quality of the data used. It is therefore necessary to carefully verify that the training and test datasets are sufficient and representative of the application scenario (section 4.5.1), as well as having the best possible balanced class distribution in the dataset to avoid possible *overfitting*.

5.3.2 Regression Metrics

Classification accuracy metrics such as precision, *accuracy*, recall and f1-score are not appropriate for regression models because they are designed to evaluate the performance of discrete class prediction models. In a regression problem, however, the goal is to predict a numerical target variable given a set of input variables. Because the target variable is continuous rather than discrete, accuracy metrics such as conventional classification accuracy cannot be used because they are unable to measure the accuracy of predicting continuous variables.

Instead, regression models are evaluated using metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE) and *Mean Absolute Error (MAE)* to quantify the predictive performance with respect to the actual values of the target variable. The MSE indicates the degree of Mean Squared Error between the model's predictions and the actual values of the target variable. It is calculated as the mean of the squared difference between the predictions and the actual values. The RMSE is the root of the MSE and thus indicates how large the average absolute error is between the model's predictions and the actual values of the target variable. The MAE on the other hand indicates the size of the average absolute error between the predictions of the model and the actual values of the target variable.

5.3.3 Comparison

Finally, in order to adequately evaluate the performance of the two trained models, they are compared with other models that have similar functionalities and their respective prediction accuracies. Thus, the DGCNN Classifier is compared with a number of different classification models such as the *Random Forest Classifier*, the *Gradient Boost Classifier*, the *Decision tree Classifier*, *K-Nearest Neighbours Classifier*, *Support Vector Classifier*, *Logistic Regression* and *Multi-layer Perceptron Classifier* and their respective accuracy (figure 5.5).

The DGCNN regression model was compared with models such as *Random Forest Regressor*, *Gradient Boosting Regressor*, Decision tree Regressor, *K-Nearest Neighbours Regressor*, *Support Vector Regressor*, *Linear Regression*, *Ridge Regression*, *Lasso Regression*, *Multi-layer Perceptron Regressor* and *Elastic Net* (figure 5.8). However, it is important to understand that these are different machine learning approaches and therefore the underlying data is required in different formats. This means, for example, that the aforementioned comparative models were trained entirely on tabular data (section B.3.4), as opposed to the graphical data used for the DGCNN models. This has to be considered as an essential part of the evaluation.

5.4 Outcome

5.4.1 Machine Learning for Energy Prediction

Both classification and regression models have been trained on the synthetic graph dataset to predict the energy performance score of an apartment. The classification models predict the energy efficiency class of an apartment, while the regression models predict the actual energy consumption in MJ/m^2.

At this point, it is crucial to re-emphasise the advantages of the trained models. Contrary to the common assumption that machine learning and artificial intelligence in the field of

AEC are mostly applied to the automatic generation of optimised architectural plans, in this experimentation the focus is purely on enabling optimised design feedback loops. In other words, the primary application of the models takes place during the conceptual phase of the design process, hand in hand with the architect or designer. The creative freedom of the architect's work is therefore in no way affected or constrained by the application of the machine learning models, but rather provides a continuous indication of the actual energy consumption of the current project. This allows the designer to easily test certain decisions, such as the placement, size or orientation of a window, in terms of changes in energy consumption, without having to continuously run costly energy simulations. More on the concrete architectural application of the results of this research can be found in the sections 6.5 and A.2.

5.4.2 Classification

This section provides a comprehensive performance comparison between several classifiers. The models have been trained on information derived from the synthetic graph dataset, allowing their respective capabilities and limitations to be assessed and evaluated in the context of the task at hand. The use of the 10-fold cross-validation technique ensures the generalisation of the models and the reliability of their performance metrics. In order to compare the performance of the DGCNN classifiers, several different models have been used, such as Logistic Regression, MLP, Support Vector, Random Forest, Gradient Boosting, Decision tree and K-Neighbors. They were evaluated based on their mean accuracy, which provides an overview of how well the classifiers performed on average. The results are presented graphically in figure 5.5 and numerically in table 5.3. In addition, a detailed analysis of the performance of the DGCNN Classifier is presented, including a mean confusion matrix which helps to understand the classification accuracy for each class individually.

5.4.2.1 Results

It can be seen that the DGCNN Classifier has the highest mean accuracy (0.83), followed by the logistic regression (0.82) and the MLP Classifier (0.79). On the other hand, the K-Neighbors Classifier has the lowest mean accuracy (0.52). The K-Neighbors Classifier is a model that relies on local patterns in the *feature space* to make predictions. This approach works well when the input data is densely packed and can be easily separated into different classes based on these patterns. However, if the data has a more complex structure, such as graph-derived data, this approach may not be effective. The DGCNN Classifier is designed to work with such graphical data and can learn the underlying features of the graph structure to make accurate classifications.

The Random Forest Classifier has a mean accuracy of 0.65, which is relatively good compared to some of the other classification models. The Random Forest Classifier is an ensemble learning method that relies on a collection of decision trees to make predictions. It is known for its robustness to overfitting and its ability to handle high-dimensional data. However, the performance of the model can be sensitive to the choice of hyperparameters, such as the number of trees and the depth of such trees. The Gradient Boosting Classifier has a mean accuracy of 0.61, which is slightly lower than the Random Forest Classifier. This model is also an ensemble learning method that uses a collection of *weak learners* to make predictions. It is known for its ability to handle high-dimensional data and its ability to predict the shape of complex non-linear functions that describe the relationship between the features and the target variable. However, like the Random Forest classifier,

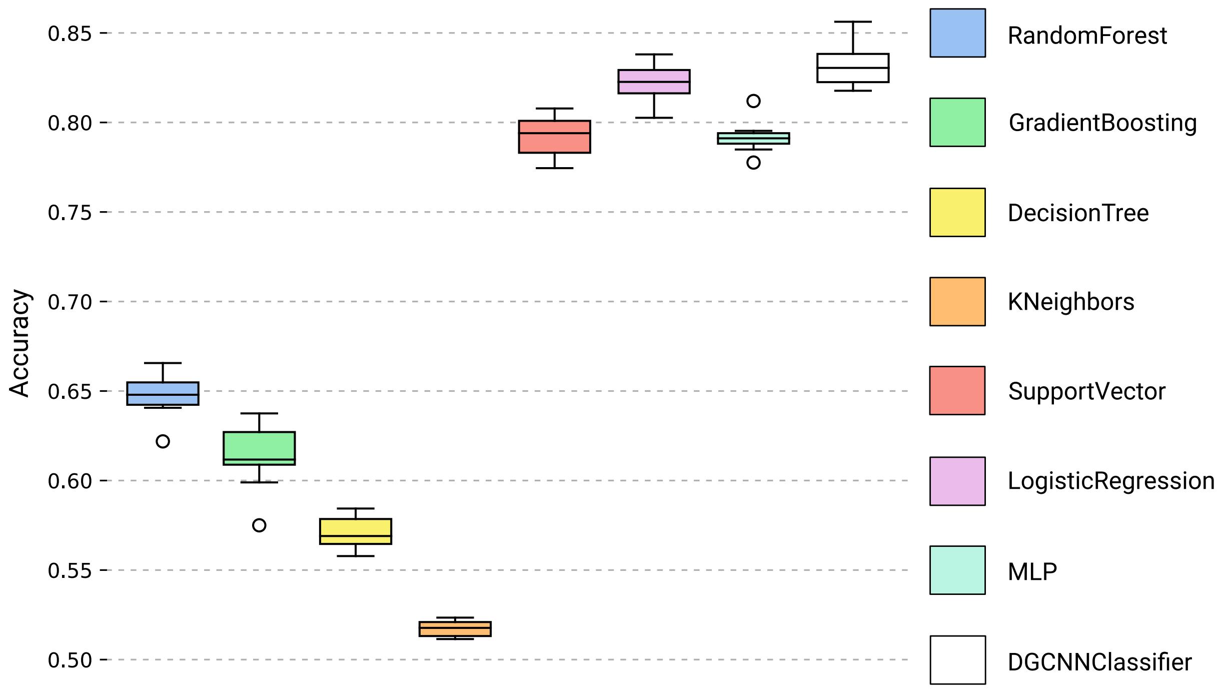


Figure 5.5: 10-Fold Cross Validation Accuracy for Different Classifiers

Classifier	**Mean Accuracy**	**Standard Deviation**
DGCNN Classifier	0.83	0.01
Logistic Regression Classifier	0.82	0.01
MLP Classifier	0.79	0.01
Support Vector Classifier	0.79	0.01
Random Forest Classifier	0.65	0.01
Gradient Boosting Classifier	0.61	0.02
Decision Tree Classifier	0.57	0.01
K-Neighbors Classifier	0.52	0.00

Table 5.3: Mean Accuracy and Standard Deviation of Different Classifiers

the performance of the model can be sensitive to the choice of hyperparameters.

Not surprisingly, the Decision tree Classifier has a mean accuracy of 0.57, which is lower than the Random Forest Classifier and the Gradient Boosting Classifier. This model is a simple tree-based learning method that makes predictions based on a set of if-then rules. It is known for its simplicity and interpretability, but it can be prone to overfitting and may not perform well on such complex datasets.

The Support Vector Classifier has a mean accuracy of 0.79, which is relatively high in comparison. This model is a *discriminative learning* method that divides the data into different classes by slicing planes in n-dimensional space according to the data features. Its main advantage is its ability to learn on tensors of higher dimensions and its robustness to noisy data. The MLP Classifier has a mean accuracy of 0.79, which is similar to the Support Vector Classifier. This model is a neural network based learning method that uses multiple layers of interconnected nodes to learn the features of the input data. It is also capable of learning non-linear functions that map the features to the target variable, but the selection of hyperparameters must be well chosen.

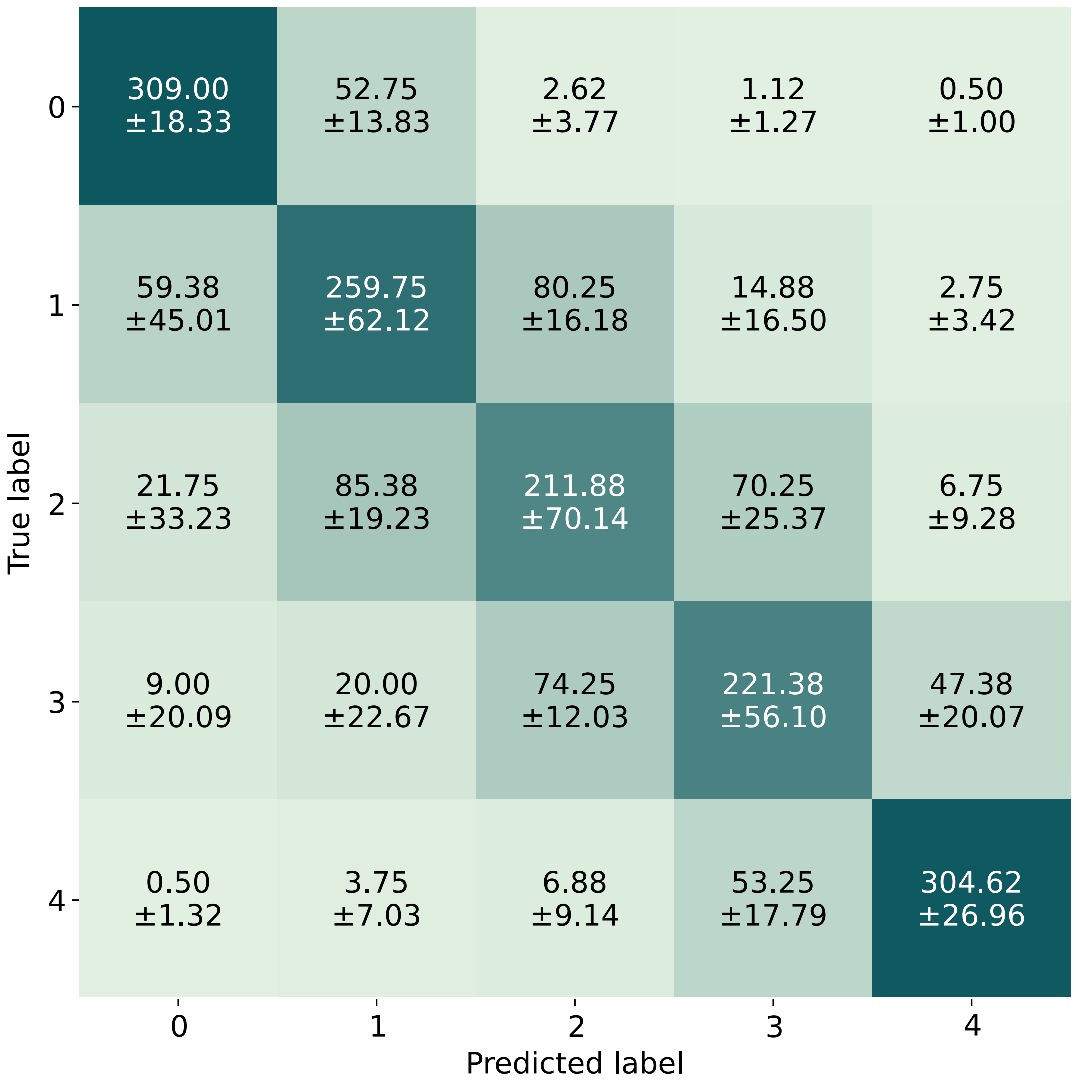
Logistic regression has a mean accuracy of 0.82, which is surprisingly the second highest of all the classifiers. This model is a probabilistic learning method that models the probability of the target variable given the input data. It is known for its simplicity and interpretability and can perform well on a wide range of datasets. It is possible that the linear nature of the logistic regression model is beneficial for this particular task, allowing the classifier to be less prone to overfitting.

Figure 5.6 shows the *mean confusion matrix* for the DGCNN Classifier. The confusion matrix represents the distribution of predicted and true labels for each class. The x-axis and y-axis of the matrix represent the predicted and true labels respectively. The colour intensity of each cell in the matrix represents the mean of the confusion matrix for the corresponding true and predicted labels. The values within each cell indicate the mean of the 10-fold cross-validation confusion matrices with the corresponding *standard deviation* value. Based on the mean and standard deviation values, it appears that some classes are easier to predict than others using the DGCNN Classifier. For example, the diagonal values for classes 0 and 4 are relatively high, indicating that these classes are easier to predict accurately. In contrast, the off-diagonal values for classes 1, 2 and 3 are likewise relatively high, indicating that these classes are harder to predict accurately. The standard deviation values also show some variability in the confusion matrix, with the highest levels of variance observed in the off-diagonal values for classes 1 and 2.

5.4.2.2 Conclusion

The comprehensive performance analysis presented in this section illustrates the comparative performance of several models predicting different energy consumption classes for unseen apartment layouts through 10-fold cross-validation. The results clearly indicate that the DGCNN classifier outperforms the other models, with the highest mean accuracy of 0.83. This superior performance can be attributed to its ability to handle the complex structure of graph-derived data, effectively learning the underlying features of the node relationships for accurate classifications.

The mean confusion matrix for the DGCNN Classifier further reveals that some classes are easier to predict than others. This highlights the importance of considering class distribution and potential class imbalances when evaluating model performance.



True label \ Predicted label	0	1	2	3	4
0	309.00 ±18.33	52.75 ±13.83	2.62 ±3.77	1.12 ±1.27	0.50 ±1.00
1	59.38 ±45.01	259.75 ±62.12	80.25 ±16.18	14.88 ±16.50	2.75 ±3.42
2	21.75 ±33.23	85.38 ±19.23	211.88 ±70.14	70.25 ±25.37	6.75 ±9.28
3	9.00 ±20.09	20.00 ±22.67	74.25 ±12.03	221.38 ±56.10	47.38 ±20.07
4	0.50 ±1.32	3.75 ±7.03	6.88 ±9.14	53.25 ±17.79	304.62 ±26.96

Figure 5.6: Mean Confusion Matrix for DGCNN Classifier

Overall, these results emphasise the crucial role of choosing the right classifier architecture depending on the nature of the dataset and the problem at hand. For the task described, which involves complex graphical data structures, more sophisticated models such as the DGCNN Classifier are expected to offer superior performance. However, simplicity and interpretability should not be overlooked, as demonstrated by the relatively high performance of the logistic regression model. Future work could focus on optimising the hyperparameters of these models, further improving their performance on the dataset, and exploring other sophisticated models designed to handle complex graphical data structures.

5.4.3 Regression

The aim of this section is to compare the performance of a set of regression models trained on tabular data derived from the information contained in the graph dataset. The dataset consists of energy consumption values in MJ/m^2, which serve as labels for the regression task. Using the Root Mean Squared Error (RMSE) as an evaluation metric, it was possible to measure the respective performance of the models. The results are presented graphically in figure 5.8 and numerically in table 5.4. These results show the best performing regressor, with details of the respective standard deviation across multiple model outputs.

5.4.3.1 Results

A scatter plot of the actual and predicted values of the DGCNN regression model is shown in figure 5.7, where the x-axis describes the actual energy consumption values of the apartments in the test dataset in MJ/m^2 and the y-axis shows the corresponding predictions made by the trained DGCNN model. In this plot, the scatter points should be as close as possible to the diagonal, as their orthogonal distance to this line represents the mean error of each prediction. This plot shows that higher energy consumption values are associated with a slightly increased RMSE, which translates into an elevated inaccuracy for these values. On the other hand it can be seen that from about 900 MJ/m^2 the number of datapoints representing these values becomes sparser, which was already visible in figure 4.29.

When analysing the results, the MLP Regressor stands out as the best performing model with a RMSE of 18.82. This demonstrates that the MLP Regressor was the most accurate in predicting energy consumption values. The MLP Regressor, a type of neural network, exploits its ability to learn complex, non-linear relationships, which is likely to have contributed to its high performance in this task.

The DGCNN Regressor closely followed the MLP Regressor, with a slightly higher RMSE of 19.33 MJ/m^2. Given that DGCNN is inherently capable of handling complex structured data, such as graph data, it is not surprising that it performed well in this task. The high performance of DGCNN and MLP suggests that the underlying patterns present in the dataset may be complex, requiring sophisticated models to accurately predict the energy consumption values.

Linear Regression and the Ridge Regressor follow with a significant lead of about 15 units, yielding an RMSE of 34.52. As these two regressors are linear models, the difference in performance compared to MLP and DGCNN suggests that the relationship between the features and the energy consumption value is likely to be non-linear.

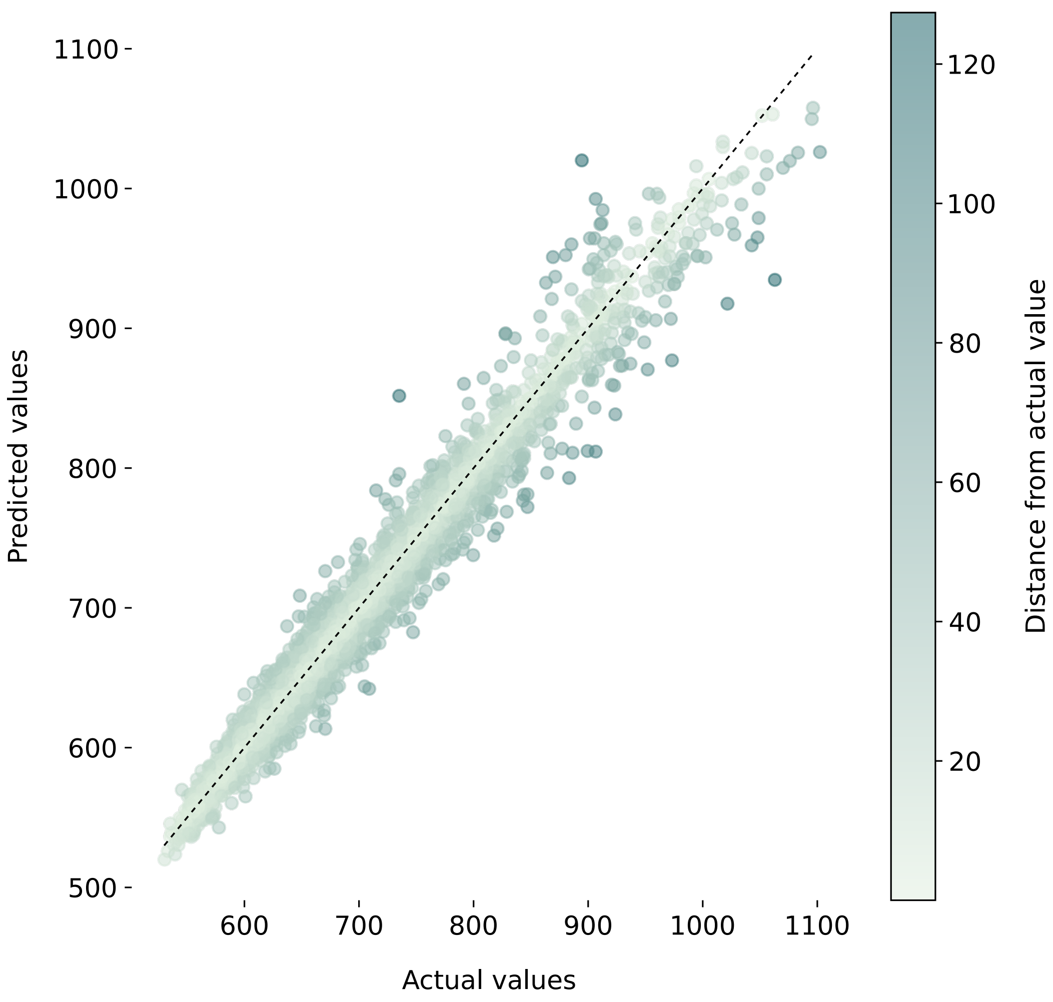


Figure 5.7: Actual and Predicted Values of the Regression Model

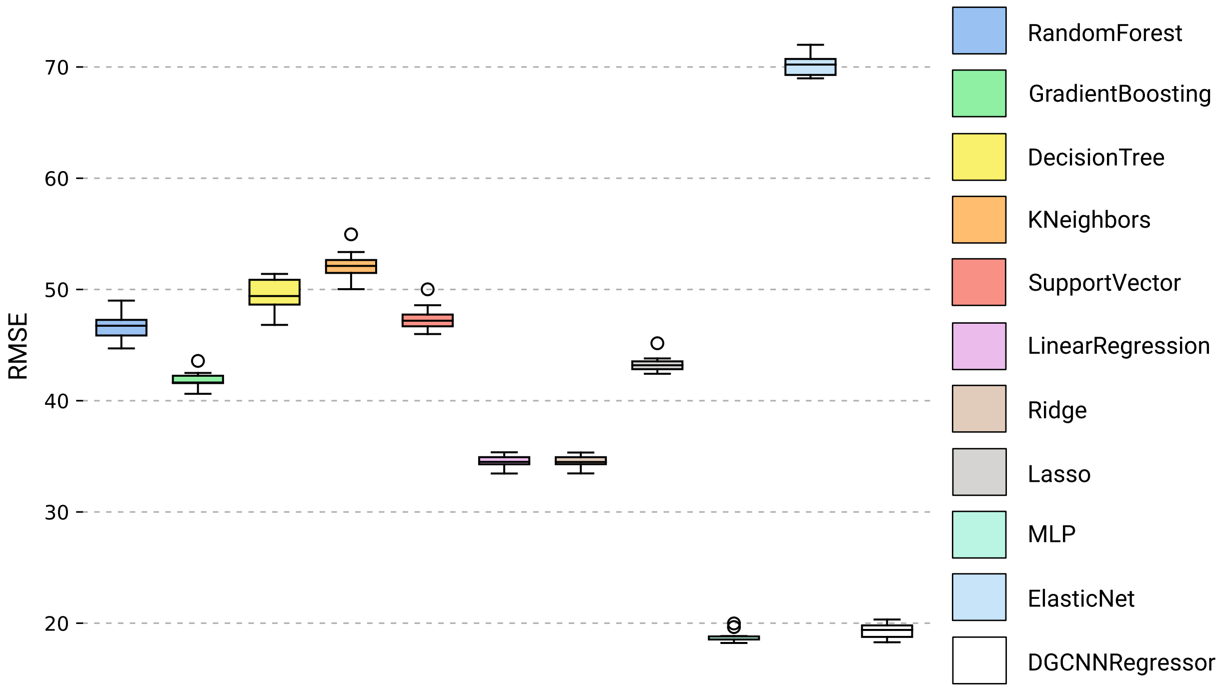


Figure 5.8: 10-Fold Cross Validation RMSE for Different Regressors

Regressor	**Mean RMSE**	**Standard Deviation**
MLP Regressor	18.82	0.53
DGCNN Regressor	19.33	0.63
Linear Regression	34.52	0.55
Ridge Regressor	34.52	0.55
Gradient Boosting Regressor	41.85	0.76
Lasso Regressor	43.29	0.76
Random Forest Regressor	46.73	1.17
Support Vector Regressor	47.41	1.13
Decision Tree Regressor	49.53	1.40
K-Neighbors Regressor	52.14	1.34
Elastic Net Regressor	70.18	0.96

Table 5.4: Mean RMSE and Standard Deviation of Different Regressors

Interestingly, the performance of more complex models such as the Random Forest Regressor and Gradient Boosting Regressor is suboptimal, with RMSE*s* of 46.73 and 41.85 respectively. These regressors typically perform well on many datasets due to their ability to handle high-dimensional data and capture non-linear relationships. However, in this case they underperform compared to simpler models such as Linear Regression and Ridge, suggesting that the features in the dataset may not be well suited to tree-based models.

Support Vector Regressor, Decision Tree Regressor and K-Neighbors Regressor have RMSE*s* higher than 47, indicating their poor performance in this task. The low score of these models may be due to their difficulty in dealing with the high complexity of the dataset and the distribution of the target variable.

Finally, the Elastic Net Regressor has the highest RMSE of 70.18, reflecting the worst performance of all the models. This could be due to the model's *l1 regularisation*, which, while useful in preventing overfitting, may have led to *underfitting* in this case.

5.4.3.2 Conclusion

Despite the good accuracy of the graph model regressor, the fact that the MLP Regressor slightly outperforms the DGCNN Regressor provides an important finding. It implies that, contrary to what was initially assumed, graph representation does not necessarily provide a substantial advantage over simple tabular representation in the case of this particular energy prediction task.

There are several factors that could contribute to this. It may be that the complexity of the graph-structured data does not provide additional benefits in predicting energy consumption values, especially compared to a well-designed tabular representation. The energy consumption of apartments may be primarily influenced by characteristics that are easily captured in a tabular format, such as apartment size, number of rooms or location, while the added topological information of the graph structure may not contribute significant predictive value.

In conclusion, the choice of data representation and machine learning model was found to have a significant impact on performance, highlighting the need for task-specific and data-driven model selection. While the DGCNN Classifier excelled in the classification task, reinforcing the value of graph data for this particular problem, the MLP Regressor outperformed the DGCNN Regressor in the regression task, suggesting that tabular data may be equally or even more effective in data-driven energy performance prediction. This requires a detailed understanding of the benefits and limitations of different data representations and models according to the specific characteristics of the dataset and the nature of the prediction task. Future research can build on these findings by exploring other sophisticated models and data transformations to take advantage of the relational information contained in graphical architecture datasets.

Chapter 6

Conclusion and Future Perspectives

The results of this study demonstrate the potential of applying topological knowledge graphs and graph machine learning to the architectural design process. The synthetic dataset generation process, which included the collection of space partitioning algorithms, allowed the creation of an unbiased architectural dataset with significant variability. Both the classification and regression models achieved high accuracy in terms of energy prediction, indicating the potential for a useful implementation of the described methods in the relevant industry. However, when compared to other machine learning models, the application of a simple neural network (MLP) trained on tabular rather than graphical data showed comparably good results. This suggests that the application of topological knowledge graphs in the context of energy simulation is not particularly advantageous compared to simple tabular data. Nevertheless, further research is needed before the usefulness of graphical topological information in the field of energy consumption prediction can be determined.

Furthermore, the importance of graph theory in architectural design is demonstrated, questioning the current role and application of building simulations in architecture. Although the proposed application in this study is currently a proof of concept rather than a complete tool proposal, it demonstrates the possibility of improving the architectural workflow and integrating architectural graph representation into design, as well as optimising simulation and design interaction. The implications of this study are broad, ranging from the advancement of graph-theoretic applications in architectural research to the enhancement of creative, artistic and organisational aspects of the architectural profession.

However, there are limitations to the proposed methods, including the lack of complex architectural models with concave spaces or patios in the dataset and the high degree of generalisation in the energy simulations. The choice of partitioning algorithm was limiting but necessary due to resource limitations that made the application of complex iterative optimisation methods more restrictive. Lack of access to an unbiased large graph dataset, limited machine power, time constraints and the difficulty of establishing connections and communication with the AEC were also limiting factors.

Despite these limitations, this study opens new avenues for research in the field of architecture and demonstrates the potential of topological information coupled with knowledge graphs in the context of automated architectural simulation and analysis methods. With further development and optimisation, this approach has the potential to find a valuable application in architectural design, allowing for more efficient building practices.

By presenting and discussing the various simulation and topological analysis methods, this thesis provides a basis for further research into graphical machine learning methods in the architectural context. Furthermore, the detailed documentation of the methodology, the software used, and the public availability of all data and code (section 6.1) allows for subsequent research based on this contribution.

6.1 Summary of Contribution

One of the main contributions of this thesis is the development of a comprehensive framework for the synthetic generation of a complete graph dataset that accurately represents real-world building designs. This involved the gathering of topological analysis methods which were then used to extract knowledge graph objects from the corresponding apartment designs. By exploring and comparing different algorithms, this research identified the most appropriate methods for creating diverse apartment designs with their topological organisation represented by graph networks. This synthetic dataset was then enriched with energy performance scores and values, allowing it to be used to train different machine learning algorithms to predict and classify the energy consumption of buildings based on their graph representations.

A detailed introduction and presentation of the concepts of graph theory, topology and simulation provides insights into the respective fields and their connection to architecture. Essential methods of each field are explained and described, such as the different graphical analysis methods, topological analysis concepts in concrete architectural applications, graph machine learning algorithms, and the most relevant simulation techniques. This summary thus provides a knowledge base for further research in the areas described.

The synthetic generation of the graph dataset involved several crucial steps, including the creation of a parametric generation pipeline and the integration of the identified space partitioning algorithms into the parametric framework for automatic floor plan generation. The defined rules and architectural requirements for the properties of each object in the dataset were also carefully considered and a post-processing phase was conducted to ensure the accuracy and reliability of the dataset. This synthetic dataset can thus serve as a useful tool for training graph machine learning algorithms to predict various parameters based on chosen architectural simulation metrics.

Another significant contribution of this thesis is the use of graph machine learning to predict the energy performance of buildings based on their graph representations. The trained regressor and classifier were able to accurately predict the energy performance of unseen apartment designs based on their topology, demonstrating the potential of integrating such methods into the architectural design process. This research has also shown the importance of feedback in the early design stages to optimise building performance, and that graphical information could be a beneficial alternative to traditional unstructured data for analytical applications.

This research has thus contributed to the field of architectural and conceptual design by demonstrating the integration of theoretical concepts such as graph theory, topology, synthetic data generation, automatic floor planning and graph machine learning in the design process. The generated dataset and the corresponding code will be made available under an open source licence, thus being open for contribution while allowing any kind of use and modification. The dataset is hosted on Google's data science platform *kaggle* (`https://kaggle.com/datasets/rabanohlhoff/architecture-graph-dataset`) and the code for the parametric framework as well as the machine learning procedure with the trained models is hosted on Microsoft's code sharing platform *GitHub* (`https://github.com/Sinasta/thesis`).

6.2 Reasearch Questions

This work brings to light the findings and advances that can be achieved by integrating complex computational and mathematical methods into architectural design practices. In doing so, it opens up a range of new possibilities and tools for architectural design, but also presents new implications and challenges that need to be addressed. The following section proposes answers to the four key research questions that arise from the interplay between architecture, graph theory, topology, machine learning and synthetic dataset creation. These questions, previously announced in section 1.4, explore the potential benefits and challenges of these integrations, ultimately aiming to provide a holistic understanding of the opportunities and complexities they bring to the field of architecture. The ensuing discussion is intended to provide valuable insights into these research topics and to encourage further exploration and innovation in these areas.

6.2.1 Graph and Topology in Architecture

The integration of knowledge graphs and topological methods in architectural practice holds immense potential[1]. Graphs provide a powerful tool for representing relational information between architectural entities, while topology can be used to understand and manipulate the spatial as well as non-spatial relationships between these entities. The meaningful application of these tools in project design can allow architects to derive insights from abstract relationships and potentially influence their design decisions in novel ways. However, such integration also presents challenges. Architects may need to acquire new skills and familiarise themselves with complex mathematical concepts, and a thoughtful approach to translating abstract relationships into concrete design decisions is required.

Successful integration into the everyday design process would bring significant programmatic, organisational and conceptual benefits to the architect, adding an essential tool to the conventional creative process. The ability to graphically represent abstract information, such as relational or hierarchical relationships between architectural elements, provides the basis for quantifying and analysing previously intangible design intentions.

6.2.2 Feedback in Early Design Stages

Feedback in the early stages of design can be invaluable in enhancing creativity and coherence between initial concepts and more detailed elaborations in later stages[2]. By receiving indicative feedback early on, architects can take informed design decisions and ensure that their initial designs are well aligned with the requirements of later project stages. However, integrating such feedback can be challenging due to the ambiguous and exploratory nature of early design stages. Machine learning models can be leveraged to provide immediate, data-driven feedback to architects, helping them to iteratively improve their designs and optimise the development process.

[1]Alymani, Mujica, et al. 2023, "Classifying building and ground relationships using unsupervised graph-level representation learning".

[2]Paterson et al. 2013, "Real-time Environmental Feedback at the Early Design Stages".

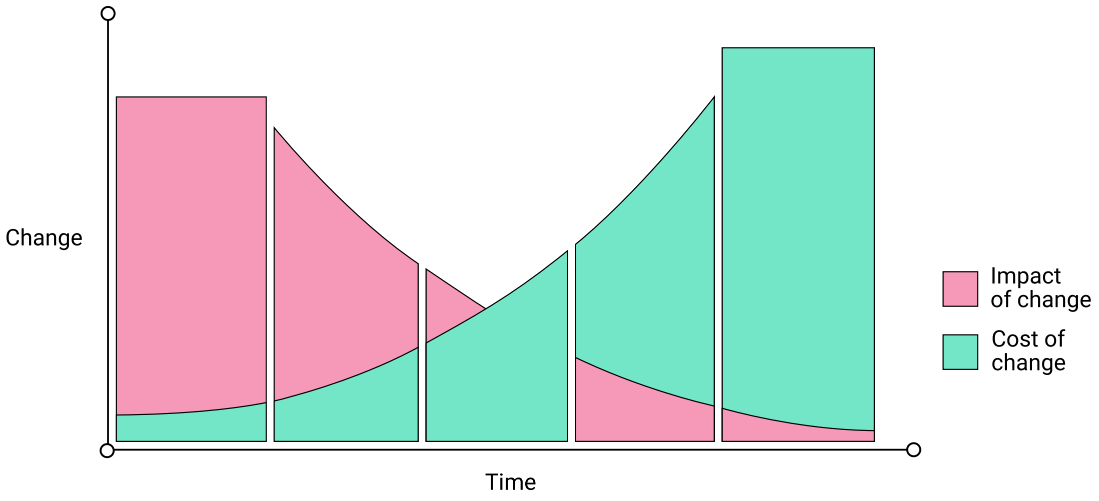


Figure 6.1: Design for Manufacturing Principle

In this work, the integration of energy performance feedback was explored using schematic apartment floor plans and their respective window positions. In a concrete application, this would allow the designer to receive immediate feedback on the energy profile of the drawn architectural object. It would also allow a comparative study of design performance by simply moving individual apertures or changing the dimensions and topology of the volumes without the need for complex simulations. This method of instant design feedback can be extended to any simulation parameter, making difficult to understand characteristics of the design object clear and assessable.

As illustrated in the *design for manufacturing principle* (figure 6.1), it becomes increasingly costly to change the design as time progresses during the design process, while the impact of the change steadily decreases. This shows that the best time to make a design change is as early as possible in the project. Since simulation and other feedback operations in the traditional architecture process occur mostly in the later stages of the project, such changes are only possible on a small scale or in a very costly framework[3]. This clearly demonstrates the essential need for early design feedback operations such as those demonstrated in this work.

6.2.3 Machine Learning in Architecture

Machine learning models can play an important role in architectural design[4]. In both practice and academic research, the application of deep learning methods can help to abstract geometric or topological information from complex architectural data in order to provide detailed insights. Trained models can therefore serve as valuable tools for design feedback iteration, transforming data into actionable recommendations for design improvement[5]. In particular, the present study found that Deep Graph Convolutional Neural Network*s*, which accept annotated graphs as input, are well suited to this task. However, a significant degree of technical skill and understanding of these models is required to adapt the

[3]As, Pal, and Basu 2018, "Artificial intelligence in architecture: Generating conceptual design via deep learning".

[4]Kiavarz et al. 2021, "Room-based energy demand classification of BIM data using graph supervised learning."

[5]Paterson et al. 2013, "Real-time Environmental Feedback at the Early Design Stages".

training process in order to use them effectively in the architectural context. In this regard, important challenges related to bias in training data and interpretability of machine learning models need to be carefully considered and addressed.

The abstraction of BIM models to topological knowledge graphs proved to be an advantageous method for dataset creation and the subsequent training process of graph machine learning algorithms. By a sophisticated selection of the node and graph labels of the respective datasets, promising results can be achieved in the architectural application.

6.2.4 Synthetic Architecture Datasets

The synthetic creation of architectural graph datasets is an innovative approach that can abstract complex relational information. Automatic floor plan generation can provide a rich source of diverse and adaptable design data[6], providing a valuable resource for machine learning models. This approach can help mitigate origin bias by providing a broader and more varied representation of architectural designs than a dataset based on the analysis of a limited number of architecture plans or styles. However, the automatic generation of such datasets also presents challenges. Ensuring the quality, realism and variability of the generated floor plans requires careful tuning of the generation algorithm. Evaluating the creative diversity and adaptability offered by synthetic datasets also poses methodological challenges, requiring metrics that capture both the quantitative and qualitative characteristics of architectural designs.

The dataset generation pipeline developed in this thesis provides an extensible framework for dataset synthesis. The technical decomposition into architectural control rules and extensible generation algorithms is thus ideally suited for adaptation to the specific requirements of the graph datasets to be generated.

6.3 Learned lessons

Through the development of an end-to-end framework for the synthetic generation of a complete graph dataset and the use of machine learning algorithms to predict the energy performance of buildings based on their graph representations, this thesis has uncovered several important lessons for the field of Computer Aided Design. These lessons have the potential to impact architectural research and design practice in terms of topological analysis methods through the application of graph machine learning models. In this section, these learned lessons will be explored in more detail, including the accuracy of the machine learning models, the importance of graphical information and retrieval, and the potential of DGCNN*s* in architectural design optimisation. These learned lessons provide valuable insights for researchers, architects and designers looking to integrate topological and graph-theoretic analysis methods into their research or design processes.

- By applying topological and graphical analysis methods to architecture, mathematical concepts can be used to automatically explore and evaluate possible design proposals, providing the designer with a useful tool in the creative process.
- Graph machine learning algorithms can achieve good accuracy for both classification and regression tasks when predicting the energy performance of buildings based on their graph representations. This demonstrates the potential of using graph machine

[6]Carta 2021, “Self-Organizing Floor Plans”.

learning algorithms as a tool for optimising building designs and improving their energy efficiency. By accurately predicting the energy performance of buildings, it becomes possible to make more informed decisions and to design more efficient and sustainable building layouts.

- The comparable accuracy of the Deep Graph Convolutional Neural Network and Multi-layer Perceptron neural network developed in this work indicates that the role of the topological relationships between individual spaces and their respective apertures do not have a significant impact on the energy performance of individual apartments.
- However, the relation between the annotated graph information, specifically node labels, and energy performance has been shown to be of critical importance. By carefully selecting and defining node labels in the graph representations of building designs, machine learning algorithms can predict their energy performance with high accuracy. This highlights the importance of selecting meaningful and informative node labels when generating graph datasets for architectural design.
- The ability to retrieve the graphical representation of the BIM model throughout all phases of the project allows architects and designers to continuously verify and optimise their building designs based on different graph analysis methods. This highlights the significance of integrating graph theory tools into the traditional BIM workflow to optimise the design and organisation of building topologies.

6.4 Future Perspective

Looking forward, this thesis has identified several important open issues and implications for the future of the use of graph representations and machine learning algorithms in architectural design. Addressing these open issues and considering their implications will be crucial for architectural research in this area in order to fully exploit the potential of such tools in order to produce more optimised and efficient building designs.

An important open issue is the identification of other values beyond energy performance that can be used as labels for graphs and predictions. These could include other building performance parameters such as light simulation, fire simulation, evacuation simulation or spatial syntax evaluation. In addition, it will be important to expand the dataset to include more features, such as different locations, materials, heights and contexts, to fully explore the potential of graph machine learning algorithms in the architectural design context.

Further research into model optimisation is another important open question. While this work has demonstrated the use of Deep Graph Convolutional Neural Network, further improvements in the structure of the model could be made. Another open issue is the implementation of a continuous back and forth between the graph representation and the BIM model, in particular using the IFC file format[7], which is hierarchical and can therefore be queried in a structured manner.

An important implication of this research is that machine learning algorithms should not be used primarily as recommendation tools, but rather as useful feedback indicators for architects and designers. In addition, the use of these tools should not be limited to

[7]Isaac, Sadeghpour, and Navon 2013, “Analyzing building information using graph theory”.

the architectural side, but should be extended to the engineering side of building design. Furthermore, it is important to note that the synthetic dataset used in this research should be adapted or enhanced for specific use cases to ensure its effectiveness.

To address these open issues and implications, several recommendations can be made. Adapting the synthetic dataset generation algorithms to represent more real-world variety in terms of geometry, adding variations for composite materials such as walls and windows, and simulating variations for different locations are important steps. Exploring the benefits of the IFC file format for graph retrieval[8] is also recommended, as it could provide a simple and potentially widely adaptable method for integrating topological graph analysis into the conventional design workflow.

6.5 Added Value for Architects

The development and application of a comprehensive framework for the synthetic generation of a complete graph dataset and the use of machine learning algorithms to predict the simulated performance of buildings based on their graph representations provides significant added value to architects and designers. By integrating machine learning algorithms and graph representations into their design processes, architects can create more efficient and optimised building designs. The use of machine learning algorithms to predict the energy performance of buildings, as explored in this work, can help architects and designers in making more educated choices and optimise their building designs to reduce energy consumption and increase energy efficiency. This can lead to significant cost savings over the lifecycle of the building, as well as improved environmental performance.

In addition, the use of graph representations can provide architects and designers with a powerful tool for abstracting and visualising the complex relationships and dependencies within building designs. In fact, graph representations can be used to visualise and analyse building designs at different scales and to identify patterns and relationships that are not immediately apparent in conventional design workflows. This can help architects to better understand the implications of their design decisions and to make choices about the materials, systems and technologies used in their building designs.

The use of machine learning algorithms and graph representations can also provide architects with a valuable tool for early design optimisation. By using predictive models to analyse different design options, designers can quickly and efficiently evaluate the potential performance of different three-dimensional layout proposals and modifications. This can help reduce the time and cost associated with traditional feedback methods and ensure that the final design is optimised in terms of the chosen performance parameter.

6.6 Concluding Remarks

This thesis has demonstrated the potential of applying topological graphs in the context of architectural design. Throughout the process, from conceptual clarifications to the synthetic generation of a graph dataset to the training of machine learning models, essential foundations have been laid for research into the application of graphical-topological analysis methods to architectural objects. Several implications, open questions, perspectives and benefits have been identified in the course of this research.

[8]Nahar 2017, “Applying graph theory concepts for analyzing BIM models based on IFC standards”.

Although a concrete implementation regarding the energy consumption of individual apartments was developed in the contribution section, this work is primarily a proof of concept to lay the foundations for further research in this field. In this sense, the individual stages of the experiment can now be further refined by applying the concepts explained, and subsequently compared and evaluated using the established evaluation metrics. This can be done both at the level of data generation, by varying the geometric methods, simulation strategies or information annotation, and at the level of model training, by modifying and combining different Graph Neural Networks.

The potential of graph theory and topology to represent abstract relational information in architectural designs has been established, while at the same time highlighting the need for their meaningful interpretation in the context of project design. While these methods open up new ways of conceptualising and analysing architectural design[9], they also pose the challenge of deciphering the abstract representations and making them practically relevant.

The early integration of indicative simulation variables in the design process is a way to promote creativity and coherence in architectural projects. However, it requires the development of effective methods to manage and optimise this process, a task that has been explored and discussed in this thesis.

Machine learning, a field that has seen tremendous growth and advancement in recent years, has shown immense potential in the field of architecture. As has been explored, machine learning models can not only provide nuanced feedback in design iterations, but can also effectively process complex inputs such as annotated graphs. The exploration of DGCNN as a novel model for graph-based machine learning has presented an innovative approach to architectural design. However, the task of selecting the right model, fine-tuning the hyperparameters, and interpreting the outputs remains a complex and challenging process.

While the creation of synthetic datasets offers a promising solution to address origin bias and increase creative diversity in architectural design, it comes with its own challenges. From ensuring the validity of the generated data to dealing with the complexities of automatic floor plan generation, many facets of this promising yet challenging field have been traversed.

The interdisciplinary approach of this work, between mathematical and computer science methods and those of conceptual architecture, aims to enrich the creative design process with valuable tools, thus bridging the gap between the individual disciplines. Since effective progress would hardly be possible without collaboration and public availability of research and results, an essential ambition of this work is the publication of all code sources as well as data and information (section 6.1).

[9]Boguslawski et al. 2016, "Two-graph building interior representation for emergency response applications"; Nauata, Chang, et al. 2020, "House-gan: Relational generative adversarial networks for graph-constrained house layout generation"; Z. Wang et al. 2021, "Room Type Classification for Semantic Enrichment of Building Information Modeling Using Graph Neural Networks".

Bibliography

Abualdenien, Jimmy and André Borrmann (2021). "PBG: A parametric building graph capturing and transferring detailing patterns of building models". In: *Proc. of the CIB W78 Conference 2021.*

Aguiar, Rita, Carmo Cardoso, et al. (2017). "Algorithmic design and analysis fusing disciplines". In: *Disciplines + Disruption - ACADIA.*

Aish, Robert et al. (2018). "Topologic: tools to explore architectural topology". In: *AAG.*

Aksin, Feyza and Semra Selçuk (2021). "Use of Simulation Techniques and Optimization Tools for Daylight, Energy and Thermal Performance: The case of office module (s) in different climates". In: *Simulation, prediction and evaluation in design - eCAADe* 2, pp. 409–418.

Alammar, Ammar, Wassim Jabi, and Simon Lannon (2021). "Predicting incident solar radiation on buildings envelope using machine learning". In: *SimAUD.*

Alexander, Christopher (1977). *A pattern language: towns, buildings, construction.* Oxford university press.

Ali, Tarique (2023). "Architectural Pipeline: An experiment into the role of topological graphs in the early stages of architectural design in the era of machine learning". In.

Alymani, Abdulrahman, Wassim Jabi, and Padraig Corcoran (2022). "Graph machine learning classification using architectural 3D topological models". In: *Simulation*, pp. 1–15.

Alymani, Abdulrahman, Andrea Mujica, et al. (2023). "Classifying building and ground relationships using unsupervised graph-level representation learning". In: *Design Computing and Cognition22.* Springer, pp. 305–320.

An, Phan Thanh, Phong Thi Thu Huyen, and Nguyen Thi Le (2021). "A modified Grahams convex hull algorithm for finding the connected orthogonal convex hull of a finite planar point set". In: *Applied Mathematics and Computation* 397, pp. 125–134.

Anuradha, V, Minal Sabnis, and Vennila Thirumavalavn (2008). "Voronoi diagram voro: Application of interactive weighted Voronoi diagrams as an alternate master-planning framework for business parks." In: *Proceedings of the 13th International Conference on Computer-Aided Architectural Design Research in Asia*, pp. 399–408.

As, Imdat and Prithwish Basu (2021). *The Routledge companion to artificial intelligence in architecture.* Vol. 1. Routledge, Taylor & Francis Group.

As, Imdat, Siddharth Pal, and Prithwish Basu (2018). "Artificial intelligence in architecture: Generating conceptual design via deep learning". In: *International Journal of Architectural Computing* 16.4, pp. 306–327.

Baglivo, Jenny A and Jack E Graver (1983). *Incidence and symmetry in design and architecture.* Cambridge University Press Cambridge.

Balzer, Michael and Oliver Deussen (2005). "Voronoi treemaps". In: *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.* IEEE, pp. 49–56.

Bao, Fan et al. (2013). "Generating and exploring good building layouts". In: *ACM Transactions on Graphics (TOG)* 32.4, pp. 1–10.

Belém, Catarina, Luis Santos, and António Leitão (2019). “On the Impact of Machine Learning: Architecture without Architects”. In: *CAAD Futures*, pp. 247–293.

Bielski, Jessica et al. (2020). “Topological Queries and Analysis of School Buildings Based on Building Information Modeling (BIM) Using Parametric Design Tools and Visual Programming to Develop New Building Typologies”. In: *eCAADe-Education and research in Computer Aided Architectural Design in Europe*.

Boguslawski, Pawel et al. (2016). “Two-graph building interior representation for emergency response applications”. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 3.

Boon, Christopher et al. (2015). “Optimizing spatial adjacencies using evolutionary parametric tools: using grasshopper and galapagos to analyze, visualize, and improve complex architectural programming”. In: *Perkins+ Will Research Journal* 7.2, pp. 25–37.

Caetano, Inês, Luís Santos, and Antonio Leitao (2020). “Computational design in architecture: Defining parametric, generative, and algorithmic design”. In: *Frontiers of Architectural Research* 9.2, pp. 287–300.

Caldas, Luìsa G and Luìs Santos (2012). “Generation of Energy-Efficient Patio Houses With GENE-ARCH”. In: *Combining an evolutionary generative design system with a shape grammar*, pp. 459–470.

Calixto, Victor and Gabriela Celani (2015). “A literature review for space planning optimization using an evolutionary algorithm approach: 1992-2014”. In: *Blucher Design Proceedings* 2.3, pp. 662–671.

Campo, Matias del (2020). “How Machines Learn to Plan - A Critical Interrogation of Machine Vision Techniques in Architecture.” In: *Proceedings of the ACADIA Conference 2020*.

Canestrino, Giuseppe (2021). “On the influence of evolutionary algorithm (EA) optimization in architectural design: a reflection through an architectural envelope’s shadowing system design”. In: *Architecture in the Age of Disruptive Technologies: Transformation and Challenges*. Robert Gordon University.

Canestrino, Giuseppe et al. (2020). “Generating architectural plan with evolutionary multiobjective optimization algorithms: a benchmark case with an existent construction system”. In: *Proceedings of SiGraDi 2020*, pp. 149–156.

Carta, Silvio (2021). “Self-Organizing Floor Plans”. In: *Harvard Data Science Review HDSR*.

Chaillou, Stanislas (2021). “AI and architecture: An experimental perspective”. In: *The Routledge Companion to Artificial Intelligence in Architecture*. Routledge, pp. 420–441.

– (2022). *Artificial Intelligence and Architecture: From Research to Practice*. Birkhäuser.

Chatzikonstantinou, Ioannis (2014). “A 3-dimensional architectural layout generation procedure for optimization applications: DC-RVD”. In: *Proceedings of the 2014 eCAADe Conference*. Vol. 1, pp. 287–296.

Chatzivasileiadi, Aikaterini et al. (2018). “The effect of reducing geometry complexity on energy simulation results”. In: *Simulation, Prediction and Evaluation Explorations - eCAADe 36* 2.

Chen, Jielin and Rudi Stouffs (Apr. 2022). “Robust Attributed Adjacency Graph Extraction Using Floor Plan Images”. In: DOI: `10.52842/conf.caadria.2022.2.385`.

Coates, Paul et al. (2005). “Generating architectural spatial configurations. Two approaches using Voronoi tessellations and particle systems”. In: *8th Generative Art Conference GA2005*.

Community, Blender Online (2018). *Blender - a 3D modelling and rendering package*. Blender Foundation. Stichting Blender Foundation, Amsterdam. URL: `http://www.blender.org`.

Crawley, Drury B et al. (2001). “EnergyPlus: creating a new-generation building energy simulation program”. In: *Energy and buildings* 33.4, pp. 319–331.

Das, Subhajit et al. (2016). “Space plan generator: Rapid generation and evaluation of floor plan design options to inform decision making”. In: *ACADIA - Posthuman Frontiers.*

Dawes, Michael J and Michael J Ostwald (2013). “Applications of graph theory in architectural analysis: past, present and future research.” In: *NOVA*.

Dawes, Michael J, Michael J Ostwald, and Ju Hyun Lee (2021). “Examining control, centrality and flexibility in Palladio’s villa plans using space syntax measurements”. In: *Frontiers of Architectural Research* 10.3, pp. 467–482.

Earl, CF and LJ March (1979). “Architectural applications of graph theory”. In: *Applications of graph theory*, pp. 327–355.

Egor, Gavrilov et al. (2020). “Computer-aided approach to public buildings floor plan generation. Magnetizing Floor Plan Generator”. In: *Procedia Manufacturing* 44, pp. 132–139.

Eisenstadt, Viktor, Klaus-Dieter Althoff, and Christoph Langenhan (2020). “Student Graduation Projects in the Context of Framework for AI-Based Support of Early Conceptual Phases in Architecture.” In: *LWDA*, pp. 174–179.

Eisenstadt, Viktor, Hardik Arora, Christoph Ziegler, Jessica Bielski, Christoph Langenhan, K Althoff, et al. (2021). “Comparative evaluation of tensor-based data representations for deep learning methods in architecture”. In: *Proceedings of the 39th eCAADe conference.* Vol. 1, pp. 45–54.

Eisenstadt, Viktor, Hardik Arora, Christoph Ziegler, Jessica Bielski, Christoph Langenhan, Klaus-Dieter Althoff, et al. (2021). “Exploring optimal ways to represent topological and spatial features of building designs in deep learning methods and applications for architecture”. In: *CAADRIA*.

Eisenstadt, Viktor, Jessica Bielski, et al. (2022). “Autocompletion of Design Data in Semantic Building Models using Link Prediction and Graph Neural Networks”. In: *Education and research in Computer Aided Architectural Design in Europe Conference.*

Eisenstadt, Viktor, Christoph Langenhan, and Klaus-Dieter Althoff (2019). “Generation of Floor Plan Variations with Convolutional Neural Networks and Case-based Reasoning–An Approach for Unsupervised Adaptation of Room Configurations within a Framework for Support of Early Conceptual Design.” In: *eCAADe SIGraDi Conference, Porto.*

Fedorova, Stanislava et al. (2021). “Synthetic 3d data generation pipeline for geometric deep learning in architecture”. In: *arXiv preprint arXiv:2104.12564*.

Franz, Gerald, Hanspeter A Mallot, and Jan M Wiener (2005). “Graph-based models of space in architecture and cognitive science: A comparative analysis”. In: *17th International Conference on Systems Research, Informatics and Cybernetics (INTERSYMP 2005).* International Institute for Advanced Studies in Systems Research and Cybernetics, pp. 30–38.

Fuchkina, Ekaterina et al. (2022). “Space Matcher”. In: *Legal Depot D/2022/14982/02*, p. 39.

Galle, Per (1981). “An algorithm for exhaustive generation of building floor plans”. In: *Communications of the ACM* 24.12, pp. 813–825.

Grzesiak-Kope, Katarzyna, Barbara Strug, and Grazyna lusarczyk (2021). “Evolutionary methods in house floor plan design”. In: *Applied Sciences* 11.17, pp. 1–14.

Guglielmetti, Rob, Dan Macumber, and Nicholas Long (2011). “OpenStudio: an open source integrated analysis platform”. In.

Guo, Zifeng and Biao Li (2017). “Evolutionary approach for spatial architecture layout design enhanced by an agent-based topology finding system”. In: *Frontiers of Architectural Research* 6.1, pp. 53–62.

Harding, John and Christian Derix (2011). “Associative spatial networks in architectural design: Artificial cognition of space using neural networks with spectral graph theory”. In: *Design Computing and Cognition10*. Springer, pp. 305–323.

Harris, Charles R. et al. (2020). “Array programming with NumPy”. In: *Nature* 585, pp. 357–362. DOI: `10.1038/s41586-020-2649-2`.

Hauck, Anthony (2017). “Energy Model Machine (EMM)”. In: *SharingofComputableKnowledge!*, p. 277.

Hillier, Bill and Julienne Hanson (1989). *The social logic of space*. Cambridge university press.

Hong, Tzu-Chieh Kurt and Athanassios Economou (2022). “Five criteria for shape grammar interpreters”. In: *Design Computing and Cognition20*. Springer, pp. 191–207.

Hu, Ruizhen et al. (2020). “Graph2plan: Learning floorplan generation from layout graphs”. In: *ACM Transactions on Graphics (TOG)* 39.4, pp. 118–1.

Ibrahim, Mohamed Naeim A (2011). “Computing Architectural Layout”. In: *University of Salford*.

Isaac, S, F Sadeghpour, and R Navon (2013). “Analyzing building information using graph theory”. In: *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*. Vol. 30. IAARC Publications, p. 1.

Jabi, Wassim (2013). *Parametric design for architecture*. Hachette UK.

– (2015). “The potential of non-manifold topology in the early design stages”. In: *Environmental parametrics* 1.

Jabi, Wassim, Robert Aish, et al. (2018). “Topologic: A toolkit for spatial and topological modelling”. In: *Computing for a Better Tomorrow, Proceedings of the 36th eCAADe conference*. Vol. 2, pp. 449–458.

Jabi, Wassim, Aikaterini Chatzivasileiadi, et al. (2019). “The synergy of non-manifold topology and reinforcement learning for fire egress”. In: *Design - Artificial intelligence - eCAADe 37* 2, pp. 85–94.

Jabi, Wassim, Barbara Grochal, and Adam Richardson (2013). “The potential of evolutionary methods in architectural design”. In: *Proc. 31st Int. Conf. Educ. Res. Comput. Aided Archit. Des. Eur.* Vol. 2, p. 217.

Joyce, Sam Conrad and Ibrahim Nazim (2021). “Limits to Applied ML in Planning and Architecture”. In: *Computational design - eCAADe*.

Kalervo, Ahti et al. (2019). “Cubicasa5k: A dataset and an improved multi-task model for floorplan image analysis”. In: *Image Analysis: 21st Scandinavian Conference, SCIA 2019, Norrköping, Sweden, June 11–13, 2019, Proceedings 21*. Springer, pp. 28–40.

Kantor, Jean-Michel (2005). “A tale of bridges: topology and architecture”. In: *Nexus Network Journal* 7, pp. 13–21.

Kelley, John L (1955). *General topology*. Courier Dover Publications.

Kiavarz, H et al. (2021). “Room-based energy demand classification of BIM data using graph supervised learning.” In: *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*.

Knecht, Katja and Reinhard König (2010). “Generating floor plan layouts with kd trees and evolutionary algorithms”. In: *GA2010-13th Generative Art Conference, Milan, Italy*.

Lakshmi, L, A Madhumathi, and M Sindhuja (2017). “Graph theory and architecture”. In: *MRF Biannual Peer Reviewed International Journal*.

Lee, Ju Hyun, Michael J Ostwald, and Hyunsoo Lee (2017). “Measuring the spatial and social characteristics of the architectural plans of aged care facilities”. In: *Frontiers of Architectural Research* 6.4, pp. 431–441.

Li, Yongzhi et al. (2009). *Design with space syntax analysis based on building information model.*

Liu, Zidong (Oct. 2021). “Topological Networks Using a Sequential Method Space Structure Simplifcation for Interactive Design”. In: *ACADIA*.

Lojanica, Vladimir and Maja Dragisic (2018). “The topological principles in the contemporary architectural design process”. In: *ICCM*.

Lopes, Ricardo et al. (2010). “A constrained growth method for procedural floor plan generation”. In: *Proc. 11th Int. Conf. Intell. Games Simul.* Citeseer, pp. 13–20.

Lu, Yueheng et al. (2021). “Organizational Graph Generation for Structured Architectural Floor Plan Dataset”. In: *International Conference of the Association for Computer-Aided Architectural Design Research in Asia* 1, pp. 81–90.

Marson, Fernando and Soraia Raupp Musse (2010). “Automatic real-time generation of floor plans based on squarified treemaps algorithm”. In: *International Journal of Computer Games Technology* 2010, pp. 1–10.

McKinney, Wes (2010). “Data Structures for Statistical Computing in Python”. In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman, pp. 56–61. DOI: `10.25080/Majora-92bf1922-00a`.

Moult, Dion (2020). *OSArch Community*. https://community.osarch.org/.

Muslimin, Rizal (2023). “Experience Grammar: Creative Space Planning with Generative Graph and Shape for Early Design Stage”. In: *Buildings* 13.4, p. 869.

Nagy, Danil (2021). “AI in space planning”. In: *The Routledge Companion to Artificial Intelligence in Architecture*. Routledge, pp. 160–181.

Nagy, Danil et al. (2017). “Project discover: An application of generative design for architectural space planning”. In: *Proceedings of the Symposium on Simulation for Architecture and Urban Design*, pp. 1–8.

Nahar, A (2017). “Applying graph theory concepts for analyzing BIM models based on IFC standards”. In: *Master’s Thesis. Technische Universität Dresden, Dresden, Germany.*

Napong, Nophaket (2004). “The graph geometry for architectural planning”. In: *Journal of Asian Architecture and Building Engineering* 3.1, pp. 157–164.

Nauata, Nelson, Kai-Hung Chang, et al. (2020). “House-gan: Relational generative adversarial networks for graph-constrained house layout generation”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*. Springer, pp. 162–177.

Nauata, Nelson, Sepidehsadat Hosseini, et al. (2021). “House-gan++: Generative adversarial layout refinement network towards intelligent computational agent for professional architects”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13632–13641.

Newton, David (2019). “Deep generative learning for the generation and analysis of architectural plans with small datasets”. In: *Proceedings of 37th eCAADe and 23rd SIGraDi Conference*. Vol. 2, pp. 21–28.

Nisztuk, Maciej and Pawe Myszkowski (2019). “Tool for evolutionary aided architectural design. Hybrid Evolutionary Algorithm applied to Multi-Objective Automated Floor Plan Generation”. In: *Conference: Ecaade Sigradi 2019 at: Porto, Portugal*. Vol. 1.

Nisztuk, Maciej and Pawe B Myszkowski (2019). “Hybrid evolutionary algorithm applied to automated floor plan generation”. In: *International Journal of Architectural Computing* 17.3, pp. 260–283.

Nortikin (2013). *Sverchok*. https://github.com/nortikin/sverchok.

Nourian, Pirouz (2016). “Configraphics: Graph theoretical methods for design and analysis of spatial configurations”. In: *Delft University of Technology.*

Nourian, Pirouz, Samaneh Rezvani, and IS Sariyildiz (2013). “Designing with Space Syntax: A configurative approach to architectural layout, proposing a computational methodology”. In: *eCAADe 2013: Computation and Performance–Proceedings of the 31st International Conference on Education and research in Computer Aided Architectural Design in Europe, Delft, The Netherlands, September 18-20, 2013.* Faculty of Architecture, Delft University of Technology; eCAADe (Education

Paszke, Adam et al. (2019). “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32.* Curran Associates, Inc., pp. 8024–8035. URL: `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

Paterson, Greig et al. (2013). “Real-time Environmental Feedback at the Early Design Stages”. In: *Simulation, Prediction and Evaluation - Computation and Performance* 2, pp. 79–86.

Pedregosa, F. et al. (2011). “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Pena, M Luz Castro et al. (2021). “Artificial intelligence applied to conceptual design. A review of its use in architecture”. In: *Automation in Construction* 124, pp. 1–30.

Postle, Bruno (2019). “On pattern languages, design patterns and evolution”. In: *New Design Ideas* 3.

Rogers, Scarlet et al. (2022). “Running, Running, Stop”. In: *Legal Depot D/2022/14982/02*, p. 19.

Ruiz-Montiel, Manuela et al. (2013). “Design with shape grammars and reinforcement learning”. In: *Advanced Engineering Informatics* 27.2, pp. 230–245.

Rycroft, Christopher Harley (2007). “Multiscale modeling in granular flow”. PhD thesis. Massachusetts Institute of Technology, Department of Mathematics.

Sanchez-Lengeling, Benjamin et al. (2021). “A gentle introduction to graph neural networks”. In: *Distill* 6.9, e33.

Schaffranek, Richard (2015). “Parallel planning-An experimental study in spectral graph matching”. In: *Proceedings of the 10th Space Syntax Symposium: Book of Abstracts.* Space Syntax Laboratory, The Bartlett School of Architecture, UCL, pp. 151–1.

Schultz, Carl and Mehul Bhatt (2011). “Toward accessing spatial structure from building information models”. In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 38, pp. 25–30.

Sebestyen, Adam and Jakub Tyc (2020). “Machine Learning Methods in Energy Simulations for Architects and Designers”. In: *Proceedings of the 38th eCAADe Conference*, pp. 613–622.

Shekhawat, Krishnendra (2014). “Algorithm for constructing an optimally connected rectangular floor plan”. In: *Frontiers of Architectural Research* 3.3, pp. 324–330.

Shekhawat, Krishnendra and José P Duarte (2019). “A graph theoretical approach for creating building floor plans”. In: *Computer-Aided Architectural Design." Hello, Culture" 18th International Conference, CAAD Futures 2019, Daejeon, Republic of Korea, June 26–28, 2019, Selected Papers 18.* Springer, pp. 3–14.

Shekhawat, Krishnendra, Nitant Upasani, et al. (2020). “GPLAN: Computer-Generated Dimensioned Floorplans for given Adjacencies”. In: *arXiv preprint arXiv:2008.01803.*

Singh, Manav Mahan et al. (2020). “Applying Deep Learning and Databases for Energy-efficient Architectural Design”. In: *Proceedings of the 38th International Online Conference on Education and Research in Computer Aided Architectural Design in Europe.*

Vol. 2. eCAADe (Education and Research in Computer Aided Architectural Design in Europe), pp. 79–87.

Slyadnev, S, A Malyshev, and V Turlapov (2017). “CAD model inspection utility and prototyping framework based on OpenCascade”. In: *Conference Paper: GraphiCon.*

Son, Kihoon and Kyung Hoon Hyun (2021). “A framework for multivariate data based floor plan retrieval and generation”. In: *26th International Conference of the Association for Computer-Aided Architectural Design Research in Asia: Projections. The Association for Computer-Aided Architectural Design Research in Asia, Hong Kong.* Vol. 29, pp. 2021–01.

Tarabishy, Sherif et al. (2020). “Deep learning surrogate models for spatial and visual connectivity”. In: *International Journal of Architectural Computing* 18.1, pp. 53–66.

Thurow, Torsten, Christoph Langenhan, and Frank Petzold (2016). “Assisting early architectural planning using a geometry-based graph search”. In: *eCAADe 2016* 2, pp. 199–207.

Van Treeck, Christoph and Ernst Rank (2004). “Analysis of building structure and topology based on graph theory”. In: *Technische Universität München.*

Vandromme, Johann et al. (2009). *An Interactive System Based on Semantic Graphs.*

Varoudis, Tasos and Sophia Psarra (2014). “Beyond two dimensions: architecture through three dimensional visibility graph analysis”. In: *The Journal of Space Syntax* 5.1, pp. 91–108.

Velikovi, Petar (2023). “Everything is Connected: Graph Neural Networks”. In: *arXiv preprint arXiv:2301.08210.*

Virtanen, Pauli et al. (2020). “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17, pp. 261–272. DOI: `10.1038/s41592-019-0686-2`.

Wang, Minjie et al. (2019). “Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks”. In: *arXiv preprint arXiv:1909.01315.*

Wang, Yan-Chao, Feng Lin, and Hock-Soon Seah (2019). “Orthogonal voronoi diagram and treemap”. In: *arXiv preprint arXiv:1904.02348.*

Wang, Zijian et al. (2021). “Room Type Classification for Semantic Enrichment of Building Information Modeling Using Graph Neural Networks”. In: *Proc. of the Conference CIB W78.* Vol. 2021, pp. 11–15.

Wardhana, Nicholas et al. (2019). “A spatial reasoning framework based on non-manifold topology”. In: *Design - Algorithmic and Parametric* 3.

West, Douglas Brent et al. (2001). *Introduction to graph theory.* Vol. 2. Prentice hall Upper Saddle River.

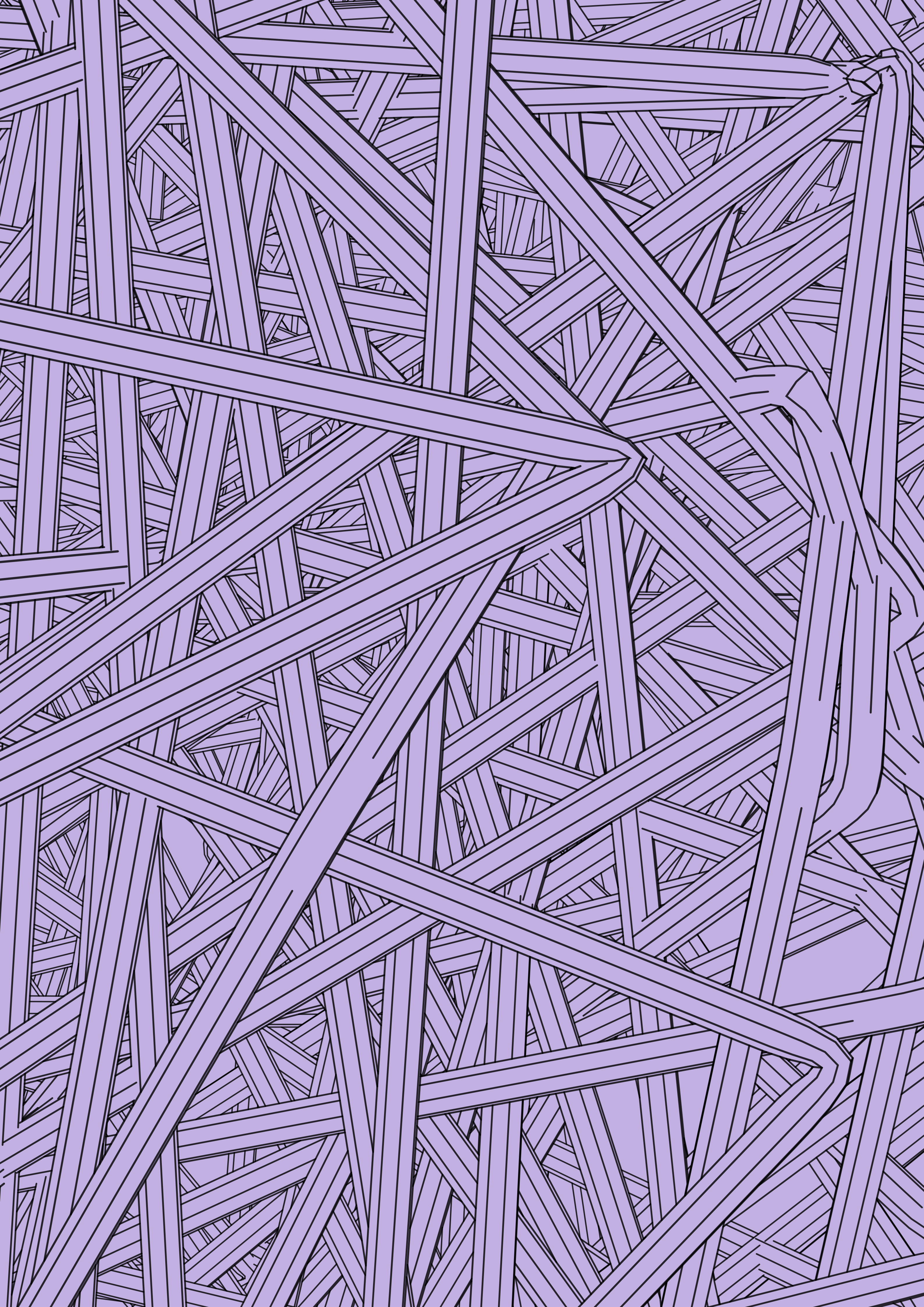
Wilson, Robin J (1979). *Introduction to graph theory.* Pearson Education India.

Yousif, S et al. (2021). “Deep-Performance: Incorporating Deep Learning for Automating Building Performance Simulation in Generative Systems”. In: *The Association for Computer-Aided Architectural Design Research in Asia (CAADRIA).* Vol. 1, pp. 151–160.

Zhang, Aston et al. (2021). “Dive into deep learning”. In: *arXiv preprint arXiv:2106.11342.*

Part III

Appendix



Appendix A

Additional Content

A.1 Further Readings

The state of the art (chapter 3) listed the main literature that provided fundamental information for the development of the experiments described in this study. However, the preliminary resource analysis revealed a much larger number of publications, research and experiments on the topics covered. For reasons of space and structure, those that did not contribute significantly to the development of the experiment, but still contained valuable and relevant information, were collected and summarised in the further reading section.

Graph Theory in Architecture

West et al. 2001, *Introduction to graph theory* is an in-depth study of graph theory with additional emphasis on the detailed explanation of essential graph-theoretic algorithms and special cases such as *perfect graphs*, *matroids*, *random graphs* and *eigenvalues.*

Napong 2004, "The graph geometry for architectural planning" documents the creation of graphical networks to analyse the geometric properties of architectural or urban structures. The *minimum path graph* is computed based on the distance between each node to provide designers with insight into the geometric potential of the object through analysis of optimal node centrality and *edge passage capacity.*

Vandromme et al. 2009, *An Interactive System Based on Semantic Graphs* demonstrates the calculation of adjacency graphs based on apartment and studio layouts using different methods to evaluate the subgraphs generated by *Bayesian causal methods.* It synthesises a set of optimal floor plan layout graph variants. This work demonstrates the potential of adjacency graphs in the context of creative design practices at a conceptual level.

Lakshmi, Madhumathi, and Sindhuja 2017, "Graph theory and architecture" presents different ways of applying graph-based analysis approaches at both architectural and urban levels. Concepts such as *pseudo-graphs*, *disconnected graphs*, *degree and eccentricity based graphs*, *semi-graphs* and *cluster analysis* are discussed.

Nahar 2017, “Applying graph theory concepts for analyzing BIM models based on IFC standards” describes and develops a potential workflow from a IFC BIM model to two informative and distinct graph networks. The *meta-graph* aims to describe the relational information of the model, while the *object-graph* captures the physical relationships. The methodology described aims to manage, visualise and analyse the information within the BIM model.

Abualdenien and Borrmann 2021, “PBG: A parametric building graph capturing and transferring detailing patterns of building models” focuses on the application of so-called *parametric building graphs*, which capture constructive AEC detail patterns and automatically integrate them into new projects through *graph rewriting systems*, provided a pattern match is given. The application of graph theory in this work is thus at a physically constructive level, involving detailed investigation, where the nodes and edges of the graph represent the respective elements and connections of the detail patterns.

Topology and Space Syntax

Alexander 1977, *A pattern language: towns, buildings, construction* is a fundamental book of architectural theory which identifies a multitude of so-called patterns, each of which describes a specific problem and its solution. Together, these patterns form a *pattern language* that encompasses not only constructive but also social, psychological, political and urban aspects.

Kantor 2005, “A tale of bridges: topology and architecture” focuses on the mathematical concept of topology and makes it tangible through examples. It clarifies essential features such as the topological understanding of space as opposed to the geometric one.

Varoudis and Psarra 2014, “Beyond two dimensions: architecture through three dimensional visibility graph analysis” takes a closer look at the Visibility Graph Analysis (VGA) method of space syntax. It then develops a method for extending VGA to allow the integration of three-dimensional information into the analysis. This allows for a comprehensive understanding of spatial configuration in two and three dimensions.

J. H. Lee, Ostwald, and H. Lee 2017, “Measuring the spatial and social characteristics of the architectural plans of aged care facilities” explores the relationship between spatial configurations and cultural and social backgrounds using aged care facilities as an example. Several facilities from different countries and cultures are compared using isovist and Visibility Graph Analysis to draw conclusions about their differences and influences.

Lojanica and Dragisic 2018, “The topological principles in the contemporary architectural design process” looks at contemporary architecture through the lens of mathematical topology. Concepts of continuous deformation of geometric forms, such as *deformability*, *openness* and *continuity*, are illustrated with examples and their significance for architectural design is discussed.

Wardhana et al. 2019, "A spatial reasoning framework based on non-manifold topology" explores the benefits of non-manifold topologies for spatial reasoning and presents their implementation in the '*Topologic*' library. An example of automatic path finding within a topology model derived from a BIM model and its *dual graphs* is documented.

Bielski et al. 2020, "Topological Queries and Analysis of School Buildings Based on Building Information Modeling (BIM) Using Parametric Design Tools and Visual Programming to Develop New Building Typologies" studies the topology of different school buildings by analysing adjacency, accessibility, depth and flow. This information is transformed into graph objects, together with semantic data about the buildings and spatial functions, and analysed using machine learning methods to identify specific patterns and draw architectural conclusions.

Decisionmaking and Feedback-Tools

Bao et al. 2013, "Generating and exploring good building layouts" addresses the problem of defining a good building layout, which presents a significant challenge in the field of CAD. This task is achieved by defining specific evaluation metrics that allow users to make design decisions within the *local shape space* based on a *portal graph*.

Das et al. 2016, "Space plan generator: Rapid generation and evaluation of floor plan design options to inform decision making" explores the automated generation of design variants based on architectural requirements, similar to the previously mentioned publications. Different layouts are calculated using an evolutionary algorithm-based floor plan generator, taking into account certain user-defined constraints such as site outline, number of floors, total area and adjacencies.

Thurow, Langenhan, and Petzold 2016, "Assisting early architectural planning using a geometry-based graph search" creates a searchable architectural database using graph-based *semantic building fingerprints* to enable designers to explore multiple similar typologies in the early stages of the design process.

Nagy et al. 2017, "Project discover: An application of generative design for architectural space planning" includes the automated generation of layout variants based on user input. What makes it different, however, is the evaluation matrix developed, which evaluates a variety of architectural characteristics of the generated plans. A multi-objective genetic algorithm is then used to filter out the best optimised variant.

Eisenstadt, Langenhan, and K.-D. Althoff 2019, "Generation of Floor Plan Variations with Convolutional Neural Networks and Case-based Reasoning– An Approach for Unsupervised Adaptation of Room Configurations within a Framework for Support of Early Conceptual Design." This publication documents a methodology for generating numerous possible design variations that could represent the current design using graphically programmable information and a *generative adversarial network*. The motivation for this research was the common need for extensive formal research during the initial design phases and the time intensity that this task brings.

Shekhawat, Upasani, et al. 2020, “GPLAN: Computer-Generated Dimensioned Floorplans for given Adjacencies” explains the developed software '*GPLAN*', which is able to generate a series of rectangular floor plans according to the chosen parameters by specifying a desired programmatic adjacency graph. In addition, the research presented allows the generation of orthogonal floor plans, which allows for greater diversity. The second part of the paper discusses another method for the automatic dimensioning of drawn floor plans.

Son and Hyun 2021, “A framework for multivariate data based floor plan retrieval and generation” is primarily concerned with the generation and quantification evaluation of multivariate design data from floor plans. In this context, an automated generation pipeline for the application of the evaluation method was also developed. Key assessment parameters include the number and function of individual rooms, perimeter, room shape, adjacency and connectivity. However, other elements such as walls and doors and their respective positions and dimensions are missing from the evaluation method.

Optimisation in Early Design-Stages

Jabi, Grochal, and Richardson 2013, “The potential of evolutionary methods in architectural design” explores the beneficial application of evolutionary algorithms in conjunction with shape-packing algorithms in the design process. These automatic optimisation methods in two-dimensional space have proven their effectiveness in examples such as shading facade patterns and urban residential layouts. The key benefits of this method are the automation and optimisation of multiple design parameters that influence each other.

Boon et al. 2015, “Optimizing spatial adjacencies using evolutionary parametric tools: using grasshopper and galapagos to analyze, visualize, and improve complex architectural programming” experiments with the application of evolutionary parametric optimisation methods to improve architectural layouts by defining desired parameters using a visual scripting language and genetic solver extensions. However, the chosen fitness criteria still pose a problem due to the lack of a uniform evaluation syntax.

Grzesiak-Kope, Strug, and lusarczyk 2021, “Evolutionary methods in house floor plan design” shows a novel approach to define optimal solutions. In fact, fitness criteria can take different forms in architectural optimisation applications. In this example, the genotypes represent two-dimensional vectors analogous to wall junctions in geometric space. Thus, within a defined framework, the placement of walls corresponding to desired room sizes and total area can be automatically adjusted to achieve the most optimal layout.

Performance-Based Design

Ibrahim 2011, “Computing Architectural Layout” conducts an experiment on several architectural pavilions and their topological shapes to optimise the use of space. Graphical parameters, such as the topological relationship of each space, are defined as an optimisation function to generate an adapted morphology. The formal freedom given to the structure in this experiment is interesting; however, it makes the developed method less applicable in the specific domain of residential architecture.

Aksin and Selçuk 2021, "Use of Simulation Techniques and Optimization Tools for Daylight, Energy and Thermal Performance: The case of office module (s) in different climates" shows an example of performance optimisation through the definition of multiple evaluation metrics. The application of conventional genetic algorithms has attempted to optimise daylighting and thermal performance while minimising energy consumption. However, the continuous simulation of individual building morphologies proves to be resource and time intensive, making it unsuitable as a supporting tool in the initial design phase.

Impact of Machine Learning

Belém, Luis Santos, and Leitão 2019, "On the Impact of Machine Learning: Architecture without Architects" examines the role of machine learning in its respective application areas and then explores its adaptation in the AEC industry. Possible applications in different architectural areas such as conceptualisation, algorithmisation, modelling and optimisation tasks are discussed and their significance for the future of the industry is explained and evaluated.

Pena et al. 2021, "Artificial intelligence applied to conceptual design. A review of its use in architecture" represents a collection of projects and applications of artificial intelligence in architecture, focusing on its beneficial use in the experimental design phase. Of interest is the emphasis on the creative capacities of trained models and computer science algorithms, allowing the discovery of innovative and original forms.

Design and Optimisation Applications

Harding and Derix 2011, "Associative spatial networks in architectural design: Artificial cognition of space using neural networks with spectral graph theory" explores the organisation of an exhibition space on a two-dimensional level using *self-organising maps* and a *growing neural network*. The neural network is used to optimally distribute changing input representing exhibiting individuals within the architectural space. Experimental approaches to three-dimensional spatial organisation using Voronoi diagram algorithms are also included.

Newton 2019, "Deep generative learning for the generation and analysis of architectural plans with small datasets" investigates the use of GANs for the synthesis of architectural floor plans. Of note is the specific learning of certain architectural styles, such as that of the architect Le Corbusier, and the small size of the dataset, which is addressed by noise augmentation methods. However, this involves the generation of pixel-based plans, which have their own limitations.

Sebestyen and Tyc 2020, "Machine Learning Methods in Energy Simulations for Architects and Designers" presents a smaller-scale experiment using neural networks to predict sunlight hours and radiation levels based on the formal aspects of a designed building facade. The entire pipeline has been implemented using a visual scripting language, providing a simplified understanding of the framework for non-expert designers.

Eisenstadt, K.-D. Althoff, and Langenhan 2020, "Student Graduation Projects in the Context of Framework for AI-Based Support of Early Conceptual Phases in Architecture." delves through a compilation of work into another area of intelligent spatial configuration in the early stages of architectural design. It presents various extensions to the existing methodology, focusing on concepts such as *explicable AI*, *game theory*, *natural language processing and generation*. The common thread of these research efforts is the interaction with the designer and the usability of the developed model.

Alammar, Jabi, and Lannon 2021, "Predicting incident solar radiation on buildings envelope using machine learning" documents the training of two different machine learning methods, *Artificial Neural Networks (ANNs)* and decision trees, on a synthetic parametrically generated dataset for solar radiation of office buildings. The labelled data from the environmental simulation tool suite *Ladybug* served as the training base, allowing a comparison of the accuracy of the two trained machine learning models. The decision tree classification model showed higher accuracy, indicating the categorical nature of the data.

Liu 2021, "Topological Networks Using a Sequential Method Space Structure Simplifcation for Interactive Design" introduces a generative design method that abstracts complex topological data into spatial layouts, enabling the organisation and structuring of spatial networks according to user requirements. The research explores the application of Recurrent Neural Networks in the generative design process through experimentation.

Graph Machine Learning in Architecture

As, Pal, and Basu 2018, "Artificial intelligence in architecture: Generating conceptual design via deep learning" describes the process of encoding different buildings as knowledge graphs, and using graph machine learning methods to identify specific patterns and subgraphs. These learned structures and distinctive building blocks of the designs were then used as input to modified GANs. Using graph-based neural networks, the tool generated high-quality organisational graph objects based on desired criteria such as *livability* and *sleepability*.

Hu et al. 2020, "Graph2plan: Learning floorplan generation from layout graphs" analyses the generation of new floor plans with optimised programs based on user input about the desired layout graphs and building boundaries. *Graph2plan*, a trained graph neural network, has learned the adjacency information of the graphs in relation to their outlines. This tool provides suggestions for the internal functional structure during the design process.

Nauata, Chang, et al. 2020, "House-gan: Relational generative adversarial networks for graph-constrained house layout generation", "House-gan++: Generative adversarial layout refinement network towards intelligent computational agent for professional architects" document the use of actual architectural floor plans and their generated layout graphs describing room adjacency and function to create a GAN capable of generating plausible room configurations based on new bubble diagrams. The machine learning model, *houseGAN*, was later extended to handle non-rectangular room shapes, doors, entrances and functional graphs instead of adjacency graphs using a new vector-based dataset.

Sanchez-Lengeling et al. 2021, "A gentle introduction to graph neural networks" aims to illustrate the basic concepts and main application areas of deep learning methods with graphical data through visual examples. It presents essential concepts related to different graphical information and the different challenges related to graph components in machine learning. Finally, it visualises and explains terminology such as pooling, *message passing*, *edge and global representations*, sampling and *batching.*

Velikovi 2023, "Everything is Connected: Graph Neural Networks" emphasises the importance of graph structures in our natural environment in terms of physical and biological phenomena, as well as cultural and social elements such as language, text, information and social networks. The authors also explain the basic principles of Graph Neural Networks and their application in various classification and regression tasks.

Ali 2023, "Architectural Pipeline: An experiment into the role of topological graphs in the early stages of architectural design in the era of machine learning" describes and documents the creation of a pipeline for generating graph datasets, starting with pixel-based actual architectural floor plans. Using spatial recognition methods similar to Kalervo et al. 2019, "Cubicasa5k: A dataset and an improved multi-task model for floorplan image analysis", the plans were vectorised to extract topological graphs based on the *robust attributed adjacency graph extraction method* described in Chen and Stouffs 2022, "Robust Attributed Adjacency Graph Extraction Using Floor Plan Images".

Parametric Design and Algorithms

Coates et al. 2005, "Generating architectural spatial configurations. Two approaches using Voronoi tessellations and particle systems" examines the use of Voronoi diagrams and Delaunay triangulations in the context of automatic space partitioning. The two methods offer a number of advantages due to their flexibility and controllability by seed points, which are evaluated in terms of spatial configuration generation. Topological concepts such as *pathfinding* are also investigated.

Chatzikonstantinou 2014, "A 3-dimensional architectural layout generation procedure for optimization applications: DC-RVD" discusses and explains the *rectangular Voronoi subdivision* method in an architectural context. A disadvantage of regular Voronoi subdivision is that the resulting regions always have an irregular polygonal shape. With the developed algorithm, the resulting regions retain rectangular shapes, making it more suitable for traditional architectural applications, but it also has drawbacks such as gaps between individual spatial elements.

Aguiar, Cardoso, et al. 2017, "Algorithmic design and analysis fusing disciplines" presents an algorithm that allows the generation of a twin model in parallel with the conventional modelling process. The twin model has a simplified structure that is well suited for analytical purposes. The parametric algorithm is implemented in a visual scripting language that allows for continuous visualisation of the process.

Postle 2019, "On pattern languages, design patterns and evolution" introduces a software tool capable of automatically generating three-dimensional BIM models. The tool's functionality is based on the optimisation of patterns defined by Alexander and their solutions by evolutionary algorithms. It combines the language of architectural patterns with design patterns from computer science to generate optimised architectural objects.

Caetano, Luís Santos, and Leitao 2020, "Computational design in architecture: Defining parametric, generative, and algorithmic design" discusses the state of the art of computational design in the architectural context through a meta-analysis. The study explains different terminologies and their use and clarifies key concepts and synonyms such as parametric, generative, algorithmic, adaptive and evolutionary.

Automatic Floor Plan Generation

Lopes et al. 2010, "A constrained growth method for procedural floor plan generation" provides a method for automated space planning through a relatively simple algorithm that simulates and generates room allocations based on user-defined functional constraints. The growing structures resemble a combination of Voronoi diagram computation with a grid-based approach, but not exactly slicing or aggregation methods.

Shekhawat 2014, "Algorithm for constructing an optimally connected rectangular floor plan" presents a method for space allocation that is not significantly different from those mentioned above. However, it introduces an element called '*extra space*', which can take various forms such as corridors, terraces, balconies or storage space.

Calixto and Celani 2015, "A literature review for space planning optimization using an evolutionary algorithm approach: 1992-2014" categorises, analyses and compares space planning algorithms supported by evolutionary algorithms developed since the 1990s. It delves into different approaches to space partitioning, such as *half plans*, K-3D trees, shape grammars, *blocking*, assignment and slicing trees.

Guo and B. Li 2017, "Evolutionary approach for spatial architecture layout design enhanced by an agent-based topology finding system" studies the combination of multi-agent topology search systems and evolutionary solution methods. The algorithm belongs to the family of agent-based aggregation methods that use topological analysis metrics as evaluation strategies.

Nisztuk and P. B. Myszkowski 2019, "Hybrid evolutionary algorithm applied to automated floor plan generation", "Tool for evolutionary aided architectural design. Hybrid Evolutionary Algorithm applied to Multi-Objective Automated Floor Plan Generation" compares greedy-based algorithms and *hybrid evolutionary algorithms* in their ability to automate space allocation and planning, and presents an implementation of a resulting tool. User-defined optimisation parameters include room area, location and topological connectivity.

Carta 2021, "Self-Organizing Floor Plans" presents and compares a collection of ANNs and GANs applications in automated space planning methodologies. The meta-analysis raises questions about the acquisition and quality of architectural datasets, as these have a significant impact on the performance and influence of trained models.

Architectural Datasets

Fedorova et al. 2021, "Synthetic 3d data generation pipeline for geometric deep learning in architecture" develops a generation pipeline to create a synthetic three-dimensional architectural dataset. While datasets providing two-dimensional floor plan information are available in various forms, geometric deep learning tasks may require volumetric information. During the course of these experiments, individual architectural objects were augmented with additional information such as building components, material textures and building classes.

Eisenstadt, Arora, Ziegler, Bielski, Langenhan, K. Althoff, et al. 2021, "Comparative evaluation of tensor-based data representations for deep learning methods in architecture", "Exploring optimal ways to represent topological and spatial features of building designs in deep learning methods and applications for architecture" test, compare and evaluate different tensor-based encoding methods of architectural and topological information in conjunction with deep learning methods. The tensor formats tested include *multilayer maps*, *textual maps* and *one-hot-encoded maps*, with the latter method showing the highest accuracy with 98%.

A.2 Potential Application and Usage

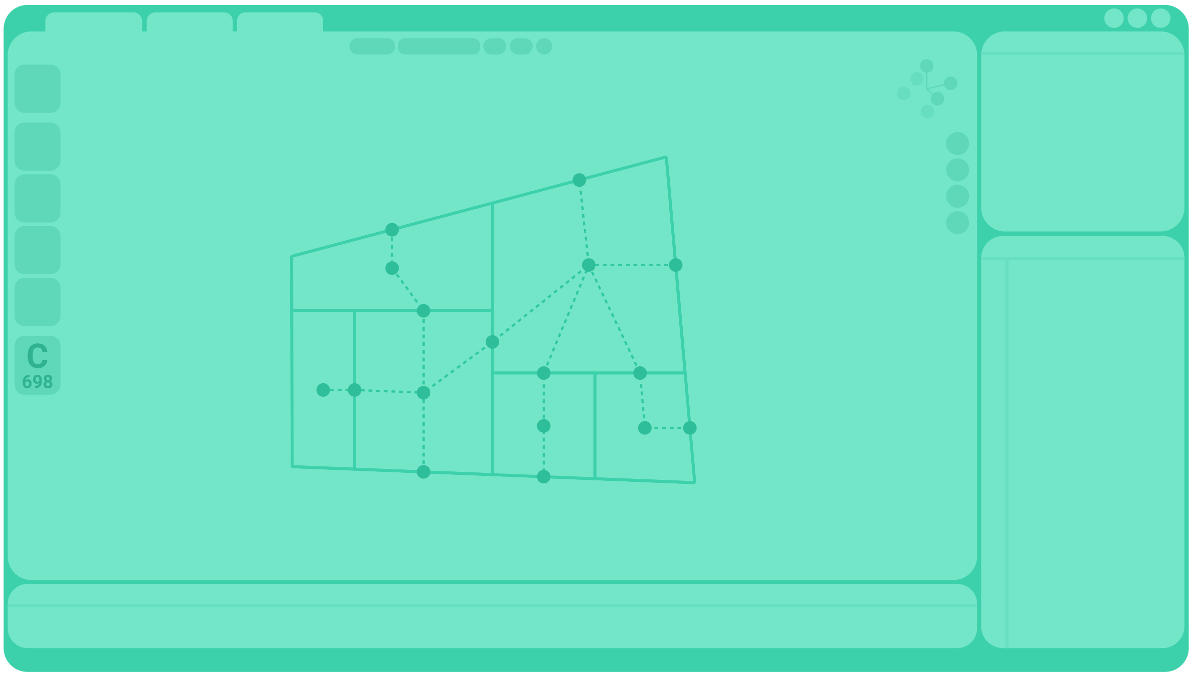
The scope of this work does not allow for a concrete application of the developed methods due to time constraints. Nevertheless, a presentation of a possible application workflow is helpful for a solid understanding of the usefulness of the models trained in the experimental phase. In the following sections, the possible software interaction is first presented and explained using visual examples. Then technical details of a possible implementation are given, and finally considerations for community-based improvement and learning methods are listed.

A.2.1 Workflow Concept

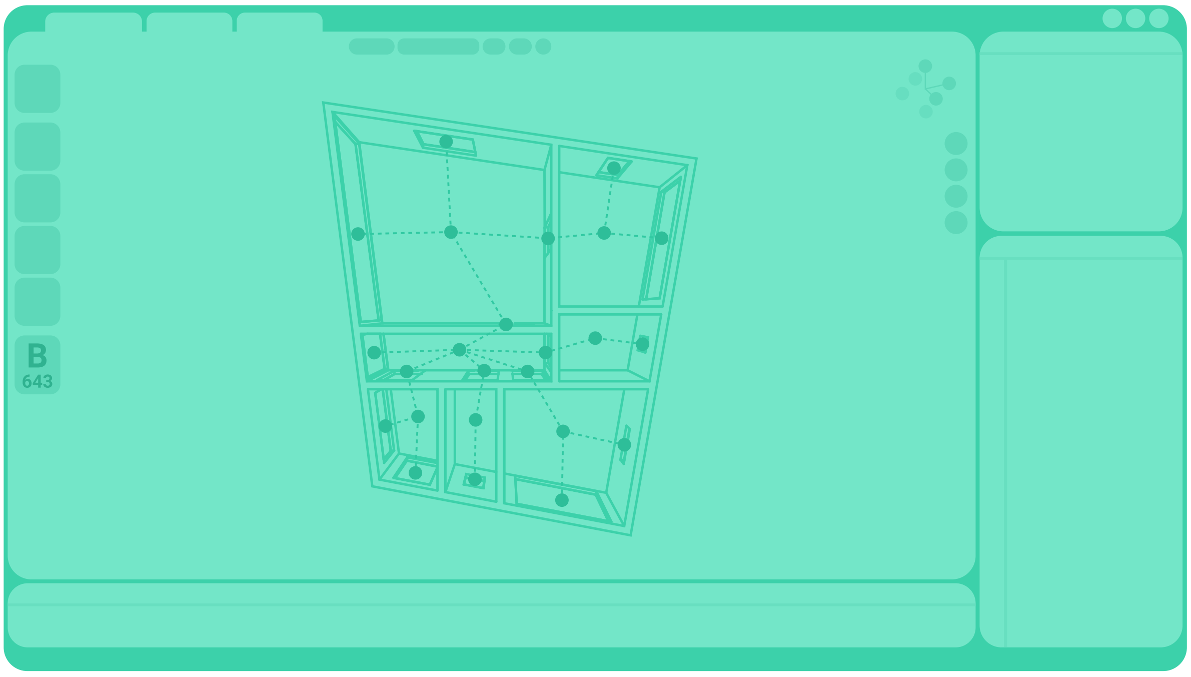
The implementation of the model-based feedback provider should be achieved through visual representations of the predicted values that are as intuitive as possible. Therefore, by using the API extension of the commercially available architecture software packages, an interaction with the user interface for the visualisation of the results can be ensured within the applied software. Thus, the application of the feedback method does not require any adaptation on the part of the user and the need for a large number of different software packages is avoided.

As shown in figure A.1, a small floating button is displayed in the center of the left half of the screen, continuously indicating the predicted energy consumption in MJ/m^2 and the corresponding energy class of the current design. In addition, an intuitive colour code can be used here, which could adopt a green-yellow-red gradient to effectively and conveniently convey good, mediocre or poor results.

The fact that the dataset, and thus the input to the trained model, is based on graphical information is extremely valuable at this point, as the level of detail and dimensionality of the developed design does not pose any problem or difference in the prediction process. The only requirement is to extract the topological information from the two dimensional plan or three dimensional building and compute the topological graph of the design based on this data. This can easily be done using the *TopologicPy* library, as long as a uniform aperture and space designation is given. In effect, this means that the prediction of simulation values can be performed in the background at all design stages without any effort on the part of the user (figure A.2), thus saving considerable cost and time in design practice.



(a) Feedback for 2D Floor Plan



(b) Feedback for 3D BIM Model

Figure A.1: Mock-up of Potential Implementation

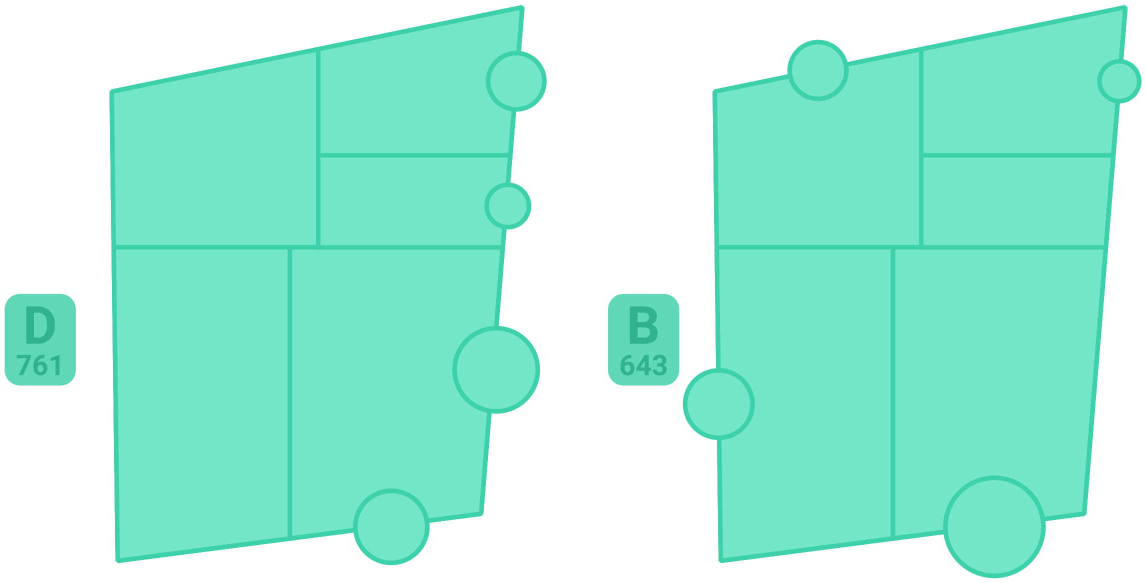


Figure A.2: Immediate Design Feedback

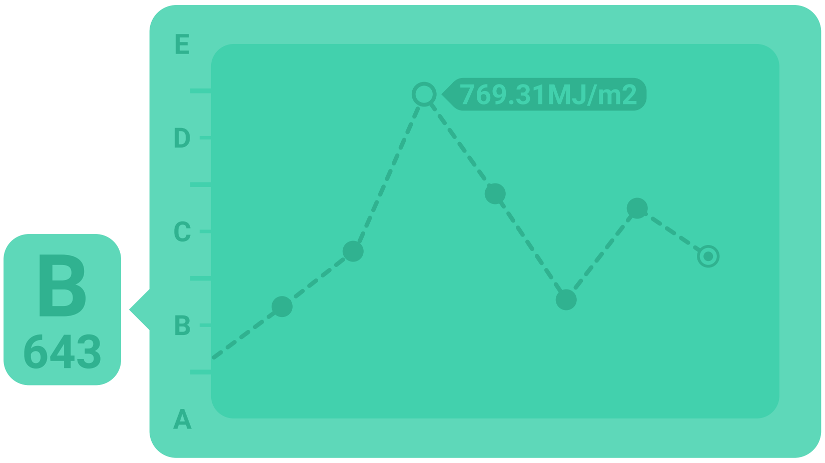


Figure A.3: Pop-Up Graph Showing Previous Design Variants and Performance

The key functionality of the developed method becomes apparent as soon as changes are made to the created design. By clicking on the box displaying the results, 'save points' can be created that store the performance of the design at that point in time. This allows detailed comparisons to be made between design variations, whether these are minor changes such as the size or position of a window, or major changes such as the orientation of the building or completely different building typologies. These comparisons are displayed in a pop-up window and visualised by graphs (figure A.3). By clicking on the individual datapoints, it is easy to go back to a previous design, which can be useful if the previous version was found to be more energy efficient than the current layout.

A.2.2 Technical Implementation

The presented functionality entails some prerequisites and requirements on a technical level. First of all, the interface between the user and the machine learning model has to be defined, since user-friendliness is an important criterion in the architectural application. On the one hand, one could develop an independent software that builds its interface through libraries such as GTK and uses open source kernels such as Open CASCADE as a geometry processor, or the development of an add-on that could be split into a software-independent processing part and a API for the dominant architecture tools, allowing easy integration into conventional design workflows.

As the proposed machine learning model is relatively small and therefore fast and resource efficient and could be distributed pre-trained, no special hardware requirements are necessary. Furthermore, an essential part of the technical implementation would be the development of a general design language for intuitive feedback visualisation that has uniform semantics across all software. Last but not least, the choice between a centralised server back-end or a decentralised local model interaction would have to be discussed.

A.2.3 Crowd Sourcing

During the research for the experiments and analyses presented in this paper, subjects around the degree of openness and democratisation of architecture were identified that offered a variety of interesting and pioneering ideas. The concept of open source architecture[1], both in construction and in the design process, is about making the acquired knowledge of individuals publicly available, in order to offer more inexperienced or non-specialist individuals the opportunity and access to the respective corpus of knowledge. The publication of the plans and rights of several projects by Pritzker prize winner Alejandro Aravena or the open source design initiative *OpenStructures*, which makes its designs available online free of rights, has sparked a general interest in the political-social movement in the AEC industry.

Another concept of knowledge sharing in the field of architecture is *urban mining*, in which urban areas and their buildings are understood as a kind of warehouse of building components, and thus a database of available constructive and decorative elements can be created and searched by anyone. This would make it theoretically possible to design an entire project based on a list of available components, thus addressing essential issues of today's society such as component recycling, carbon emissions, life cycle and energy consumption in the production of materials.

In the context of this master's thesis, the crowdsourcing of architectural designs, for example, could lead to a significant increase in the variety, complexity and reality of datasets such as the developed graph dataset, as personal designs are progressively fed in. It would be possible to increase the number and quality of datapoints in the dataset and to improve the labelling of the data through sensor measurements. This type of data collection through continuous sensor-based real-world measurement in an urban context is often referred to as *smart city* or *Internet of Things (IoT).*

A.3 Open Source Software and Knowledge

The term *open source software (OSS)* refers to the type of software whose source code is publicly available and published under certain permissive licences. Terms such as free software broaden the conceptual framework to include essential notions such as the freedom to use the program without restriction, to modify and redistribute the program as desired, and to publish improvements. In the early days of program development, these concepts were implicitly the norm and contributed significantly to software development. How-

[1]Postle 2019, "On pattern languages, design patterns and evolution".

ever, the diametrically opposed concept of proprietary software was introduced shortly afterwards for economic reasons and has since become the dominant approach to software development. As a result, various institutions such as the *Open Source Initiative (OSI)* or the *GNU Project* were founded to protect, regulate and centralise the values and standards of free software.

A.3.1 Benefits of Open Source

Open source and free software offer significant benefits that have widely changed research and software development and continue to drive technological progress[2]. They promote public accessibility and inclusion, enabling individuals around the world to benefit from advanced tools. Source code transparency enhances security and encourages community collaboration, leading to rapid innovation. The use of open source software reduces vendor lock-in and ensures longevity. In addition, open source file standards such as IFC and DXF in the AEC industry enable easy interoperability between different software packages and stakeholders. The use of these concepts is also consistent with the ethical principles of knowledge sharing and collective progress. In practice, however, large software vendors often resist the adoption of open source standards, as demonstrated by the public letter[3] addressed to *Autodesk* and signed by 324 influential AEC industry professionals.

A.3.2 AEC Software and Efforts

In addition to established OSS such as *NumPy*, *SciPy*, *PyTorch*, *Scikit-learn*, DGL and *Pandas*, a number of libraries and software have been tested and studied in the course of this work. In particular, simulation software corresponding to the main simulation topics (section 2.3) was collected. Due to their accessibility, Python-based or Python API were preferred. In the following subsections, the most important OSS related to the topics of this thesis can be found in a categorised list.

A.3.2.1 BIM Tools

- **BlenderBim** is a sophisticated Building Information Model (BIM) tool that integrates with *Blender* and uses the **ifcOpenShell** library to allow efficient visualisation and modification of Industry Foundation Classes (IFC) data, providing a complete BIM workflow.
- **FreeCAD** is a versatile parametric 3D modelling engine that provides a **BIM Workbench**. In combination with this add-on, FreeCAD becomes a proper BIM tool, providing architectural design, modelling and analysis capabilities.
- **Sverchok** is a visual scripting tool for parametric modelling within *Blender*. While not exclusively a AEC tool, it can be used to create rule-based geometries and automate specific tasks in architectural projects such as performance analysis.
- **TopologicPy** is a library for topological operations on point sets with integrated graph functionalities based on non-manifold elements. It can also be integrated into sverchok's visual programming engine and provides useful tools for graph and geometry analysis.

[2]Chaillou 2022, *Artificial Intelligence and Architecture: From Research to Practice.*

[3]Ali 2023, "Architectural Pipeline: An experiment into the role of topological graphs in the early stages of architectural design in the era of machine learning".

- **Homemaker Add-On** is a *Blender* add-on that simplifies architectural design by converting spatial configurations into IFC building models through a user-friendly approach that combines evolutionary algorithms with Christopher Alexander's *A pattern language: towns, buildings, construction*.
- **Open CASCADE** is a powerful 3D modelling and numerical simulation engine that provides essential components for CAD software.

A.3.2.2 Open Formats

- **IFC** is a file format used to store and exchange BIM data between different software applications. It facilitates interoperability throughout the project development cycle.
- **DXF** or Drawing Exchange Format is a file format developed by Autodesk and commonly used to exchange 2D and 3D drawings between different CAD software applications.
- **gbXML** or Green Building XML is a file format that represents the environmental characteristics of a building. Its primary use is in the simulation and analysis of the energy performance of building designs.

A.3.2.3 Structural, Thermal and CFD Analysis

- **ADA-Py** is a Python library used in *Finite Element Analysis (FEA)*. It provides structural analysis tools that allow advanced simulations of the mechanical performance of building designs.
- **CalculiX** is another FEA solver that provides various methods for structural and thermal analysis. It can calculate complex mechanical scenarios and simulate a variety of different material behaviours.
- **Code Aster** is a powerful structural and thermo-mechanical analysis software. Developed and maintained by EDF (Electricité de France), it has found wide application in the AEC industry.
- **CFAST** or Consolidated Fire and Smoke Transport is a simulation engine for fire propagation and smoke analysis in buildings. It can be used to assess fire safety and verify evacuation systems.
- **Elmer** is a multiphysics simulation software that provides different types of analysis such as thermal, structural or electromagnetic simulations.
- **FDS** or Fire Dynamics Simulator is a *Computational Fluid Dynamics* software used to analyse fire and smoke spread in buildings. It can help to understand fire behaviour and develop fire safety measures.
- **OpenFOAM** is another CFD software package that provides a wide range of numerical modelling capabilities for fluid flow and heat transfer analysis.
- **OpenSEES** or Open System for Earthquake Engineering Simulation offers a framework for simulating the seismic behaviour and resistance of structures. It is mainly used in earthquake analysis and research.

A.3.2.4 Environmental Analysis

- **Ladybug Tools** is a collection of environmental analysis plug-ins for various 3D modelling software. It allows architects and engineers to perform various environmental simulations such as daylighting, solar radiation and energy analysis by providing multiple access points.
- **OpenLCA** is a tool for *Life Cycle Assessment (LCA)* that allows users to analyse the environmental impact of construction processes, making it a suitable feedback tool for sustainable design and decision making.
- **CarboLifeCalc** is software that focuses on calculating the carbon footprint of building materials and projects. It helps to quantify the environmental impact of construction practices and enables environmentally conscious design choices.
- **Radiance** is a suite of lighting simulation and ray tracing tools. It is most commonly used for daylight analysis and shadow simulation in architectural space.
- **Vi-Suite** is an add-on for *Blender*, originally designed for contextual and performative building analysis. Over time it has expanded to include dynamic functionality such as parametric lighting, shadows and building energy analysis through the integration of Radiance, EnegyPlus and OpenFOAM.
- **CEA** or Comprehensive Environmental Assessment is a software tool used to assess and subsequently optimise the environmental performance of buildings. It considers multivariate factors such as energy use, carbon emissions and indoor environmental quality.
- **EnergyPlus** is a widely used energy simulation software from the *National Renewable Energy Laboratory (NREL)* that models the energy consumption of buildings in combination with HVAC systems. It enables the assessment of building energy performance and provides metrics that can be used to develop energy efficient designs.
- **OpenStudio** is a platform used to perform energy modelling and simulation for buildings. It provides an interface to EnergyPlus and additional tools for advanced building performance analysis.
- **SAM** or System Advisor Model is another toolkit developed by the NREL for the design of systems integrating renewable energy concepts. Its main application is to assess the economic and environmental impact of renewable energy sources such as wind, solar and others.

A.3.2.5 Traffic and Pedestrian Analysis

- **GAMA** is a platform for modelling and simulation of complex systems such as traffic and pedestrian dynamics. It can be used to study the behaviour of agents in different environments, such as urban areas or public buildings.
- **SUMO** or Simulation of Urban Mobility is a traffic simulation software used to model individual vehicles and pedestrians to analyse traffic flow, congestion and transport.
- **JuPedSim** is another pedestrian simulation software designed to study crowd dynamics and pedestrian behaviour. It provides tools to help design public spaces and optimise pedestrian flow within building complexes.

A.3.2.6 Acoustic Simulation

- **I-Simpa** is an acoustic simulation software capable of analysing and calculating sound propagation in different media and environments. Its capabilities enable architects to evaluate and optimise the acoustic performance of buildings and spaces.
- **NoiseModelling** is a tool that allows users to create and analyse noise maps and models. It is mainly used to reduce noise pollution in urban areas.

A.3.2.7 Urban Analysis

- **UrbanSim** is another open source software with a focus on urban simulation and land use modelling. It can help planners understand urban growth patterns, infrastructure and the impact of urban regulations.
- **URBANopt** is an urban energy modelling platform that combines energy and urban simulation to analyse the energy performance of buildings and their interactions with the surrounding environment.
- **QGIS** or Quantum GIS is a *Geographic Information System (GIS)* software that allows users to analyse, visualise and interpret a variety of different spatial data, providing an important tool for urban planning.
- **UrbanPy** is a Python library that facilitates urban data analysis and simulation by providing tools to work with geospatial data and perform various urban analyses and simulations automatically.
- **OSMnx** is a Python library used to retrieve and analyse OpenStreetMap data. It is capable of generating urban road networks, visualising urban patterns and performing network-based analysis.
- **Mega-Polis** is a data-focused urban add-on for *Blender* that provides a range of capabilities for collecting, analysing, generating and visualising urban data. It uses Python libraries such as GeoPandas, NetworkX, OpenCV and Shapely to enhance its functionality.

A.3.2.8 Resources and Datasets

- **OSArch Community** is a platform for sharing and discussing open source architectural design and BIM tools. It provides resources, forums and channels for the exchange of information in the AEC industry.
- **Brick** is a data model for representing buildings and their systems. It helps standardise and access building data for many purposes, such as energy modelling or performance analysis.
- **MaterialsDB** is a database of building materials and their properties. It provides information for architectural and engineering simulations, allowing the selection of appropriate materials based on selected performance criteria.
- **ÖkobauDat** is a German database that catalogues ecological building materials and products, making it an important resource for sustainable and environmentally friendly building practices.
- **BDG2** or Building Data Genome 2 dataset includes 3 053 energy meters in 1 636 buildings, providing two years of hourly measurements. The data includes

electricity, heating, cooling water, steam and irrigation meters.

- **CubiCasa5k** is a dataset of 5 000 floor plans of residential buildings. It could be used as a basis for various machine learning tasks in the field of architectural automation.
- **RPLAN** is an extensive dataset of annotated and cleaned floor plans extracted from actual residential building layouts.

A.3.3 Application in this Thesis

This work is largely based on open source knowledge and software and would certainly not have been possible without it. Furthermore, the data, information, models and results of this work will be made publicly available and openly accessible, downloadable and modifiable on online portals appropriate to the type of data (section 6.1). During the development of this work, a partial ambition was to use exclusively open source programs to conduct a feasibility study on the state of the art of OSS in the AEC industry. Despite the primarily academic nature of the work, a complete OSS workflow was developed without much difficulty, thus proving the feasibility of such an approach.

In addition to basic software such as Python, Linux, *Blender* and *Pytorch*, programs such as *Sverchok*, *TopologicPy* and DGL proved particularly helpful in their application. The most important aspect of their implementation, however, was the exchange with the respective developers about the functionality, use and operation of the individual libraries. For example, contact with Professor Wassim Jabi[4], the principal developer of the *TopologicPy* library, proved invaluable. The opportunity to study the source code allowed a deep understanding of the functionality and, after repeated exchanges, it was possible to develop new functions and integrate them into the software source code as so-called pull requests. In this way, functions and methods for dataset balancing, energy simulation optimisations and the implementation of graph regression training and evaluation were contributed.

This shows that an equally important part of the OSS community is the sharing and publication of knowledge and results, in addition to the disclosure of source code. In this context, media such as the *OSArch Community*[5] and its associated wiki have proven to be extremely helpful in process development and in the search for inspiration. Publicly available project and code repositories such as *GitHub*, *GitLab* and *kaggle* also provided essential resources, mostly through detailed introductory texts and easy contact with developers and interested contributors. Last but not least, the publication of scientific papers can also be seen as a source of academic collaboration and contribution.

[4]Jabi 2013, *Parametric design for architecture.*
[5]Moult 2020, *OSArch Community.*

Appendix B

Raw Data

The experimental part of this work is a mostly data-based approach, especially the machine learning part. Thus, large amounts of data have been stored, reviewed and modified during the process in human-readable tabular, graphical or geometric visual forms. However, since the purpose is to demonstrate the process rather than to guide the reproduction of the results, the excerpts of this data are found in the appendix rather than in the body of this document. Nevertheless, the form of this information played an essential role in the parametric generation and training process. The samples of data presented in the following paragraphs have been applied in different areas and are broadly divided into geometric, graphical and informative data. The geometric part mainly refers to information that was needed, generated or defined as an initial rule during the geometric generation step. The graphical data represents the storage of the knowledge graphs in machine readable form as input to the deep learning section, and the informational data deals with the storage in JSON format, node label and quantile computation, and graph label computation.

B.1 Geometry Data

B.1.1 Input Parameter

One of the initial requirements of the geometric generation pipeline to ensure architectural coherence was the definition of maximum and minimum room sizes according to the respective room types. Furthermore, this size definition also determined which individual rooms the different apartments contained in their program, according to their number of rooms. For example, a three-room apartment has only a living room, a bathroom and a bedroom, whereas a ten-room apartment has three utility rooms and four bedrooms, as well as a toilet, a bathroom and a living room.

The tables B.1 and B.2 show the minimum and maximum room sizes respectively. The amount of rooms is shown per column and the room types are represented by individual rows. The last line shows the value of the sum of all living areas of the respective apartment type, thus describing the maximum and minimum total apartment sizes to be generated per amount of rooms.

Room amount	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Living	28	26	28	29	28	30	33	36
Bedroom	8	9	10	13	12	10	12	14
Bedroom					9	8	9	11
Bedroom						6	7	8
Bedroom								6
Bathroom	3	4	4	4	4	4	6	7
Toilet			1	1	1	1	1	2
Utility		3	5	3	3	1	3	4
Utility				12	8	8	12	12
Utility							4	6
total	39	42	48	62	65	68	87	106

Table B.1: Minimum Room Sizes per Room Amount

Room amount	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Living	32	30	32	33	32	34	37	40
Bedroom	12	13	14	17	16	14	16	18
Bedroom					13	12	13	15
Bedroom						10	11	12
Bedroom								10
Bathroom	7	8	8	8	8	8	10	11
Toilet			5	5	5	5	5	6
Utility		7	9	7	7	5	7	8
Utility				16	12	12	16	16
Utility							8	10
total	51	58	68	86	93	100	123	146

Table B.2: Maximum Room Sizes per Room Amount

B.1.2 Geometry Storage

Geometric bodies can be stored in a machine-interpretable form in a number of ways. In this work, the use of the geometry kernel *OpenCASCADE*[1] led to the decision in favour of Boundary Representation, as this allowed seamless integration with *TopologicPy*. The Brep files are hierarchically structured and are therefore similar to the functionality of the Python library used, since elements of higher order such as shells, solids and compounds are composed and described by smaller elements such as vertices, edges and faces.

The text-based file format (listing B.1) defines and stores different elements by Cartesian coordinates, which are stored through three-dimensional matrices or single coordinates with respect to the origin. The bodies described in this way are first specified by their position in three-dimensional space and then defined by three-dimensional points according to their respective geometry. The geometric section of the Brep file is therefore made up of the following elements: *2D curves*, *3D curves*, *3D polygons*, *polygons on triangulations*, *surfaces* and *triangulations*. In addition to its seamless integration into the geometry generation pipeline, this file format is also suitable because it provides the geometry description privileged by the IFC format.

```
Curve2ds 4
1 0.4 0.88 -1 0
1 0.4 -0.88 0 1
1 -0.4 0.88 0 -1
1 -0.4 -0.88 1 0

Curves 4
1 4.7 1 1.6 1.1 0 -1
1 4.7 2.8 1.6 0 -1 0
1 4.7 1 0.8 0 1 0
1 4.7 2.8 0.8 -1.1 0 1

Surfaces 1
1 4.7 1.9 1.25 1 0 0 -0 0 1 0 -1 0

    TShapes 10
    Ve
    1e-07
    4.7 1 1.6
    0 0

    ...
```

Listing B.1: Brep File Structure

B.1.3 Energy Simulation Parameter

An essential part of the dataset generation was the implementation of energy performance simulations of each BIM model. As described in section 4.4.2 and figure 4.24, a successful energy simulation using the *EnergyPlus* and *Openstudio* software required the definition

[1]Slyadnev, Malyshev, and Turlapov 2017, "CAD model inspection utility and prototyping framework based on OpenCascade".

Location	**Windows**	**Int. Walls**	**Ext. Walls**	**Type**	**Insulation**
Berlin	Double Glazed	Gypsum	Concrete	Midrise Apt.	12cm

Table B.3: Energy Simulation Values

Graph ID	**Energy Class**	**Energy Consumption**	**Node Amount**
8	3	758.59	14

Table B.4: Graph Data

of several parameters. The main initial values chosen for the experiment are shown in the table B.3.

The definition of the location (in this case Berlin) is essential, as the latitude and the associated climate difference have a significant influence on the results of the energy balance. In order to generalise the climate, a number of additional datapoints would have to be added to the dataset, differing only in the location of the EnergyPlus Weather Format (EPW) file. In addition, the B.3 table shows that the materials of the individual apartment walls and ceilings, including the insulation, have a generic composition with concrete or gypsum depending on their function. Again, greater versatility in the training process is possible by extending the dataset and graph labels.

B.2 Graph data

The human-readable graph objects produced by DGL are stored in a specific text-based format that allows the generated graphs to be reviewed, modified, or integrated into various data science processes. However, to save space, large graph databases are typically stored as binary files using the standard Python library *Pickle*.

The resulting files consist of three separate text files corresponding to the graph components: node, edge and graph. The information shown in the table B.4 is an excerpt from the graph-wide text file, which first defines in tabular form a graph identification number per object in the dataset. For performance reasons, the number of nodes contained in the graph is also stored here. This information is already sufficient to describe a complete graph. However, to create a meaningful dataset it is also necessary to store the nodes and edges in their individual text files and to add one or more labels to the graph object. These labels are, in this case, the energy class and the energy consumption value, which allow the machine learning model to learn the relationships between the graph structure and the energy performance.

According to the number of nodes, the individual vertices are defined in the node file (table B.5) for each graph of the dataset and assigned to the individual graphs with the graph identification number. The nodes are also given an individual identification number, which allows each node of the whole dataset to be referenced precisely and is essential for defining the edges of the graphs. The most important part of the node file, however, is the node label column, where each node can be assigned an arbitrary label, which, together with the graph labels, will be used as input for the machine learning process. In the procedure demonstrated in this thesis, the node labels consist of encoded information about the element type, its size and its orientation. More detailed information about the node label calculation can be found in section B.3 and table B.9.

Now that the general graphical data and node specific information has been defined,

Graph ID	8	8	8	8	8	8	8	8	8	8	8	8	8	8
Node ID	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Label	89	60	65	80	2	78	72	69	83	26	78	10	66	33

Table B.5: Node Data

ID	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
SRC	0	0	0	1	1	1	1	1	1	3	3	5	5	7	7	7	8	8	10	10	12
DST	3	2	1	7	3	5	8	6	2	8	4	7	6	8	10	12	9	10	11	12	13

Table B.6: Edge Data

only the edges between the nodes need to be described and stored. This is likewise achieved by a tabular text file (table B.6) that summarises the total number of edges in the dataset. Each individual edge is described by a column of the table and is first assigned to the respective graphs by the graph identification number, as in the case of the nodes. Furthermore, each node is defined by its source (SRC) and destination (DST) node points, which are referenced by the node identification numbers. The edge file thus represents a classical edge list notation. In the case of this work, we are dealing with undirected graphs, which means that an edge with (u v) also appears as (v u) in the edge list (table B.7).

B.3 Information Data

B.3.1 BIM Model Storage

The values generated during the information annotation process had to be correlated using specific file formats in order to avoid an unnecessarily complex data generation process. The key-value pair based text format JSON was chosen as the main format for data storage. This choice was made not least because the applied geometry and analysis library *TopologicPy* is equally based on JSON files for importing and exporting the individual topological geometries.

This allows all the information about the individual BIM elements to be stored in a single file for each apartment. In addition to the JSON format, the IFC file type was also considered, but in the end the simplicity of the JSON structure was preferred. The B.2 listing shows a simplified example file of annotated apartment geometry, consisting of information about the geometry as Brep strings and values at different element levels.

First, the Brep of the apartment and the individual rooms are listed with their respective types, surface sizes and labels. Then the apartment level information is listed, such as number of rooms, room types and total area. Also stored at this level are any values that provide information about the energy balance of the architectural object, such as heating and cooling energy, general energy consumption, energy class, window-wall ratio and window opening per orientation. Although in the end only the site energy consumption

ID	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
SRC	3	2	1	7	3	5	8	6	2	8	4	7	6	8	10	12	9	10	11	12	13
DST	0	0	0	1	1	1	1	1	1	3	3	5	5	7	7	7	8	8	10	10	12

Table B.7: Edge Data Undirected

per square meter and the energy class were used for training, it was important to test the information storage capabilities of the file format for experimental purposes. Finally, under the key: 'faceApertures', the windows of the model are noted with their respective geometry as brep strings, area, orientation and label.

```
[
    {
        'geometry': 'brep',
        'cellDictionaries': [
            {
                'dictionary': {
                    'area': 3.11,
                    'element': 'room',
                    'label': 84,
                    'type': 'bathroom'
                }
            },
            {
                'dictionary': {
                    'area': 8.1,
                    'element': 'room',
                    'label': 77,
                    'type': 'bedroom'
                }
            },
            {
                'dictionary': {
                    'area': 28.54,
                    'element': 'room',
                    'label': 56,
                    'type': 'livingroom'
                }
            }
        ],
        'dictionary': {
            'element': 'apartment',
            'room amount': 3,
            'area': 39.75,
            'room types': [
                'bathroom',
                'bedroom',
                'livingroom'
            ],
            'site energy cooling GJ': 0.61,
            'site energy cooling MJ/m2': 19.0,
            'site energy heating GJ': 18.18,
            'site energy heating MJ/m2': 567.51,
            'total exterior wall amount': 4,
            'total site energy consumption GJ': 24.73,
            'total site energy consumption MJ/m2': 771.98,
            'total source energy consumption GJ': 85.16,
        'total source energy consumption MJ/m2': 2658.05,
        'energy class': 3,
        'total wall area m2': 61.13,
        'total window—wall ratio %': 8.52,
        'total window amount': 3,
        'total window opening area m2': 5.21,
        'wall area east m2': 15.28,
        'wall area north m2': 15.28,
        'wall area south m2': 15.28,
        'wall area west m2': 15.28,
        'window/wall ratio east %': 0.0,
        'window/wall ratio north %': 34.09,
        'window/wall ratio south %': 0.0,
        'window/wall ratio west %': 0.0,
        'window opening area east m2': 0.0,
        'window opening area north m2': 5.21,
        'window opening area south m2': 0.0,
        'window opening area west m2': 0.0
    },
    'faceApertures': [
        {
            'geometry': 'brep',
            'dictionary': {
                'area': 1.35,
                'element': 'window',
                'label': 24,
                'orientation': 'N'
            }
        },
        {
            'geometry': 'brep',
            'dictionary': {
                'area': 3.35,
                'element': 'window',
                'label': 32,
                'orientation': 'N'
            }
        },
        {
            'geometry': 'brep',
            'dictionary': {
                'area': 0.51,
                'element': 'window',
                'label': 0,
                'orientation': 'N'
            }
        }
    ]
}
```

	XXS | XS	**XS | S**	**S | M**	**M | L**	**L | XL**	**XL | XXL**
Window	0.503	0.823	1.304	2.214	3.538	5.327
Livingroom	27.397	29.685	31.376	33.409	35.349	37.953
Bedroom	8.137	9.255	10.261	11.322	12.491	14.218
Toilet	2.419	3.288	3.853	4.303	4.814	5.463
Bathroom	4.759	5.863	6.662	7.367	8.121	9.147
Utility	5.171	6.186	7.007	8.206	10.361	13.114

Table B.8: Room Size Quantile Calculation

```
]
```

Listing B.2: JSON File Structure

B.3.2 Node Label Calculation

In order to calculate the node labels, the element types living room, bedroom, toilet, bathroom and window must be divided into relative size classes. These seven classes range from XXS to XXL and describe the size of the item in relation to the total set of sizes. A division into so-called quantiles of the distribution was therefore necessary, but could only be carried out after the entirety of the BIM models had been generated. The quantile values calculated in this way can be found in the table B.8 and describe the threshold value that separates two classes.

The final step in calculating the labels shown in the listing B.2 was to encode the categories of each element into integer labels ranging from 1 to 91 (figure B.9). This is a simple type of encoding where the three categories: type, size and, in the case of windows, orientation, are listed and then simply labelled by their index in the list. This method allows the individual elements to be categorised and distinguished from each other. Other types of encoding were also considered, such as feature-based encoding, where the described procedure is applied to each feature individually, thus generating two labels for rooms and three for windows. Nevertheless, the first described method was chosen for this experiment. For the node labels, however, it is important to note that the integer labels are *one-hot-encoded* anew when the DGL dataset creation methods are applied.

B.3.3 Energy Class Definition

Since the classification task involves the prediction of distinct classes, the site energy consumption values first had to be divided into energy classes. Similarly to the method described in table B.8, all the energy simulations were first performed in order to divide the general distribution (figure 4.29) of the energy values into quantiles according to the desired number of classes. Table B.10 shows the quantile thresholds generated in this way for five energy classes. It is important to note that these classes, unlike the node labels, are hierarchically related. This means that an apartment with an energy class of zero will perform better in terms of energy efficiency than an apartment with a class of one, and so on.

B.3.4 Graphical to Tabular Data Conversion

In order to perform the comparisons described in section 5.3.3 with different machine learning models, the initially purely graphical information had to be converted into con-

Label	**Type**	**Size**	**Orientation**
1	window	XXS	N
2	window	XXS	NE
3	window	XXS	E
4	window	XXS	SE
5	window	XXS	S
6	window	XXS	SW
7	window	XXS	W
8	window	XXS	NW
9	window	XS	N
..	......	...	...
57	livingroom	XS	
61	livingroom	XL	
64	utility	XS	
68	utility	XL	
71	toilet	XS	
75	toilet	XL	
78	bedroom	XS	
82	bedroom	XL	
85	bathroom	XS	
89	bathroom	XL	
..	....	...	

Table B.9: Node Label Calculation

	0 | 1	**1 | 2**	**2 | 3**	**3 | 4**
Energy Class	617.19	659.78	702.98	777.72

Table B.10: Energy Class Quantile Calculation

	18	**45**	**56**	**76**	**92**	**109**	**125**	**141**
Rooms	8	3	9	8	10	7	5	4
Windows	8	8	9	8	10	11	9	9
Surface	84.95	32.04	100.18	92.15	131.11	79.87	59.48	49.14
Consumption	704.06	878.09	667.81	666.68	634.52	791.09	818.39	845.12
Energy Class	3	4	2	2	1	4	4	4
Window m^2 N	9.06	6.57	0	0	0	4.46	10.36	7.46
Window m^2 E	8.34	1.31	6.62	4.86	2.48	11.27	0	9.3
Window m^2 S	0	0.65	6.13	7.35	14.57	10.09	11.3	1.56
Window m^2 W	4.83	2.54	11.17	8.73	10.69	5.89	1.74	1.5

Table B.11: Graph Derived Data

18 :	86, 78, 59, 80, 79, 63, 71, 67, 47, 48, 50, 2, 16, 2, 24, 26
45 :	84, 77, 56, 14, 32, 6, 32, 12, 30, 0, 26
56 :	76, 60, 80, 68, 63, 78, 66, 90, 77, 46, 54, 2, 2, 42, 20, 6, 44, 26
76 :	75, 77, 89, 60, 64, 67, 78, 82, 18, 2, 52, 34, 20, 4, 54, 38
92 :	76, 78, 79, 62, 81, 82, 68, 68, 88, 65, 52, 12, 44, 14, 46, 10, 36, 10, 46, 18
109 :	68, 83, 81, 86, 58, 63, 71, 18, 52, 20, 34, 4, 47, 15, 40, 34, 42, 8
125 :	75, 57, 66, 87, 80, 19, 51, 22, 27, 0, 40, 19, 14, 48
141 :	64, 86, 79, 57, 8, 42, 4, 42, 4, 24, 20, 30, 40

Table B.12: Graph Derived Data Labels

ventional tabular form, since DGCNN is the only one of the models that can be trained on data presented in the form of graphs. To achieve this conversion, first had to be considered what information was contained in the graphical dataset (section B.2). So the categories: Space and number of windows, window area by cardinal direction and previous node labels were identified. As a target variable, according to the prediction task, the energy class or the energy consumption value was used in the same way as in the graphical dataset. Table B.11 shows an extract of this tabular dataset for the datapoints: 18, 45, 56, 76, 92, 109, 125 and 141. For spatial reasons, the nodelabel list of each datapoint is shown in the table B.12, but in reality these datapoints are stored in a single database. The resulting table, not least because of its tabular form, contains different information to the graphical dataset, which should be taken into account when evaluating the performance comparison of the different models.

Faculty of Architecture La Cambre Horta
Université Libre de Bruxelles
2023