Optimization of cost and discrepancy for architectural building redrawing using prefabricated wall elements

Tesi di Laurea Magistrale in
Computer Science Engineering - Ingegneria Informatica

Author: **Tiberio Galbiati**

Student ID: 952607
Advisor: Prof. Federico Malucelli
Co-advisors: Prof. Pietro Luigi Belotti
Academic Year: 2021-22

Abstract

Off-site and prefabricated construction is a promising alternative to traditional on-site building. It has benefits in cost-effectiveness, environmental impact, and development speed. In order to take advantage of off-site construction, architects need to consider some geometric constraints in their drawings, given by the prefabrication manufacturing processes. However, most building projects are not designed with a prefabrication mind-set. Therefore, manufacturers of prefabricated elements need to redraw the models to fit them with their elements. This time-consuming process is currently performed manually, which does not guarantee optimal results. This thesis proposes a method to automate the redrawing of architectures. Given a 3D building design and a set of prefabricated elements, it outputs a minimum discrepancy new version assembled with the set's parts. A Mixed Integer Programming model is developed and tested to address this problem. The optimization's target is to minimize the modifications from the original design and the cost, with constraints on obtaining a closed polygonal perimeter. The method is tested with different building geometries and gives fast results. It is the first step toward a fully automated software for prefabrication building design redrawing.

Keywords: optimization, operation research, MIP, IFC, prefabrication, off-site construction, design for manufacturing and assembly (DfMA)

Abstract in lingua italiana

La costruzione off-site e prefabbricata è una alternativa promettente alla costruzione tradizionale in cantiere. Presenta vantaggi in termini di economicità, impatto ambientale e velocità di realizzazione. Per trarre vantaggio dalla costruzione off-site, è necessario che gli architetti considerino alcuni vincoli geometrici nei loro disegni, definiti dai processi produttivi di prefabbricazione. Tuttavia, la maggior parte dei progetti edilizi non viene progettata con la consapevolezza della prefabbricazione. Pertanto, i produttori di elementi prefabbricati sono costretti a ridisegnare i modelli per adattarli ai loro componenti. Si tratta di un processo laborioso, attualmente eseguito manualmente, che oltremodo non garantisce risultati ottimali. Questa tesi propone un metodo per automatizzare il ridisegno delle architetture, per la sua soluzione è stato ideato, sviluppato e testato un modello di Programmazione Lineare Misto-Intera. Dato un progetto di edificio in 3D e una lista di elementi prefabbricati, il metodo produce una nuova versione del disegno, assemblato con elementi della lista e con la minima discrepanza rispetto all'originale. L'obiettivo dell'ottimizzazione è minimizzare le modifiche rispetto al modello originale ed il costo finale, con il vincolo di ottenere un perimetro poligonale chiuso. Il metodo è stato testato con diverse geometrie di edifici e ha dato risultati in tempi rapidi. È il primo passo verso un software completamente automatizzato per il ridisegno di progetti di edifici per la prefabbricazione.

Parole chiave: ottimizzazione, ricerca operativa, MIP, IFC, prefabbricazione, off-site

Contents

Introduction

This thesis intersects three disciplines: Operation Research, Computational Geometry, and Construction Engineering. It uses methods and tools of Computer Science, Optimization, and Computational Geometry to help manufacturers deal with architecture projects for off-site construction. In off-site construction, structural elements, such as wall panels, are made in an off-site factory and assembled later on the building site. The goal of this thesis is to automate the design for off-site manufacturing.

The issue. Building project contractors should decide to use off-site construction ahead to adapt the design to consider the constraints and information given by the manufacturers. However, this rarely happens. The manufacturers of prefabricated elements instead receive building drawings that have not been designed considering their production constraints. Thus, they must redraw the models to fit them with their construction system. A construction system consists of different elements with pre-defined geometry and technical properties produced in an off-site factory and later assembled on-site. Redrawing building projects is a task currently performed manually: it is time-consuming as there are no automation tools that can be adopted for every project. Moreover, manual redrawing does not guarantee the optimal results in terms of the final design costs and, simultaneously, to minimize the discrepancy with the original model. Some final customers of a building are willing to accept slight modifications in the structure's geometry provided that the project's final cost and development time using prefabrication are lower than in traditional construction methods. Hence, optimizing the redrawing task is crucial to prove the advantages of prefabrication to customers.

At the current state of the art, no completely automatic solution can transform a 3D building project into another one made of discrete prefab elements. This is a barrier to the adoption of prefabrication. That is why the industry needs to implement automation methods for the redrawing process to make it more manageable, precise and fast.

What is presented. The approaches presented in this thesis prove that, given a 3D building model, it is possible to generate a new geometry made out of parts specified in an element library given by the manufacturer. The output will either be optimized for having the minimum difference from the original one in geometric dimensions, the minimum cost of manufacturing using prefabricated elements, or a combination of the previous two. This thesis demonstrates that it is possible to build almost any geometry floor plan using standardized elements with slight changes in the original design.

Why is it useful. In the first instance, this thesis approach optimizes for a combination of minimum cost and output discrepancy. If the method can satisfy both requirements, it means that a wall manufacturer could reduce the costs of building products and guarantee limited changes over the original design geometry not intended for prefabrication. In contrast, the manual process currently adopted does not guarantee the fulfilment of those goals.

Secondly, the method proposed will decrease the time spent on the task of redrawing. Interviews with prefab manufacturers in the industry revealed that this process takes an average of one human working day using the available tools. This thesis approach can lower the time to less than a minute. Reducing time is relevant, especially for companies that perform the redrawing process to provide a cost estimate for a project bidding. Providing a fast and accurate estimate of the project's cost can help reduce the resources needed for the bidding calculations, increasing the ability to provide price quotes and resulting in more orders.

Finally, the impact of the approach in this thesis can be even more profound and substantial: this work wants to be one step forward in increasing the adoption of off-site construction. Automated tools for redrawing can make off-site construction (OSC) more competitive than the traditional methods. Allowing the adaptation of current designs by architects will help in shifting the paradigm in the construction industry. The reason is that it will be possible to adopt standardized industrialized production of buildings without starting the design phase over again.

Lastly, buyers of buildings are increasingly considering waste reduction and environmental impact. Optimizing a standardized production can fulfil those target goals, providing better quality and value for money without sacrificing customization entirely. Customers are expected to accept standardized products and processes as long as the final result costs less money and is faster to produce [1].

How this thesis is structured. This thesis is the report of a process that involved the creation of an original optimization model and a method to extract data from the 3D drawings generated by the Computer-Aided Design (CAD) software used by architects. The first part is the core of the thesis; it uses methods of Operation Research to create an optimization model. It consists of a Mixed-Integer Programming (MIP) model with a twofold objective: minimum discrepancy redrawing of a 2D floor plan geometry using a list of defined elements and minimum manufacturing cost.

The second part presents an approach for automatic quantity and geometry extraction from Industry Foundation Classes (IFC) files; this part is needed as a step to prepare the necessary data for the optimization. IFC files contain no explicit information about the quantities and the geometry structure of elements needed to perform the optimization.

Chapter 1 briefly introduces Architecture, Engineering and Construction (AEC) industry to contextualise this thesis's work. Definitions of off-site construction types, elements, and methodologies used in industry, such as Building Information Modeling (BIM) and Industry Foundation Classes (IFC), are provided. The market of prefabrication and its adoption are discussed to justify the need for the thesis.

Chapter 2 defines the specific problem addressed in this thesis and the approach taken to solve it. The scope is defined in terms of the input available and the desired output. This chapter presents other relevant works in the literature and discusses them, comparing their objectives and content with this thesis.

Chapter 3 formally and mathematically describes the optimization problem to be solved. A Mixed-Integer Programming (MIP) model is presented with input specifications, variables, constraints, and objective function(s).

Chapter 4 presents the original algorithms developed to extract the relevant geometric data from IFC files to perform the optimization part. It is a method to decompose a complex building architecture model into a minimal, yet exhaustive, data structure to represent its geometry.

Chapter 5 firstly discusses this work's implementation details, such as the software, tools and libraries used. Secondly are presented a batch of tests of the optimization part to evaluate the model on different input instances. Finally, it presents a case study where the approaches of this thesis are tested from the geometric extraction to the optimization part, comparing them with the industry standards.

Chapter 6 derives the conclusions of this thesis and introduces future works and improvements.

1 | Background

This chapter aims to briefly overview the industry to which this thesis's optimization process and geometrical model will be applied. Here are presented the definitions relative to the construction industry, architecture, and the data structures used in computer science applied to building modeling.

1.1. Off-site Construction background

The first section presents the specific part of Architecture, Engineering and Construction (AEC) to which this thesis applies, namely off-site construction (OSC).

1.1.1. Definitions

The term off-site construction (OSC) refers to producing building components and accessories (such as floor slabs, wall panels, staircases, and balconies) in a factory. Those are then transported for assembly on-site later in a different location. Different types of off-site construction exist. Terms like *prefabrication* and *modular construction* are used interchangeably to specify different methods of implementing the broader term of off-site construction.

The term modular design emphasizes the increase of standardization of building elements which can be built in series and combined. With the term prefabricated building, the focus is on the practice of transferring a large amount of work from the traditional on-site operation to an off-site factory.

Therefore, a core aspect of OSC is the production of building elements, which can have different scales and complexity. [2] categorizes the elements that can be produced off-site as follows:

- **Components**: non-structural elements such as doors and windows. Those are infeasible to be produced on-site;
- **Panels**: structural elements such as walls, roofs, and floors. They are defined also

as *non-volumetric*;

- **Pods**: 3D structures self-contained. An example: bathroom or kitchen pods, which are then assembled to a frame on-site;
- **Modules**: ready to use building elements, including complete fixtures and fittings;
- **Complete buildings**: the entire building is manufactured off-site as a single unit and then transported on-site to be installed and connected to the foundation.

Figure 1.1 shows a fully finished wall panel made by a structural framing with exterior cladding and fenestration; prefabricated walls can also include water/air/vapour barriers and insulation. After being built in a factory, walls are transported on-site, where they are lifted and installed.

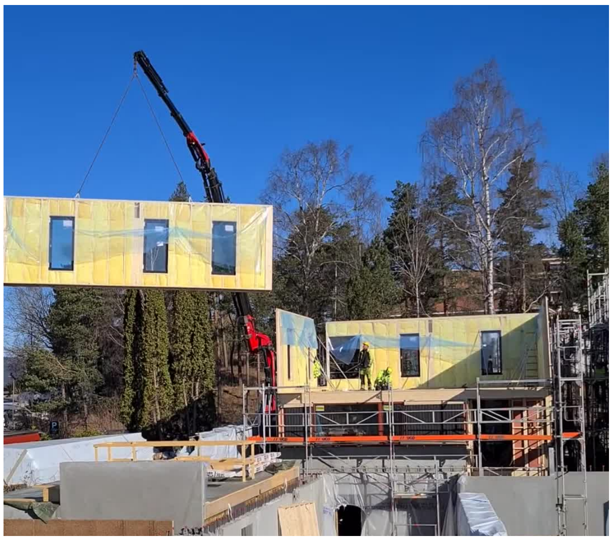


Figure 1.1: *Example of installation of a wall element in a construction site in Oslo, Norway. Source: author personal collection.*

To better understand the scale and scope of prefab elements, Figure 1.2 reports a schema of possible elements used in OSC, showing different levels of complexity of the element and the scale in terms of the physical dimensions of the element itself. A broad and continuous spectrum of elements can be considered under the umbrella of OSC elements. However, there is another more straightforward way to categorize these elements, just dividing them into two main types: 2D panelized and 3D volumetric.

This thesis directly addresses the 2D panelized paradigm of OSC, focusing on the geometric optimization for those panels. One example of wall panels is the Cross-laminated timber (CLT) which consists of different layers of timber, usually spruce, larch or pine, combined with structural adhesives [4].

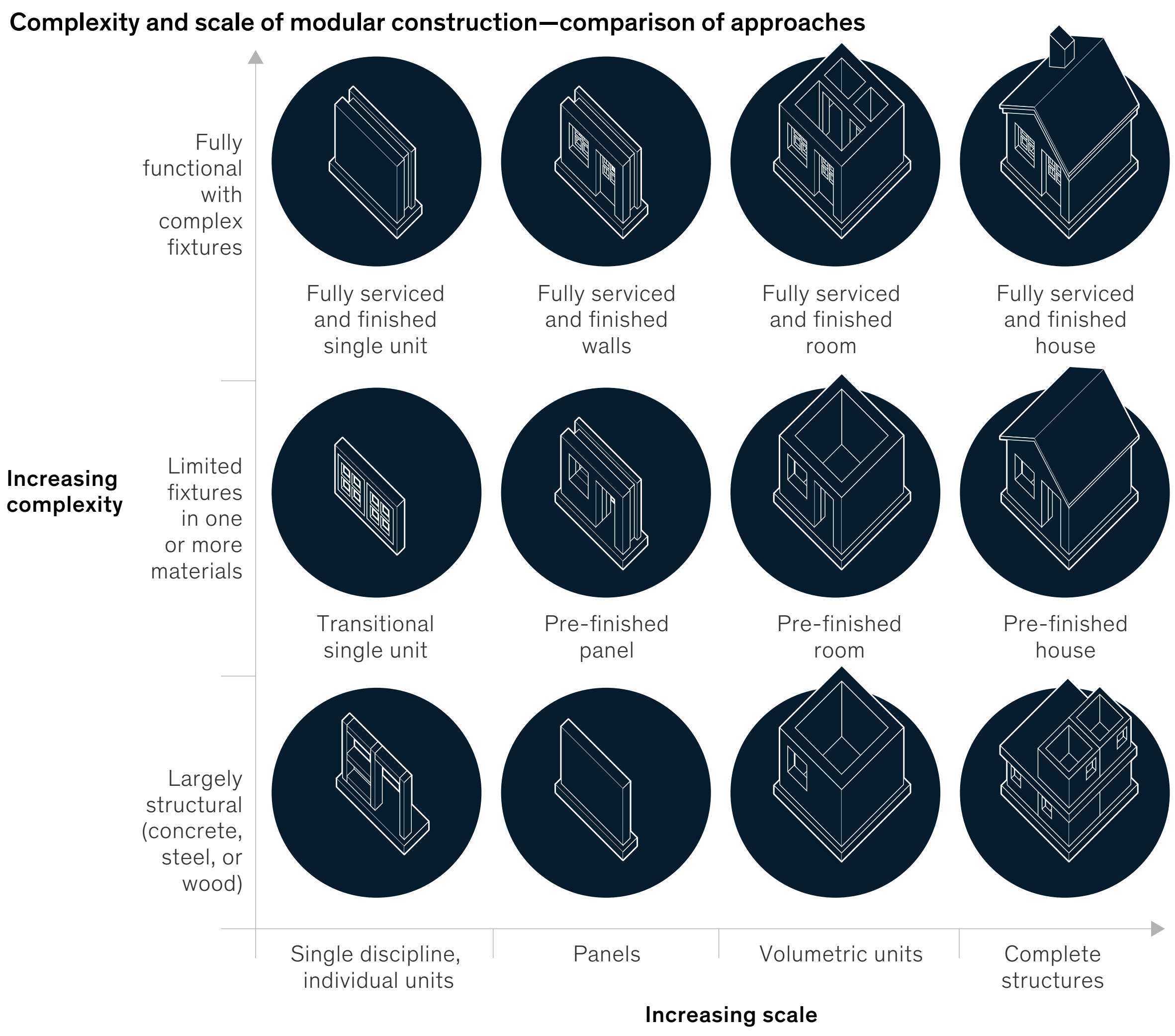

Complexity and scale of modular construction—comparison of approaches
Increasing complexity
Fully functional with complex fixtures
Limited fixtures in one or more materials
Largely structural (concrete, steel, or wood)
Fully serviced and finished single unit
Fully serviced and finished walls
Fully serviced and finished room
Fully serviced and finished house
Transitional single unit
Pre-finished panel
Pre-finished room
Pre-finished house
Single discipline, individual units
Panels
Volumetric units
Complete structures
Increasing scale

Figure 1.2: *Scale and complexity of modular construction element. Source: [3].*

1.1.2. Market and adoption, current state

Modular or prefabricated construction is nowadays just a niche market in most of the world, but it is expected to grow rapidly soon.

There is no uniform geographical distribution of the percentage of adoption and the type of elements mainly used. As an example, volumetric pre-assembly is the de-facto approach in North America and Australia [5], while the 2D-panelized one is the most used in many European countries [6].

According to [7]:

> In North America, the modular building industry has increased from 2.37% of construction expenditure in 2014 to 3.17% in 2017, while the off-site construction share in the Japanese market is 12–15%, and 50–90% in Sweden where

> panelized construction is dominating the market.

In China, according to [8] :

> During the 13th Five-Year Plan period, the average annual growth rate of new prefabricated building areas reached more than 50%. In 2020, 630 million square meters of prefabricated building area will be started, accounting for 20.5 percent of China's total new construction area that year.

Prefabricated elements have been used not only for new buildings but also to retrofit existing ones to improve energy efficiency [9].

1.1.3. Potential of OSC

This section addresses the reasons why OSC can have substantial advantages over traditional construction methods. [10] discusses the main drivers and advantages of OSC, which can be summarized in the following points:

1. faster construction time;
2. reduction in construction cost;
3. increased quality;
4. waste minimization;
5. reduction in carbon emissions.

The first advantage, reduction in construction time, is intrinsic in OSC since the time spent on site is reduced by prefabricating the elements in an off-site factory using industrialized approaches. This reduction in time has been estimated to be around 20–50% compared to on-site [3, 4]. Reduction of construction time leads to another advantage: faster solvency for developers and cost-effectiveness, which also turns into a faster return on investment [10]. All-weather production is another advantage of OSC, especially for countries in northern Europe.

The second advantage can refer to the construction cost and the life-cycle costs. On-site operation costs are drastically reduced using prefab elements that only need to be assembled on-site. Regarding life-cycle costs, for example, the airtight joining of components and superior insulation placement given by factory-driven precision will reduce the life-cycle cost of heating in a prefab building [5].

Finally, reducing carbon emissions is another crucial aspect of modular construction, especially if the modules are made of wood. Using timber elements allows storing almost

50% of carbon in the mass of the structure [9].

1.1.4. Challenges in the adoption of OSC

It is also crucial to understand in which contexts OSC will have the most impact. It has traditionally been considered when the structure has a degree of repeatability and the costs of transporting the elements are less than the on-site operations [3]. However, this thesis wants to prove that it is possible to build almost any geometric floor plan using mostly standardized elements. This will help solve the lack of standardization for prefab elements [4].

Other obstacles to adopting OSC are regional differences in regulations, which make prefabrication a risk for some property developers not used to this paradigm. Evaluations on whether to choose prefabrication to develop a project are out of the scope of this thesis. However, the methods presented here can be used to assess better if a project is suitable for prefabrication. This is because this thesis method will provide an exact list of elements needed to prefab a project, which can help assess the final costs.

1.2. BIM

The second part of the background is about the tools and the methodologies of the Architecture, Engineering and Construction (AEC) industry related to Computer Science.

The first concept to be introduced is Building Information Modeling (BIM). BIM can be defined as a technology that allows representing a digital virtual model of a building structure, including details coming from different engineering and design disciplines. BIM is based mainly on 3D models, enriched with information database technology and includes the software tools needed to operate with the models. The use of BIM is spread across different stakeholders in a construction project, such as architects, engineers, and contractors [11]. BIM is considered a core part of the digitization of AEC, but there are still barriers to its adoption in the industry [12].

1.2.1. IFC standard

One of the challenges in BIM is the need to exchange information between different parties involved in a construction project. That is where the need arises for a recognized standard to implement a data structure to allow collaboration and communication between systems and stakeholders.

Industry Foundation Classes (IFC) is an ISO standard, ISO 16739-1:2018 [13], created by BuildingSMART, which creates a “standardized, digital description of the built asset industry which is vendor-neutral and usable across different software platforms, hardware devices, and interfaces for different use cases” [14].

One core aspect of this data model is that it is made out of a representation of both the geometry and semantic structure of a building model using an object-oriented approach. This means that a building is represented by splitting it down into different building component entities with a geometric representation and some semantic properties and relationships.

The IFC schema codifies objects, properties and relationships from different domains related to the AEC industry. Those domains include building and architectural elements and processes and management related to the building. IFC helps different stakeholders in a construction project exchange information relative to architecture, building service, structural engineering, procurement, construction planning, facility management, project management, client requirement management, and building authority for permits and approval.

This thesis focus is only on the architectural part of the IFC schema.

IFC encodings

IFC files can be encoded in different ways; the most commonly used is the STEP Physical Format (SPF or IFC-SPF), the most compact format available for encoding [15]. Other formats include XML and JSON, but these have larger dimensions. IFC-SPF is based on the ISO standard for clear text representation of EXPRESS data models ISO 10303–21 [16]. In this thesis are used IFC-SPF files. An extract of one of those is reported on Listing 23 as an example of how the data is structured. Some peculiar aspects of the EXPRESS language are:

- An **entity type** is the equivalent of a class in object-oriented programming theory;
- For every entity type, different **attributes** can be defined;
- An attribute can either be: **explicit** (where a direct value is assigned), **derived** (when the attribute refers to another entity type), **inverse** (the same as derived but referring to the other end of the relationship);
- Derived attributes are used to describe relationships between different entity types. For example, type A can have as attribute an object of type B. This creates a relationship A $\rightarrow$ B;

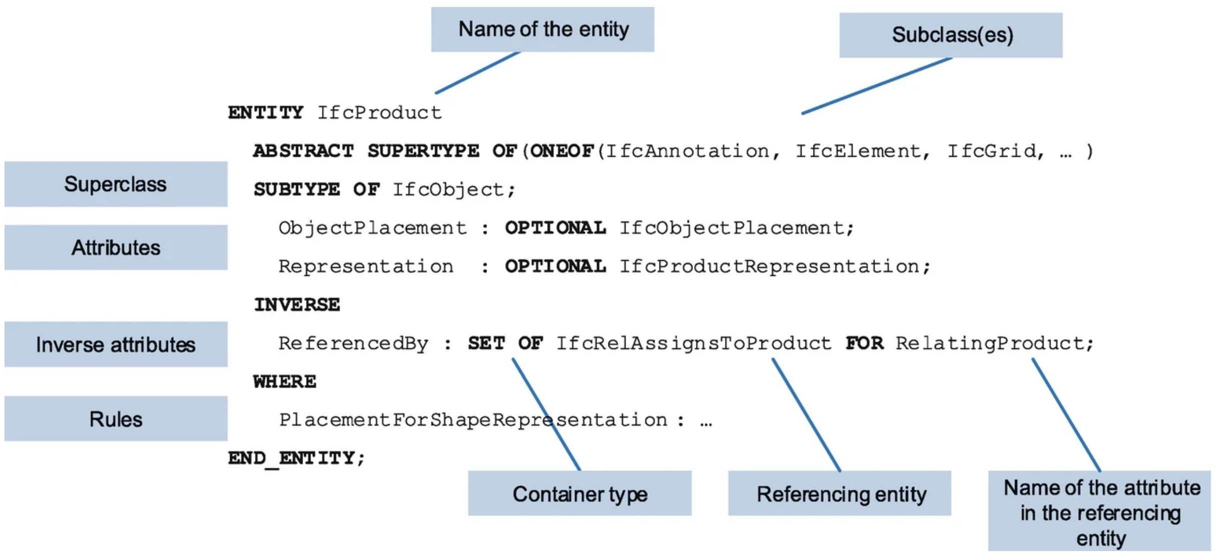
- Inverse attributes define the relationship in reverse order, so in the example above, type B will have an inverse attribute which relates to type A;
- An entity type can be the **subtype** of another entity type and also a **supertype**. In other words, the EXPRESS schema implements the object-oriented concept of inheritance.

The first level of subdivision of an IFC-SPF file is in two parts:

1. HEADER section, containing metadata about the file itself (lines 2–6 in Listing 23);
2. DATA section, containing the project information (lines 8–17 in Listing 23).

This thesis considers only the DATA part. Every line in the data part encodes an IFC entity. Every line starts with a unique ID, called ExpressID, representing univocally an entity in the file. For example, considering the entity (#323), after the ExpressID there is the name of the entity type. In this case, it is *IfcWallStandardCase*.

After the name of the class, in round brackets, there is the list of attributes of the entity. In this example, the first one is the UUID `0BM7tu9KX2pBqXi82G7tTo`. The second is a reference to another entity, in this case it is to (#13) of type *IfcOwnerHistory*, the fourth is the name given to the entity (`'mywall'`) and so on. The EXPRESS schema gives the complete list of the attributes, inverse attributes and rules. Figure 1.3 gives an example of EXPRESS schema with some annotation to describe the different parts of the definition of an entity type.



```
ENTITY IfcProduct
  ABSTRACT SUPERTYPE OF(ONEOF(IfcAnnotation, IfcElement, IfcGrid, … )
  SUBTYPE OF IfcObject;
    ObjectPlacement : OPTIONAL IfcObjectPlacement;
    Representation  : OPTIONAL IfcProductRepresentation;
  INVERSE
    ReferencedBy : SET OF IfcRelAssignsToProduct FOR RelatingProduct;
  WHERE
    PlacementForShapeRepresentation : …
END_ENTITY;
```

Figure 1.3: *Example of EXPRESS definition for entity type IfcProduct Source: [17].*

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('ViewDefinition␣[CoordinationView]','Option␣[Filter:␣VisibleElements]'),'2;1');
FILE_NAME('examplefilename.ifc','2010-10-27T21:32:24')
FILE_SCHEMA(('IFC2X3'));
ENDSEC;

DATA;
#13= IFCOWNERHISTORY(#12,#5,$,.ADDED.,$,$,$,1288207919);
#36= IFCDIRECTION((0.,0.,1.));
#54= IFCPROJECT('247zxMA4zFuef9Vjr$GJAi',#13,'Default␣Project',$,$,$,$,(#51,#145,#247),#26);
#247= IFCGEOMETRICREPRESENTATIONCONTEXT('Plan','Model',3,1.0000000E-5,#44,#243);
#320= IFCLOCALPLACEMENT(#125,#317);
#323= IFCWALLSTANDARDCASE('0BM7tu9KX2pBqXi82G7tTo',#13,'mywall',$,$,#320,#393,'0B587DF8-2548-42CC-BD-21-B080901F7772');
#380= IFCARBITRARYCLOSEDPROFILEDEF(.AREA.,$,#376);
#381= IFCAXIS2PLACEMENT3D(#40,#36,#28);
#384= IFCEXTRUDEDAREASOLID(#380,#381,#36,500.);
#387= IFCSHAPEREPRESENTATION(#247,'Body','SweptSolid',(#384));
#393= IFCPRODUCTDEFINITIONSHAPE($,$,(#354,#387));
#15559= IFCPRODUCTDEFINITIONSHAPE($,$,(#15543,#15555));
#15543= IFCCARTESIANPOINT((349.64918,70.,1459.2768));
#15567= IFCMATERIAL('Betong');
```

Listing 1.1: *Example of IFC file with STEP encoding.*

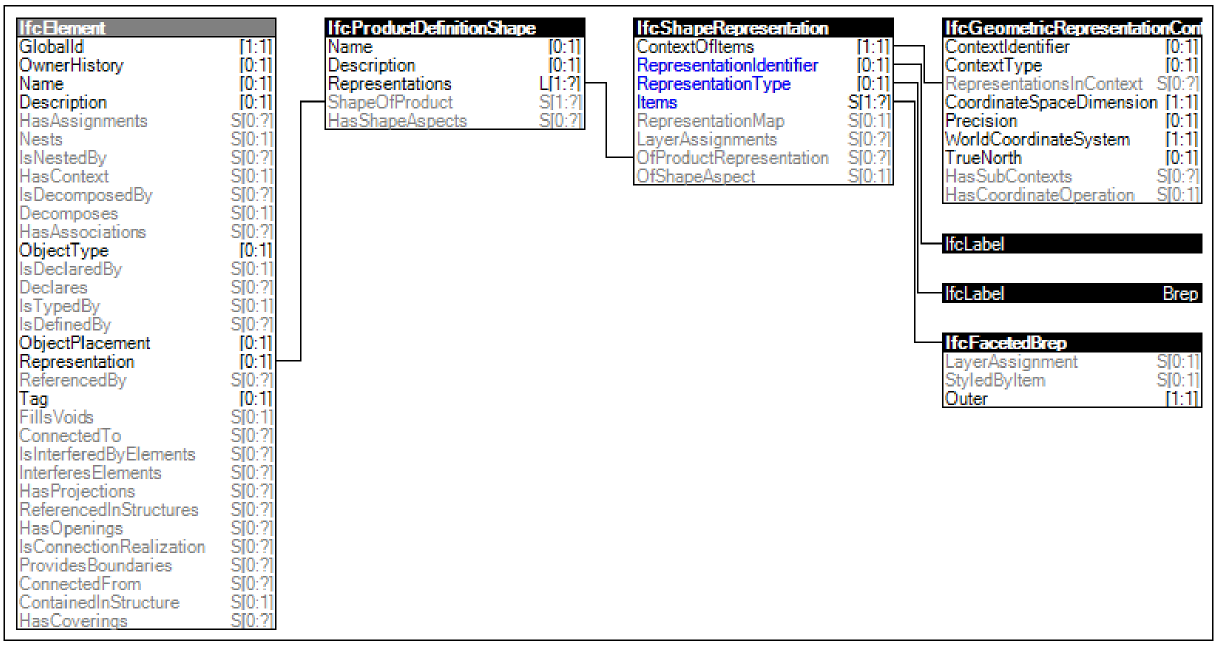


Figure 1.4: *Inheritance graph for IfcElement. Extract of the IFC schema [14].*

For this thesis work, it was essential to understand the EXPRESS language to work with IFC files, especially for developing the algorithms in Chapter 4.

Boundary Representation

The IFC standard uses different ways to represent 3D solid objects: including boundary representation (B-rep), swept solid, constructive solid geometry (CSG), clipping, mapped representation, and, in some cases, surface model [18]. Figure 1.4 reports an extract of the IFC schema with the inheritance graph for an entity of type *IfcElement*, for which it is defined an attribute **Representation**. Following the nested relationships comes the **RepresentationType** which can be of type `Brep`.

In this thesis, all the geometry from IFC files has been converted to B-rep. In a B-rep, a solid is represented by surfaces that create a boundary between the inner and the outer surrounding the solid itself. An example is given in Figure 1.5. The boundary representation of a model is made of two parts: topology and geometry. The topology defines the structure of the model, the geometry its shape [19].

The topological part is made out of vertices, which, connected, make an edge. Then edges are connected to create the faces. The geometry part consists of points, curves and surfaces. A face is a bounded portion of a surface, an edge is a bounded piece of a curve, and a vertex lies at a point.

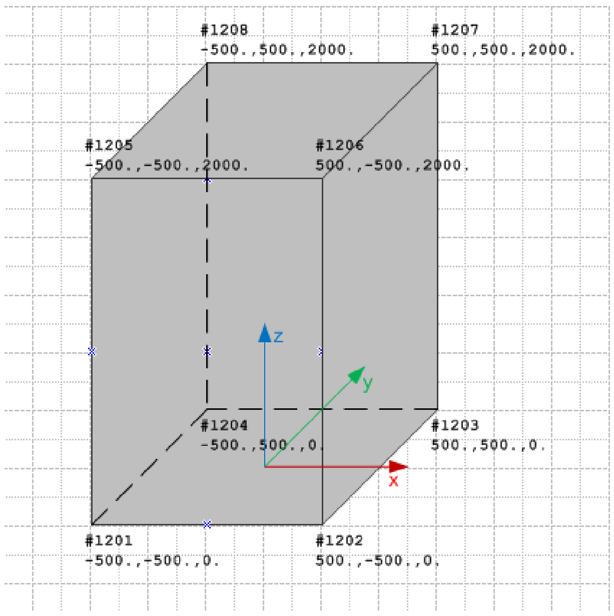


Figure 1.5: *Example of block geometry represented as boundary representation (B-Rep).*

Figure 1.6 reports a schema of the data structure of the boundary representation.

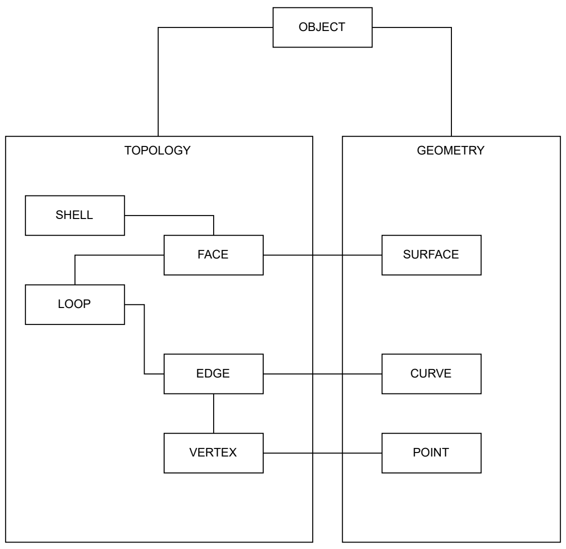


Figure 1.6: *Data structure of Brep representation. Source: [19].*

2 | Problem formulation and state of the art

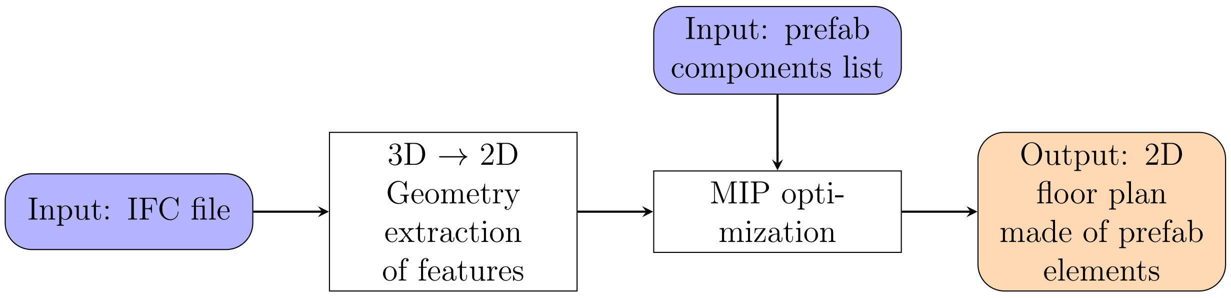
This chapter discusses the problem to be solved with this thesis work. The target users are wall panel manufacturers who work off-site. A typical scenario is when a manufacturer receives a building model from an architect that has not developed it with prefabrication in mind.

The model received usually consists of an IFC file since it is the open standard for data exchange in the construction industry. It is not always possible to assemble the model's exact geometry with discrete length panels. Lots of variability in the model dimensions can occur, and the lack of parametrization brings the need to redraw it. The manufacturer performs manually this redrawing, using its elements as essential pieces to generate a new geometry, usually using CAD tool. During this process, the manufacturer wants to minimize the cost of the prefabricated building. However, simultaneously, it aims to modify the geometry to the minimum.

This thesis wants to automate the process of redrawing and optimizing architectural designs. Specifically, given as input a 3D model of a building and a list of prefabricated elements, the goal is to return a new model made using the elements provided. The output should have the minimum discrepancy over the input and the minimum manufacturing cost.

2.1. Approach

The problem of automation of the redrawing consists of two main parts. The first is obtaining a method to decompose a complex building architecture model into a minimal but exhaustive data structure to represent its geometry. The second part uses the above data structure and the prefabrication constraints to create a new optimized model. The order of exposition of the two parts is inverted in this thesis. Chapter 3 discusses first the optimization part, and then Chapter 4 the geometry extraction from IFC files. The inversion is to underline the importance of the optimization part more.


Input: prefab components list
Input: IFC file
3D → 2D Geometry extraction of features
MIP optimization
Output: 2D floor plan made of prefab elements

2.2. Challenges and requisites in quantity extraction from IFC files

Without a standardized input, it is impossible to feed the optimization model in an automated way using relevant architectural data. For this reason, it has been chosen to use IFC files as the process's input. However, the extraction of quantities from those files is a challenge itself. The next part of this chapter discusses the technical complications regarding extracting information from those files, which are well pointed out in [20]:

> It is challenging for construction practitioners to obtain quantities in connection with construction activities from a BIM design model. Considerable human intervention must thus be involved to interpret the process model and to manually quantify the BIM product model in accordance with the process description.

The main challenge resides in the semantic/spatial separation of information in IFC files, which makes them ambiguous, highly redundant, offering multiple ways to define objects, relations and attributes [21]. This is well explained in [22]:

> BIM models are able to represent many facets of a building, in addition to geometrical and relational information, for example by using predefined and extensible property set or reference to external data sources. However, datasets from practice (DURAARK 2015) show that BIM models typically contain this information only partly and have heterogeneous levels of information. Metadata records that might exist for one building might be absent for the other.

In the following subsections are discussed some issues in the quality of IFC files which make the data extraction a challenge.

2.2.1. Misclassified entities

One prerequisite for this thesis work is that IFC entities are correctly labelled. Without a correct assignment of the IFC types to the entities, it is impossible to filter out only the walls from all other entities in an IFC file. This is needed because the optimization step needs only data relative to wall elements; it is not applicable if incorrect geometric data is given. Another case is if there is no entity classification in an IFC file. Experience from industry and IFC datasets shows that some BIM files lack type metadata. It can happen that in some IFC files most of the entities are of *IfcBuildingElementProxy* type[1]. This generic entity type holds no information on the real use of an entity in an IFC. Just by reading the semantic data, it is impossible to distinguish a wall from a slab or a roof.

This is a problem known in the literature: [23] proposes a method to classify IFC elements using a supervised Geometric Deep learning algorithm, using only the geometry of the entities. The accuracy is 85% for their custom dataset, but for the needs of this thesis and the challenges in BIM it is necessary to reach an accuracy of almost 100%, especially for the classification of *IfcWalls*.

2.2.2. Clash detection

Since the IFC files are human-generated with a CAD software, it can happen that two elements in the design overlap in the same space. In IFC files, physical constraints, such as avoiding overlapping, are not enforced. That is why software exists to solve the problem of clash detection, identifying if, where, or how two parts of the building (e.g., plumbing, walls, etc.) interfere with one another [24].

This thesis work assumes that the step of geometrical clash detection has already been taken.

2.2.3. Spatio-semantic consistency

Another issue that IFC files can present is how the spatial structure is represented. This is because there is a separation between the semantic description and the geometry in IFC. The first is the description of properties and the relationship between entities, which can also have a geometry description. An example can be the semantic description of the relationship *belonging to a storey* between an entity like a wall and the storey that contains that wall. In IFC this relationship is called *IfcRelContainedInSpatialStructure*.

[1]The *IfcBuildingElementProxy* is a proxy definition that provides the same functionality as an *IfcBuildingElement*, but without having a defined meaning of the particular type of building element it represents [14].

However, the wall's geometrical representation is not bound to this relationship, so a wall entity belonging to the second storey of a model may be actually geometrically positioned on the first floor. In other words, the IFC standard does not ensure that if an object is said to be contained within a particular building story, it must be spatially located in that story. This problem is discussed in [25] where a Query Language for Building Information Models (QL4BIM) [26] is used to check the spatial and semantic consistency in IFC models.

[22] presents a Supervised Learning approach based on Neural Networks in order to detect a storey in a BIM file automatically. The approach can segment into floors, metadata that is not semantically available in the IFC file but can be extracted by the supervised method, feeding it with spatial data.

This thesis assumes that the spatio-semantic consistency has already been checked in the input models, since it is a prerequisite for the proper extraction of data.

2.2.4. Inner and outer walls

When assembling a project with prefabrication, inner and outer walls are made of diverse materials. There are manufacturing enterprises that can only supply the outer walls of a building, and then the internal ones can either be built by another company or, more commonly, cut on-site.

Therefore, when dealing with a digital building project for prefabrication, it is crucial to include this metadata information for each wall entity. The IFC schema allows adding to an entity of type *IfcWall* a boolean property called *IsExternal*. This property is part of the *Pset_ WallCommon*. However, adding this metadata property is optional and must be done manually.

Nevertheless, the ontological fact of being an outer or inner wall can be inferred from the geometrical structure. [27] defines an *internal wall* as "the one which has two of its biggest vertical faces intersecting shell representations of spaces. If a wall shares two common faces with a space, then it is not external". Whereas an *external wall* "is not separating two internal spaces". Using the above definition, they classified inner and outer walls in an IFC file with a geometric algorithm. However, as the authors stated, the results relied on a proper definition of entities and the proper modelling of spaces. Without these conditions, it misclassifies the walls regarding their externality. Again, the prerequisite of the correct classification of entities is essential. Other methods are presented in the literature, such as a ray-tracing method for inner/outer contour detection [28]. Commercial software able

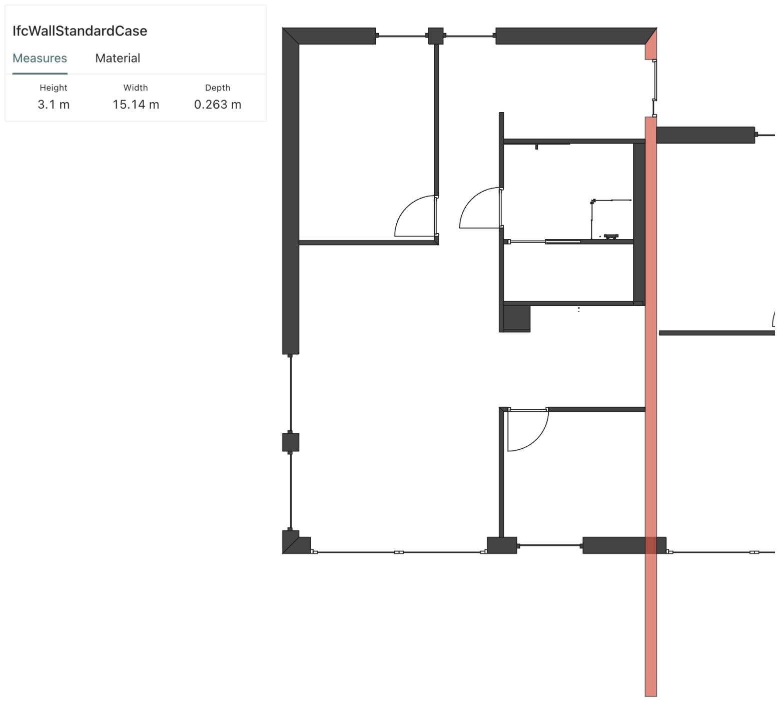


Figure 2.1: *Example of entity that is both inner and outer. Courtesy of Modulize AS [29].*

to classify inner and outer walls exists, like the Modulize platform [29].

Ontology of IFC elements

Here we briefly address the ambiguity problem when representing a physical entity digitally. Figure 2.1 shows a floor plan generated from an IFC file. In red is marked one singular IFC entity. It can be noted that the entity spans all over the building and is both inner and outer. For prefabrication purposes and data extraction in this thesis, a more coarse subdivision of entities is needed. In theory, an architect could even merge all walls into one entity in an IFC file, making it impossible to distinguish different structural elements from a manufacturing point of view. In [21], the need to have a standardized ontology for IFC files is discussed to remove ambiguities associated with differing viewpoints.

This thesis assumes that the input file is correctly split regarding the elements when dealing with the geometric extraction.

2.3. Similar works

This section reports a brief literature review of similar articles and papers to this thesis. It is divided into two parts to reflect the dual nature of the problem to solve. In the first part are reviewed articles mainly focusing on the optimization of building design; in the second part are reviewed articles primarily focused on the automatic geometric extraction of quantities from IFC files.

2.3.1. Optimization focus

Optimization is being applied to different building design problems such as: massing, orientation, facade design, thermal comfort, day-lighting, life cycle analysis, structural design analysis, energy and cost [30]. The methods used to perform the optimization are based on simulations and parametric modeling [31]. However, those methods are not always implemented, especially with a focus on prefabrication. Usually, the optimization methods should be applied in the preliminary design phase. However, this thesis addresses specifically the optimization of the geometry with constraints given by the prefabrication manufacturing processes. Therefore, the optimization that needs to be applied in this work is an added step after the initial design, made without prefabrication constraints.

In this section are reviewed some relevant works which apply operation research optimization to various problems in the construction industry. Those works are then compared with what is intended to do in this thesis.
[32] exposes an optimization model to optimize the configuration of the wall panels, subjected to panel fit constraints. Multi-wall panels are made of single-wall panels, all of the same height, framed together. The problem was formulated as a one-dimensional cutting stock optimization. The objective function was to maximize the length of multi-wall panels, which resulted in minimizing the number of multi-wall panels to be manufactured. The constraints were on the maximum and minimum length of the multi-wall panel. In their work, only one dimension was considered in the optimization, and constraints on the final geometry of the building were not added. The proposed solution was implemented as an add-on to the commercial software Autodesk Revit. They did not state which optimization solver was used. The most significant advance of this thesis work, compared to [32], is considering 2D constraints on the final geometry of the building and not just mono-dimensional ones.

[33] proposes a framework to optimize the design of buildings concerning construction waste. The core part of this work is to use a database of building elements that contains data regarding the waste. Then they propose an optimization at the design stage that involves human decisions when designing a building. Their pipeline is very similar to what we intend to do with this work; it only differs in the objective function. In real-life

design and construction practice, construction waste minimization will not be the only objective but needs to be considered in conjunction with other criteria such as time, cost, quality, safety, sustainability.

[34] proposes a method to optimize the generation of floor plans for residential housing. The paper aims to optimize the house's design parameters, such as the functionality, insulation, shielding from external noise, and outside view attractiveness. This work method is based on generating first some floor plans, then applying optimization to choose the best fitting ones from the set of the generated ones. This is a multi-objective optimization with the use of quadratic assignment. The novel contribution of [34] was modelling a subjective attribute, such as aesthetic attractiveness, into an optimization problem. Comparing [34] with the present thesis, they did not consider prefabrication in the generation of the floor plans. However, the present thesis focuses only on geometric optimization. Considering both geometric and functionality objectives can be an expansion of the present work.

[35] proposes an optimization model of floor plans generated with Generative Adversarial Networks (GANs). The objective was to minimize the area of the building. In order to keep the optimization problem linear, the authors used the sum of the height and width of the bounding box of the floor plan as the objective function.

2.3.2. IFC and BIM focused

[36] presents an automatic method to extract quantities from a BIM file. The goal of this work was to facilitate the process of quantity takeoff of wall-framing from BIM files. This implementation uses a proprietary visual programming extension called Dynamo for Autodesk Revit. The use of proprietary software is a limitation of the cited paper compared to the present thesis, which uses only open formats to achieve a more extensive use by stakeholders in construction. Another requirement of the cited article was not to change the input geometry. In contrast, this thesis allows slight changes in the geometry to allow higher degrees of standardization to reduce the costs.

[37] discusses the limits of using IFC files for the specific requirements of prefabricated construction. The researchers propose expanding the IFC schema that can be useful to model information relative to a prefabricated wall. A new *IfcPrefabricateWall* entity type is proposed to specialize the *IfcWall* type, adding new properties to describe the connection method of the elements and the reserved holes in the prefabricated element. This

proposed expansion of the IFC schema could be helpful in the future to represent the output of this thesis using the IFC format, mainly thanks to the modelling of connection relationships.

[38] shows a method to query IFC files after transforming them into a graph data structure, where nodes are the IFC entities, and the IFC relationships give the edges. However, the connection relationship is not always present in IFC files. The connection relationship is the most relevant for this thesis work. For this reason, the methods explained in [38] were not used. Instead, this thesis presents in Section 4.6 a method based on the geometry data of IFC to create a graph of connection of walls.

3 | Mixed Integer Linear Optimization Model

This chapter presents the original optimization model developed for floor plan geometry redrawing.

3.1. Input data structure

This chapter assumes data is ready and already computed with the structure requested for the optimization.

Some notes regarding the terminology used here: "segment of a wall", or "subwall" are terms used to refer to a part of a wall that does not contain any opening or is completely enclosing an opening. A visualization of this subdivision is in Figure 3.1.

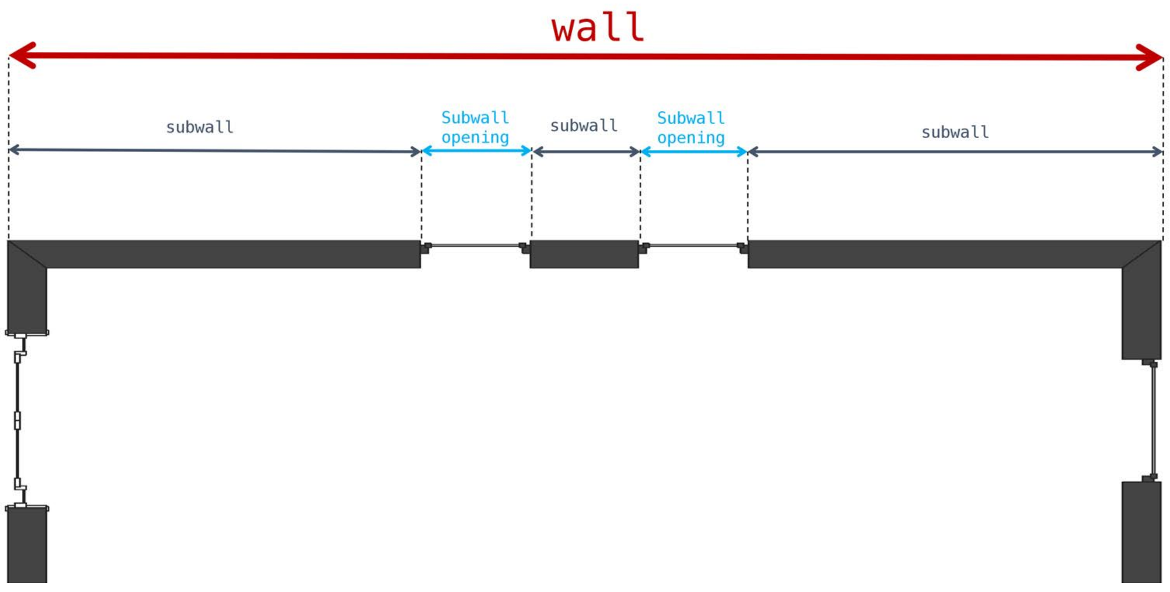


Figure 3.1: *Explanation of the subdivision of the walls. In red is the main wall. In blue and grey are the segments of a wall (also named subwalls) that compose the wall. The subwalls entirely enclosing an opening are in blue.*

The data needed in input for the model is a dataframe where every row describes a segment of a wall. The subwall can represent a part of solid wall or piece of wall containing an opening, with the following columns:

- type of subwall: either solid part of a wall or opening;
- length;
- orientation;
- order;
- coordinates x_0 , y_0, x_1, y_1 of the two extremes;

Table 3.1 is an example of a dataframe for wall segments. Here it is used the column *opening_id* to determine if it is an opening ($opening\_id \neq 0$) or a part of a solid wall ($opening\_id = 0$). There is also a column *wall_id* to identify, for every subwall, to which wall it belongs. It is possible to derive another dataframe with unique values of *wall_id*, which will be the union of all the segments that compose that wall.

For the database of prefabricated wall elements, the optimization model needs only a list of the lengths of each element. For example, the user can input this as a comma-separated list of values, like $0.3, 0.6, 1, 1.5$.

Requisites The dataframe must satisfy the following requisites to represent the polygonal perimeter of a house:

- the orientation of the walls can only be of those values $0°, 90°, 180°, -90°$;
- all rows with the same *wall_id* should have the same orientation (*deg* column value should be the same);
- the value column *len* should be equal to the Euclidean distance between (x_0, y_0) and (x_1, y_1):

$$len = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2} \tag{3.1}$$

- given two consecutive rows m, n (the order given by *index_col*):

$$index\_col(n) = index\_col(m) + 1 \implies x_1(m) = x_0(n) \wedge y_1(m) = y_0(n) \tag{3.2}$$

- given the first row a and the last row z

$$x_1(z) = x_0(a) \wedge y_1(z) = y_0(a) \tag{3.3}$$

wall_id	opening_id	x0	y0	x1	y1	len	deg	index_col
26008	0	-0.105	-0.105	-0.105	0.101	0.206	90.0	0
26008	26135	-0.105	0.101	-0.105	1.601	1.5	90.0	1
26008	0	-0.105	1.601	-0.105	3.956	2.355	90.0	2
26008	28322	-0.105	3.956	-0.105	5.756	1.799	90.0	3
.....	.....	......	.....	......	......	.....	.....	..
31085	37723	2.35	-0.105	0.097	-0.105	2.253	180.0	41
31085	0	0.097	-0.105	-0.105	-0.105	0.202	180.0	42

Table 3.1: *Example of some rows of the .csv input for the optimization.*

3.2. Optimization model

This section describes the Mixed-Integer Programming (MIP) model used to optimize the input geometry to use prefabricated walls and opening elements.

3.2.1. Input

- I : set of walls in the building;
- K : set of prefab elements to use;
- J_i : set of subwalls of wall i which are not enclosing openings;
- O_i : set of openings of wall i, which are not filled walls: $J_i \cap O_i = \varnothing$;
- l_{ij} : length of wall segment $j \in J_i$ in wall i, referred as subwall ij;
- l_{io} : length of opening $o \in O_i$ in wall i, referred as subwall io;
- m_k : length of prefab module k in list K.

3.2.2. Decision variables

In an optimization model, the decision variables are used to describe the quantities that the decision-makers would like to determine. In this thesis model, it has to be decided the number and type of prefab elements needed to build each wall of the building. This thesis does not consider determining the position of the prefab elements inside each wall of the output. It is sufficient to know the number and the type of elements to assemble each wall.

The other decision is how much discrepancy to allow between the original input and the redrawn.

Here is a recap of the variables introduced in the model:

- $\forall i \in I, \forall j \in J_i, \forall k \in K \quad x_{ijk}$: integer number of prefab elements of type k needed to build subpart wall j of wall i ;
- $\forall i \in I, \forall j \in J_i \quad \delta_{ij}^{+}$: continuous positive tolerance upward of approximation for subwall (ij);
- $\forall i \in I, \forall j \in J_i \quad \delta_{ij}^{-}$: continuous positive tolerance downward of approximation for subwall (ij).

In the variables above, two tolerances are needed. The reason is to avoid divergence of the objective function since it is a minimization problem. Therefore the tolerance variables used in the objective are positive continuous variables. Whereas in the constraints, the sign is added to make them respectively a tolerance upward and downward.

Some optimization tests have been performed with only the variables defined above. However, it happened to get an infeasible problem for certain combinations of geometry input and database of prefab lengths K. To solve this issue is needed to enrich the database of prefab elements with some "custom elements" that can have any length in a given range. A semi-continuous variable for every subwall ij that does not contain an opening is added to model the presence of custom elements. A semi-continuous variable is defined by [39] as "a variable that may be either zero or else in between one and some larger specified upper bound". Here follows the added variable:

- $\forall i \in I, \forall j \in J_i \quad c_{ij}$: semi-continuous variable, either 0 or in range $[C_{\min}, C_{\max}]$.

Where $C_{\min}$ is the minimum, and $C_{\max}$ is the maximum allowed length of a custom wall part.

In a linear model, semi-continuous variables are not allowed. Therefore the semi-continuous variable introduced must be decomposed. In practice, a binary variable is introduced for every semi-continuous variable, and other constraints are added [40]. In this case, the binary variable is:

- $\forall i \in I, \forall j \in J_i \quad y_{ij}$: binary variable, 0 if a custom element is not used to build subpart j of wall i otherwise 1.

The constraints added are:

$$\forall i \in I, \forall j \in J_i : \quad c_{ij} \geq y_{ij} C_{\min} \tag{3.4}$$

$$\forall i \in I, \forall j \in J_i : \quad c_{ij} \leq y_{ij} C_{\max} \tag{3.5}$$

Where c_{ij} is now a continuous variable, bounded between $[0, C_{\max}]$, although, due to the constraints defined, it will have value either 0 or in range $[C_{\min}, C_{\max}]$.

3.2.3. Constraints

The first set of constraints ensures that the version made out of prefab walls approximates the design wall's length. In other words, the δ variables can be considered a "penalization" for the output. The δ^- is subtracted from the δ^+. This difference gives the actual discrepancy of the output length since δ^- and δ^+ are both non-negative.

For every subwall ij:

$$\forall i \in I, \forall j \in J_i : \quad \sum_{k \in K} m_k x_{ij,k} + \left(\delta^+_{ij} - \delta^-_{ij}\right) = l_{ij} \tag{3.6}$$

Considering as well the custom elements the equation becomes:

$$\forall i \in I, \forall j \in J_i : \quad c_{ij} + \sum_{k \in K} m_k x_{ij,k} + \left(\delta^+_{ij} - \delta^-_{ij}\right) = l_{ij} \tag{3.7}$$

Then the following constraint is added to ensure that for every subwall not an opening, the output will use at least one prefab element.
For every subwall ij of every wall i:

$$\forall i \in I, \forall j \in J_i : \quad \sum_{k \in K} x_{ij,k} \geq 1 \tag{3.8}$$

Cycle constraints

Then have been considered the constraints to obtain an output perimeter closing in a loop. This means the output is a closed polygon. Under the assumption of orthogonality of wall joints (see Section 3.1), couples of subsets are identified, vertical "positive" walls and vertical “negative” walls. The same goes for the horizontal orientation.

The set of walls I is partitioned into four subsets K, two for each direction, vertical and horizontal:

$$I = K_1^{hor} \cup K_2^{hor} \cup K_1^{vert} \cup K_2^{vert} \tag{3.9}$$

The partition into subsets is obtained by grouping walls with the same orientation. According to requisites 3.1, the values in the *deg* column can only be of four distinct values, which guarantees the subdivision into the cycles. Figure 3.2 shows an example of subdivision of a perimeter of a building into cycles.

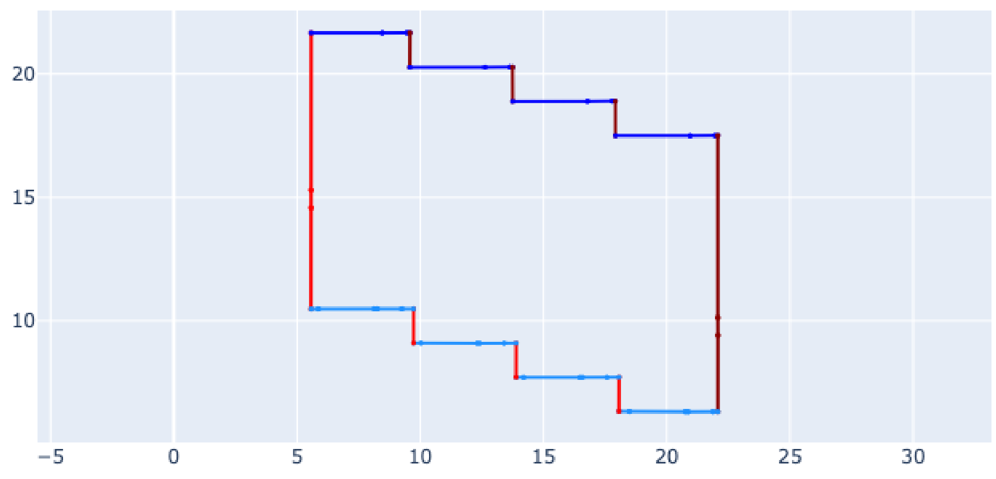


Figure 3.2: *Example of visualization of the cycles of a floor plan. Horizontal cycle 1 is in dark blue, cycle 2 is light blue. Vertical cycle 1 is red, vertical cycle 2 is dark red.*

Finally are written the following constraints for couple of subsets:

- Horizontal cycles:

$$\sum_{i \in K_1^{hor}} \Gamma_i = \sum_{i \in K_2^{hor}} \Gamma_i \tag{3.10}$$

- Vertical cycles:

$$\sum_{i \in K_1^{vert}} \Gamma_i = \sum_{i \in K_2^{vert}} \Gamma_i \tag{3.11}$$

Where Γ_i represents the output length of wall i. It is calculated with a sum of the following: the sum of all apertures, the optional custom piece and the sum of all parts of walls made of prefab pieces.

$$\Gamma_i = \sum_{o \in O_i} l_o + \sum_{j \in J_i} \left(c_{ij} + \sum_{k \in K} m_k x_{ijk} \right) \tag{3.12}$$

Combining the (3.12) with the (3.10) and the (3.11), respectively, the two cycle constraints are obtained:

Horizontal cycles:

$$\sum_{i \in K_1^{hor}} \left[\sum_{o \in O_i} l_o + \sum_{j \in J_i} \left(c_{ij} + \sum_{k \in K} m_k x_{ij,k} \right) \right] - \sum_{i \in K_2^{hor}} \left[\sum_{o \in O_i} l_o + \sum_{j \in J_i} \left(c_{ij} + \sum_{k \in K} m_k x_{ij,k} \right) \right] = 0 \tag{3.13}$$

Vertical cycles:

$$\sum_{i \in K_1^{vert}} \left[\sum_{o \in O_i} l_o + \sum_{j \in J_i} \left(c_{ij} + \sum_{k \in K} m_k x_{ij,k} \right) \right] - \sum_{i \in K_2^{vert}} \left[\sum_{o \in O_i} l_o + \sum_{j \in J_i} \left(c_{ij} + \sum_{k \in K} m_k x_{ij,k} \right) \right] = 0 \tag{3.14}$$

3.2.4. Objective

In this problem, two distinct objective functions differ in the goal to be achieved when choosing the variables. The first objective is obtaining the minimum discrepancy of the output concerning the original input lengths. The second optimization objective is to reduce the project cost calculated by counting the wall elements used. When modelling the cost of a wall panel, it has been considered unitary for any of the standard elements. After some direct meetings with wall manufacturers, the author was told to consider custom elements of a quadruple cost compared to the standard ones.

In order to handle the multi-objective optimization problem described above, this thesis uses the weighted-sum method. This consists of multiplying each objective with a user-defined weight [41, 42]. A convex combination of the objective is used in the present problem, with the coefficients λ and $1-\lambda$, and $\lambda \in [0,1]$.

Minimum discrepancy

When optimizing for the minimum output discrepancy, the tolerances for each subwall are the target to minimize.

In Section 3.2.2 two tolerances were introduced for each subwall: δ^+ and δ^-, both of them positive continuous variables. When those are summed, the total approximation error for each wall of the input is obtained. Therefore Δ_i is defined as the sum of the approximation error for wall i

$$\Delta_i = \sum_{j \in J_i} \delta^+_{ij} + \delta^-_{ij} \tag{3.15}$$

Equation 3.15 is the first objective function.

Minimum cost

When optimizing for the minimum output cost, the number of elements used is the target to minimize. A unit cost value is given to the standard elements and a quadruple cost to the optional custom element.

For a wall i has been defined N_i as the number of standard elements used to assembly the output wall i :

$$N_i = \sum_{j \in J_i} \sum_{k \in K} x_{ijk} \tag{3.16}$$

C_i is defined as the number of custom elements used to assemble the output wall i. This value is the cardinality of the set of semi-continuous variable c_{ij} when it has a non-zero value:

$$C_i = |c_{ij} > 0| \quad \forall j \in J \tag{3.17}$$

However, since the semi-continuous variable c_{ij} is modelled with a binary decision variable y_{ij}, the value of C_i is as follow:

$$C_i = \sum_{j \in J_i} y_{ij} \tag{3.18}$$

Assuming the binary variable is taking the integer values 1, 0 when is True and False respectively.

The second objective function is:

$$N_i + 4C_i \tag{3.19}$$

Finally, the convex combination of the two objectives is expressed as:

$$\min \sum_{i \in I} \lambda \Delta_i + (1-\lambda)(N_i + 4C_i) \tag{3.20}$$

Writing the Equations (3.15), (3.16), (3.18) in Equation (3.20) it is obtained the following complete objective function:

$$\min \sum_{i \in I} \left[\lambda \left(\sum_{j \in J_i} \delta_{ij}^{+} + \delta_{ij}^{-} \right) + (1-\lambda) \left(\sum_{j \in J_i} \sum_{k \in K} x_{ijk} + \sum_{j \in J_i} y_{ij} \right) \right] \tag{3.21}$$

3.3. Complete final model

Here is reported the final MIP model.

$$\begin{aligned}
\min \quad & \sum_{i \in I} \left[\lambda \left(\sum_{j \in J_i} \delta_{ij}^{+} + \delta_{ij}^{-} \right) + (1-\lambda) \left(\sum_{j \in J_i} \sum_{k \in K} x_{ijk} + \sum_{j \in J_i} y_{ij} \right) \right] \\
\text{s.t.} \quad & \sum_{k \in K} x_{ijk} \geq 1 \qquad \forall i \in I, \forall j \in J_i, \\
& c_{ij} + \sum_{k \in K} m_k x_{ij,k} + \left(\delta_{ij}^{+} - \delta_{ij}^{-} \right) = l_{ij} \qquad \forall i \in I, \forall j \in J_i, \\
& \sum_{i \in K_1^{hor}} \left[\sum_{o \in O_i} l_o + \sum_{j \in J_i} \left(c_{ij} + \sum_{k \in K} m_k x_{ij,k} \right) \right], \\
& - \sum_{i \in K_2^{hor}} \left[\sum_{o \in O_i} l_o + \sum_{j \in J_i} \left(c_{ij} + \sum_{k \in K} m_k x_{ij,k} \right) \right] = 0 \qquad \forall i, \\
& \sum_{i \in K_1^{vert}} \left[\sum_{o \in O_i} l_o + \sum_{j \in J_i} \left(c_{ij} + \sum_{k \in K} m_k x_{ij,k} \right) \right], \\
& - \sum_{i \in K_2^{vert}} \left[\sum_{o \in O_i} l_o + \sum_{j \in J_i} \left(c_{ij} + \sum_{k \in K} m_k x_{ij,k} \right) \right] = 0 \qquad \forall i
\end{aligned}$$

4 | Geometry processing from IFC files

The quantities needed to feed the optimization model are not directly available in IFC files. There is a need to develop algorithms to extract them from the geometry data of the IFC. This chapter presents an innovative method to automatically select the walls in an IFC file suitable for prefabrication and calculate their dimensions. In this thesis work, the relevant part of the building consists of the external perimeter.

In addition, an automated way to get spatial relationships between walls is presented to understand how those are connected and their relative positioning. All this information is implicit in the geometry of IFC files and needs to be derived using the algorithms presented here.

4.1. Preconditions

The first step is to identify the target building types that can be used to apply the present method. As discussed in Section 2.2, IFC files present considerable variability and can represent a wide range of different buildings. Therefore, the domain of possible input buildings needs to be restricted to obtain consistent and meaningful results in the prefabrication optimization part.

Here are the preconditions that a generic IFC file must have before starting the extraction of features:

P.1 All the walls subjected to the feature extraction must be of the *IfcWall* type;

P.2 Elements that do not represent a wall cannot be of type *IfcWall*;

P.3 Every wall to be optimized must have rectilinear edges;

P.4 When two walls intersect, the angle between them must be a multiple of 90°;

P.5 The height of all walls on a floor should be the same;

P.6 The perimeter of every floor must be a closed polygonal chain of walls.

Some of those preconditions are restricting the geometry of the input file that will be within the scope of this work, such as the case of curved walls and walls not orthogonal in their joints. Relaxing those preconditions are possible future improvements of the current method, which are discussed in Chapter 6.1.

Other preconditions are relative to the correctness of the input model, namely the correct definition of the type of the entities, a fundamental challenge in IFC, which is discussed in Section 2.2.1.

4.2. High-level flow

Here follows the high-level description of the steps needed for the geometric extraction.

1. Opening of IFC file;
2. Extracting the *IfcBuildingStoreys* and selection of the target floor;
3. Extraction of all wall entities in the floor;
4. Extraction of all the opening entities in the floor;
5. Extraction of all faces, for each wall in the floor;
6. Filtering faces perpendicular to the floor;
7. Detection of outer faces;
8. Extraction of 2D quantities;
9. Generation of a connection graph;
10. Perimeter detection;
11. Ordering the walls;
12. Export to .csv.

4.3. Initial processing and walls extraction

The process's first step consists of extracting geometry from an IFC file. This task will not be done using an original algorithm because it has already been solved by different

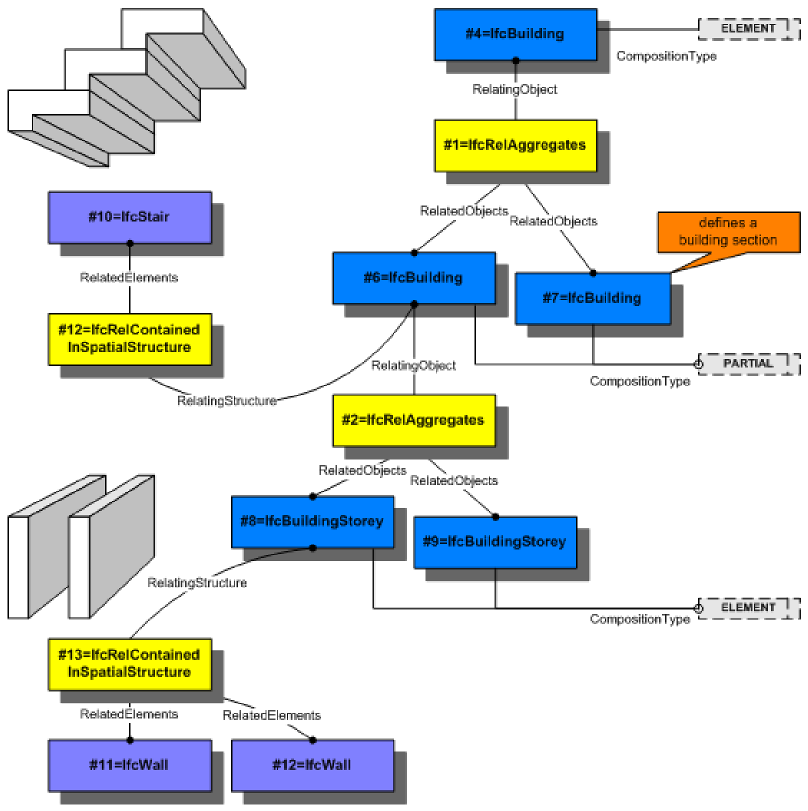


Figure 4.1: *Extract from the IFC EXPRESS schema, showing how the spatial structure is represented. In blue are the container elements such as IfcBuilding and IfcStorey. In yellow are the relationships between those and the element in purple.*

software packages, which have been reviewed more in-depth in Section 5.1.1. It is required to extract all entities of the IFC files and convert them to the B-Rep geometric representation, which is discussed in Section 1.2.1. Then the IFC file is queried to detect all *IfcBuildingStorey* entities which represent the different storeys of the building. After that, it is possible to get all the entities, particularly walls, openings, and slabs that belong to each storey previously detected. For this purpose, the objectified relationship of a storey entity called *IfcRelContainedInSpatialStructure* has been used, which references all entities of type *IfcProduct* contained in the storey. Figure 4.1 shows how IFC represents the spatial structure of a building. For example, given a wall in the IFC, using the inverse relationship *IfcRelContainedInSpatialStructure*, it is possible to get the *IfcBuildingStorey* where the wall has been positioned. Looping over all *IfcWalls*, it has been possible to extract all the walls divided by the placing storey.

This thesis approach consists in selecting a storey manually to apply the processing. Usually, it has been considered the first floor.

4.4. Abstraction of walls in 2D

Since this thesis only considers buildings in which all the walls on a floor must have the same height, as in precondition **P.5**, there is no need to extract data regarding the vertical dimension of the walls.

The minimum-discrepancy output building project will have only a standardized height, depending on the type of prefab modules used. So, a 2D feature extraction from the 3D geometry is needed to reduce the optimization part's complexity, which does not need to consider the height. That is why from now on, it will be only handled the geometry of the building from a 2D view.

This part of the algorithm aims to obtain a dataframe of walls from the IFC file. In this, there will be the vertexes of a wall and the vertexes of the openings in it. An important assumption was to ignore the detailed geometry of the door or window element contained in an *IfcOpeningElement*. A similar approach was used in [43]. The reason is that the geometric representation of elements such as doors and windows includes particulars not needed for the goal of prefabrication. The only relevant aspect to consider is the type of opening and the width.

Algorithm 4.1 Generation of a 2D Series of vertexes from B-rep shape representing a wall

```
Input: allWalls, openingsByWall
Output: wallVertexesDf
1: wallVertexesDf = new dataframe
2: for wall in allWalls do
3:     wallVertexes = wall.vertexes
4:     wallHull = ConvexHull(wallVertexes)
5:     for vertex in wallHull do
6:         wallVertexRowDf = {}
7:         wallVertexRowDf[id] = wall.id
8:         wallVertexRowDf[x] = vertex.x
9:         wallVertexRowDf[y] = vertex.y
10:        wallVertexRowDf[openingId] = 0
11:        wallVertexesDf.addRow(wallVertexRowDf)
12:    end for
13:    openingsOfWall = openingsByWall[wall]
14:    for opening in openingsOfWall do
15:        boundingBoxOpening = computeBoundingBox(opening)
16:        sectionOpeningWall = computeSection(boundingBoxOpening, wallShape)
17:        sectionVertexes = sectionOpeningWall.vertexes
18:        2dVertexesOpening = delete coordinate z from sectionVertexes
19:        for vertex in 2dVertexesOpening do
20:            openingVertexRowDf = {}
21:            openingVertexRowDf[id] = wall.id
22:            openingVertexRowDf[x] = vertex.x
23:            openingVertexRowDf[y] = vertex.y
24:            openingVertexRowDf[openingId] = opening.id
25:            wallVertexesDf.addRow(openingVertexRowDf)
26:        end for
27:    end for
28: end for
```

Algorithm 4.1 takes as input:

- `allWalls`: the list of all the relevant walls, in the form of an array of the boundary representation of the geometry of the walls;
- `openingsByWall`: a key-value dictionary that associates, for each wall entity, the

geometry of all its openings in boundary representation.

And creates as output:

- `wallVertexesDf`: a dataframe (implemented with the pandas library [44]) of all the relevant vertexes of each wall.

Figure 4.2 reports the scatter plot of the vertexes extracted from the 3D geometry using the method explained.

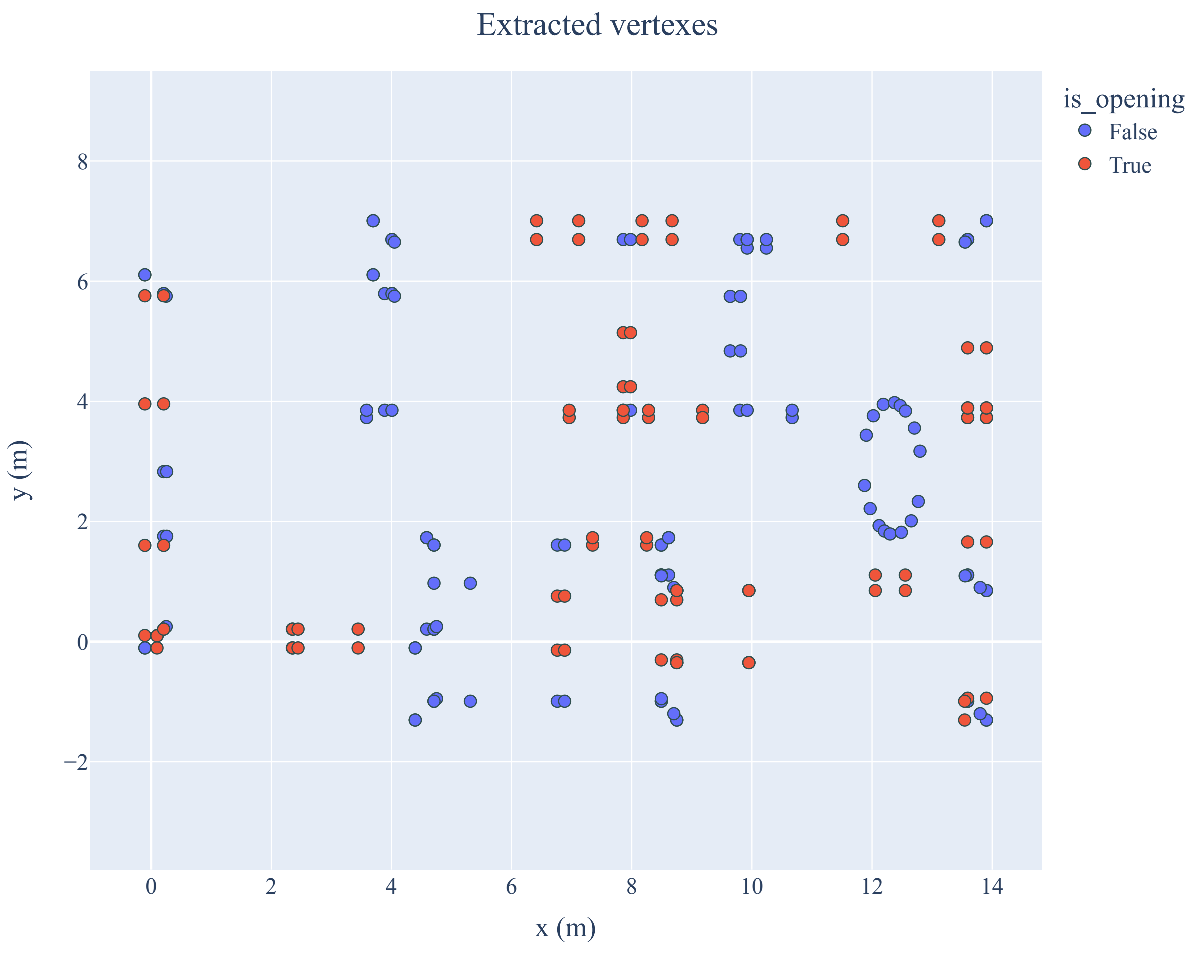


Figure 4.2: *2D plot of extracted vertexes from IFC geometry.*

4.5. Outer wall detection

One step needed to solve the problem is to detect the external perimeter of a building in an automated way. This is because the optimization model presented works only with the external walls. Therefore, segmentation of all the walls into the inner and outer categories is needed. Different algorithms that can be applied to solve this problem have been reviewed in Section 2.2.4.

This thesis uses a proprietary algorithm developed by Modulize AS [29] able to detect the outer faces of a building from an IFC file. Figure 4.3 shows the results of applying the detection algorithm to an IFC building model.

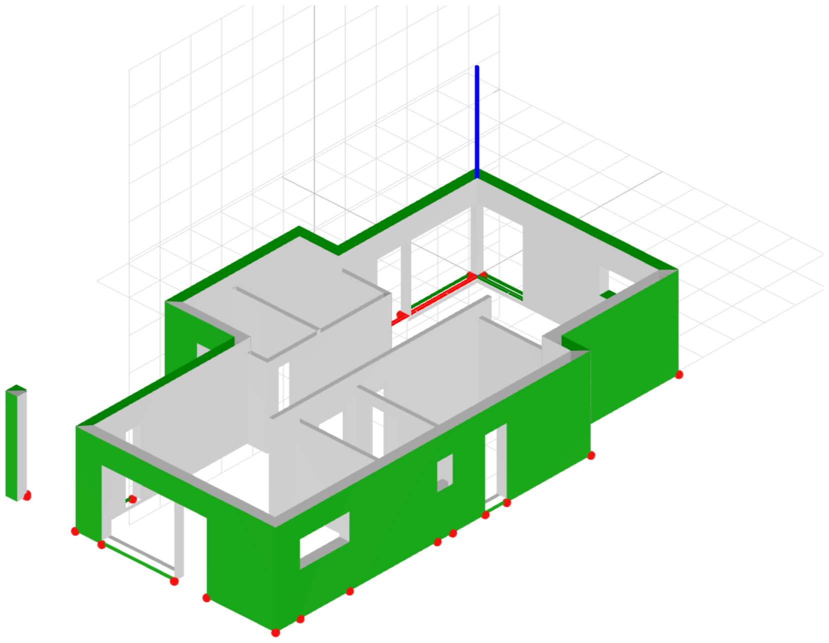


Figure 4.3: *Result of the inner and outer detection algorithm applied to an IFC building model. In green are identified the outer faces of IfcWalls present in a storey of the input model. In grey the inner ones. The red dots indicate the vertexes of the outer faces.*

4.6. Connection graph of walls

The information about which walls are connected is not explicitly present in most of the IFC files [20]. Therefore, it is needed to extract the relationship between the walls that will be joined together when using prefabricated elements from the geometric domain of IFC. This thesis proposes the creation of a *connection graph* of walls from IFC files to solve the issue of identifying connected elements.

The goal is to obtain a graph where nodes and edges are defined as follows:

- **Node**: represents a wall;
- **Edge**: between wall a, b if wall a is connected to wall b.

Algorithm 4.2 explains the steps for the creation of the connection graph. The input required are the list of the external walls (`externalWallList`), and a constant parameter (`COLLISION_DIST`) which is the minimum distance to identify two connected walls. In the implementation of this thesis it has been used as collision distance threshold a value of half of the minimum wall thickness of the building.

Algorithm 4.2 Generation of graph of connected external walls

```
Input: externalWallList, COLLISION_DIST
Output: adjGraph
1: visitedWall = new Set()
2: adjGraph = new Graph()
3: for i in externalWallsList do
4:     points_i = i.pointList
5:     adjGraph.addNode(i)
6:     visited.push(i)
7:     for j in externalWallsList do
8:         if j not in visited then
9:             points_j = j.pointList
10:            for point_i in points_i do
11:                for point_j in points_j do
12:                    dist = euclideanDistance(point_i, point_j)
13:                    if dist < COLLISION_DIST then
14:                        adjGraph.addEdge(i, j)
15:                    end if
16:                end for
17:            end for
18:        end if
19:    end for
20: end for
```

Figure 4.4 shows the results of applying the algorithm to compute the connection graph for the outer walls.

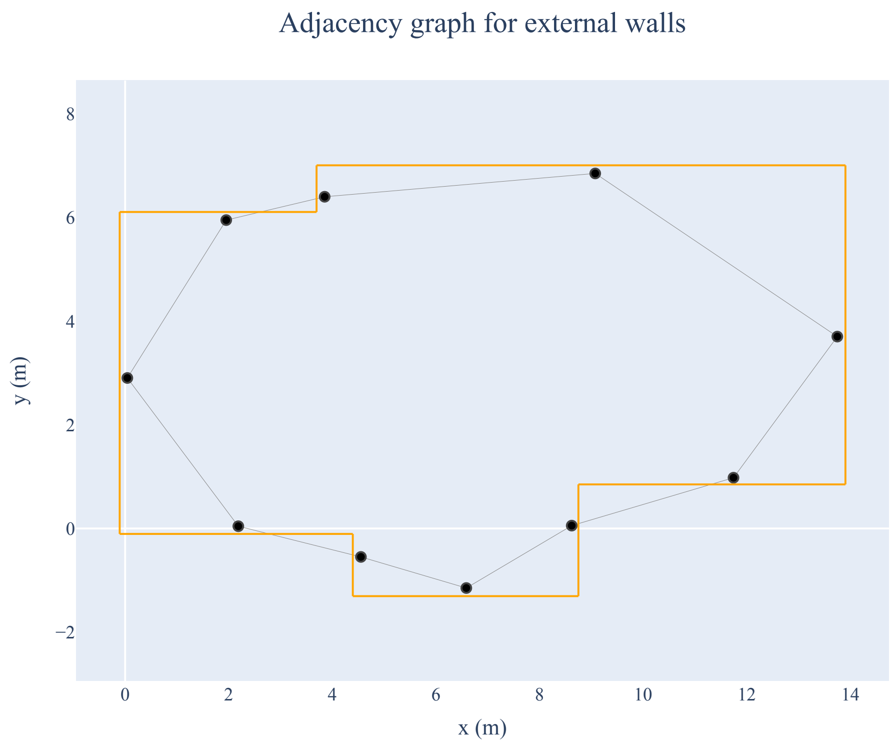


Figure 4.4: *Graph visualization of the outer walls of the building. In orange are the extracted segments representing the walls' outer profile. The black dots are the nodes of the graph, positioned in the xy plane at the centre of the wall they represent. Then the nodes are connected by edges to create the connection graph.*

4.7. Cycles identification

The creation of the connection graph was an intermediate step to identify the *cycles of walls*. With *wall cycles* in this thesis is intended a partition of the set of walls of a floor plan, such as in each cycle there are walls with the same orientation. The orientation should be considered regarding the floor plan of a building, which is a plane parallel to the ground. Therefore, the orientation here considered is in 2 dimensions.

When representing the perimeter of a house with the connection graph, it is expected to obtain a cyclic graph, according to the canonical definition of graph theory. This is because the thesis is applied to the building projects where the perimeter of a floor is a closed polygonal chain, as stated in Precondition **P.6**. Without this assumption, it is not possible to continue with the identification of cycles because the graph cycle is not guaranteed to be cyclical.

4.8. Manual extraction of geometry from floor plans

When performing the testings reported in Chapter 5, it was hard to find enough IFC files that satisfy the constraints for the automatic extraction of the geometric features. A manual pre-processing of the input was still needed in most of the IFC files used to apply the feature extraction algorithm successfully. The manual pre-processing could consist in splitting wall entities, filtering out non-relevant entities, and correctly marking the properties of the entities. This would have been too complex and would have taken too much time to prepare the dataset. Given those obstacles and the need to have enough data for the following optimization problem, the author decided to use an alternative approach to obtain the input data needed for batch testing. This approach involves using proprietary software to draw on top of existing PDF files in order to extract the 2D geometry. The advantage was the speed of manual generation of floor plan data. Figure 4.5 shows how a PDF floor plan looks after the manual marking of the geometry. The ordering needed to extract cycles was performed manually, and the software generated a .csv file automatically with the exact structure of rows and columns required for the optimization process.

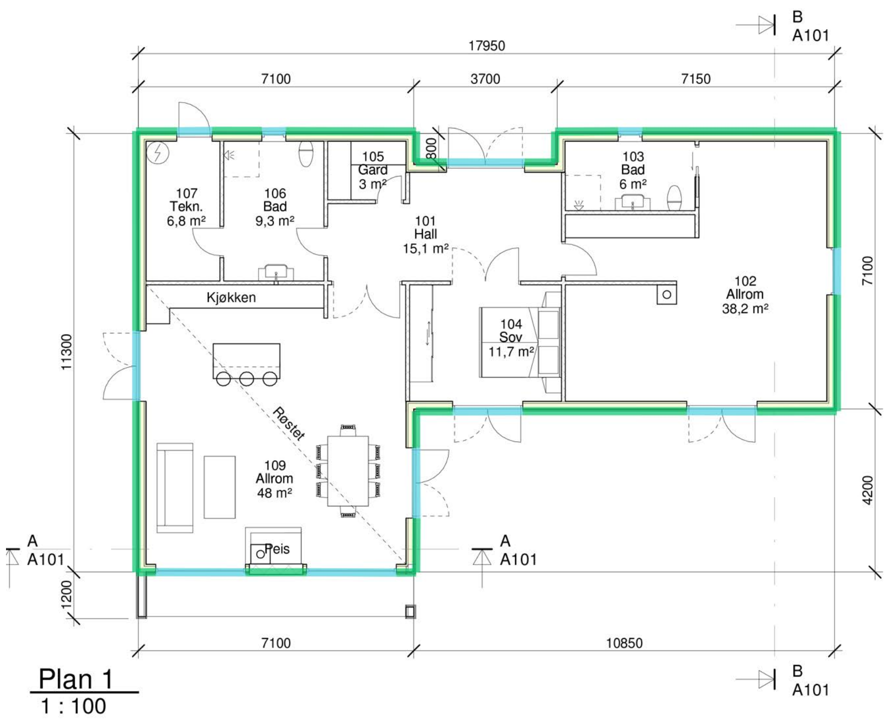


Figure 4.5: *Example of pdf after manual marking of outer perimeter. In light blue are marked the openings, and in green the solid walls. Source: Courtesy of Modulize AS [29].*

5 | Implementation, testing and results

The goal of this Chapter is to show the implementation and the results of the application of this thesis's method to different instances of building architectures. Firstly are given the implementation details on how the software was written and tested. Then are reported the tests performed relative to the optimization model. Finally, a case study is given, where the geometric extraction algorithms are tested on an IFC file of building architecture and compared with the industry manual approach.

5.1. Implementation

This section describes the computer program developed to implement the solution to the problem explained in Chapter 2. The language used for the implementation was Python, version 3.9.6, with the help of the open-source libraries described in this section.

The number of lines of code written for the current project is as follows:

- Geometry extraction: $\approx$ 1k;
- MIP model: $\approx$ 500;
- Auxiliary functions and drawing tools: $\approx$ 600.

5.1.1. Libraries used

This section presents the external software libraries used to implement the algorithms in this thesis. A brief review of the available packages is given, comparing their advantages and disadvantages in choosing them for the current work.

IFC parsing

The first step of geometric processing consists of parsing and extracting geometry data from the IFC STEP files. This task is already solved in literature and industry, so there was no need to implement a parser from scratch. Different IFC parsers and tool libraries are available, either open or closed source, with free or commercial licence. In [45, 46], different BIM/IFC libraries and software available on the market are compared, listing their peculiarities. Here is a brief list of the most used ones:

- **IfcOpenShell**: an open-source (LGPL) software library written mainly in C++ that uses under the hood Open CASCADE to convert the implicit geometry in IFC files into explicit geometry that any software CAD or modelling package can understand [47];
- **IFC.js**: a suite of open-source software packages that includes a C++ Web assembly parser and a Javascript/Typescript API to render IFC modules in a browser using Three.js rendering [48];
- **Xbim toolkit**: a .NET open-source software development BIM toolkit, the core libraries for data manipulation are all written in C#. The core of its geometry engine is written in C++ [49, 50];
- **IFC ++**: an open-source (MIT license) IFC implementation for C++. It was originally developed at the Bauhaus University Weimar [51];
- **IFC Engine**: a paid software that is able to load, edit and create Step Physical Files (as well as the XML notation) and their schema's via its own object database. This includes all currently available IFC versions [15].

The final choice was for the IfcOpenShell library [47], which satisfies the requirements desired for this thesis. These include: being open-source, allowing parsing of most of the available IFC files (mainly 2x3 and 4x1), allowing the extraction of elements geometry to a B-Rep format easy to integrate with other 3D graphic kernel libraries. Another advantage of this library is the presence of a Python module which allows fast prototyping of algorithms using Jupyter notebooks [52].

3D Graphic Kernel library

This work uses OpenCasCade Technology (OCCT), an open-source CAD kernel designed with reference to the STEP standard ISO-10303-42. OCCT includes a set of C++ class libraries providing services for 3D surface and solid modelling, visualization, data exchange and rapid application development [53]. This work uses also s Python wrapper for the C++ OCCT library to integrate better with the rest of the code; the repository is called PythonOCC [54]. It is possible to implement all the work done in this thesis in C++ as well, but using this language would have increased the development time. In order to create a production-ready solution, faster to execute, it can be considered to rewrite all the code in C++ to use the OCCT library without a wrapper.

MIP modelling

Python-MIP is a collection of Python tools for modelling and solving Mixed-Integer Programming (MIP) [55]. With Python-MIP, it is possible to define variables, constraints and the objective function using the high-level syntax of Python with their classes to manage the model. An advantage of Python-MIP is the ability to use different solvers with the same model formulation.

Mathematical programming solver

Python-MIP comes with the open-source COIN-OR CBC [56] programming solver. However, after testing it for some instances of the problem, it was decided to switch to the proprietary GUROBI [57] solver. GUROBI is the leading optimization software and is recognized as the current state of the art in terms of performance when optimizing a MIP problem. This was confirmed by testing some instances of this problem, especially with the increasing dimensions. A brief comparison of the execution time for optimization with the two solvers is given in Table 5.2.

The specific version used in this thesis work was:
`Gurobi Optimizer version 9.5.1 build v9.5.1rc2 (mac64[rosetta2])`.

Other libraries

Other software libraries used are:

- **NetworkX**: Python library for “the creation, manipulation, and study of the structure, dynamics, and functions of complex networks”, used for the graphs [58];
- **pandas**: Python data analysis library, used to manage the optimization input dataframes and .csv files [44];
- **plotly**: Python data visualization library, used to create all the original figures and graphs [59].

5.1.2. Code structure

The code was structured into two main Python Jupyter notebooks [52] and some Python script files containing additional classes and methods. One notebook was used for the IFC processing, and the other was used only for the optimization, using in input the .csv files containing the data, either generated by the IFC processing notebook or with the manual method explained in Section 4.8. The Jupyter notebooks were chosen because PythonOCC offers a 3D renderer based on WebGL that can be used to display entities' geometry from the IFC file quickly. The same goes for the presence of graphing libraries to display inline in the notebook 2D plots and graphs. Figure 5.1 reports a high-level class and module diagram of the code written.

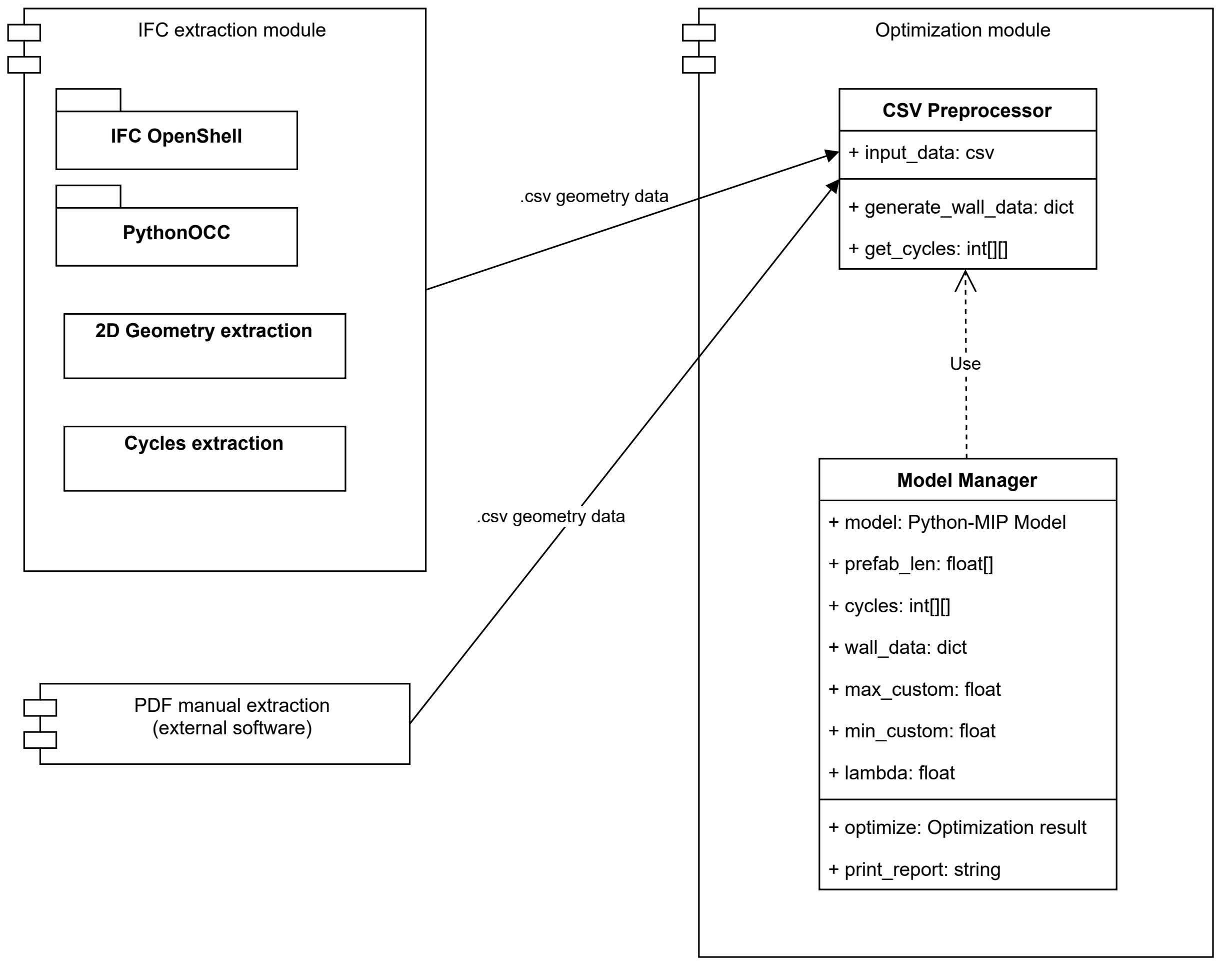


Figure 5.1: *High-level module and class diagram of the structure of the code implemented.*

5.1.3. Hardware used

All the tests and the development of this work were accomplished with an Apple MacBook Pro (16-inch, 2021) with Apple M1 Pro (ARM architecture), 10 physical cores processor, and 16 Gb of memory.

5.2. Optimization model testing

This section shows the results of executing the optimization process described in Chapter 3 with various test cases. The total number of test instances is 8. The input consists of PDF files representing a floor plan of residential housing. The test cases are picked to represent diverse geometries and dimension of floor plans. Since the results are consistent for different cases, it was not needed to use more test instances. The floor plans have been used to extract a list of walls and openings with their dimension. To do so, the author used the method explained in Section 4.8, which consists in manually drawing on top of the PDF files to measure the suitable segments of walls. The subdivision in cycles has been performed manually as well.

The result consists of a .csv table for each floor plan. These tables have the exact structure required for the optimization model, as presented in Section 3.1. The test instances have different types of floor plan geometry. Some are simple rectangular shaped, and others are polygonal, both convex and concave. Figure 5.2 shows some examples of those floor plans. On the left column is a representation of the original PDF floor plan file, and on the right is the graph obtained from the .csv representation of the plan. The openings are in blue, and in red are the standard parts of a wall. It is essential to underline that all testing instances were obtained from floor plans used in construction. This ensures applying the optimization to actual cases and not to artificially generated data. However, Section 7.1 introduces as possible future work the use of augmented or automatically generated input data to perform further tests.

Table 5.1 presents a summary of the main features of the 8 test plans used for this part of the thesis. The number of walls, subwalls and openings for each test case is reported. Other dimensional information includes the total length of the floor's perimeter, the sum of all openings and the non-opening segments.

Naming convention In order to identify each test case, from now on it has been used the following naming convention for the test instances:

$$\texttt{plan\_}XX\texttt{wall\_}YYY\texttt{sub\_}ZZ\texttt{op}$$

			sum length (m)	all walls	openings	non openings
	n. walls	n. subwalls	n. openings			
0	4	52	25	85.49	27.80	57.69
1	6	16	6	56.99	6.40	50.59
2	8	32	13	48.02	16.35	31.67
3	8	36	15	45.89	16.88	29.01
4	10	32	12	60.57	17.27	43.30
5	12	56	21	78.07	23.88	54.19
6	16	44	15	63.72	18.54	45.18
7	68	127	43	176.14	76.23	99.91

Table 5.1: *List of the 8 test instances used, representing floor plans. First column is the index number of the instance.*

Where XX is the number of walls, YYY the number of subwalls, ZZ the number of openings in the test instance.

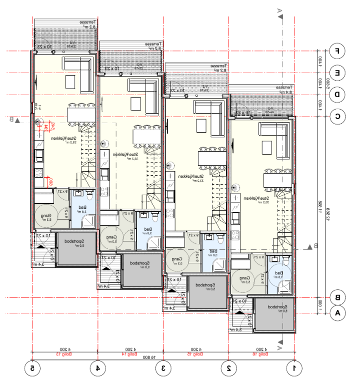
5.2.1. Parameters used for the optimization

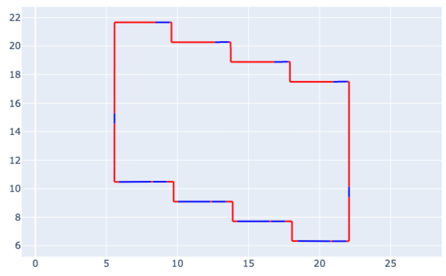
Here follow the list of the parameters used for the tests of the optimization model:

- Database of prefab element lengths: 0.3m, 0.6m;
- Minimum length of custom element: 0.2m;
- Maximum length of custom elements: 1.5m;
- No mandatory corner elements.

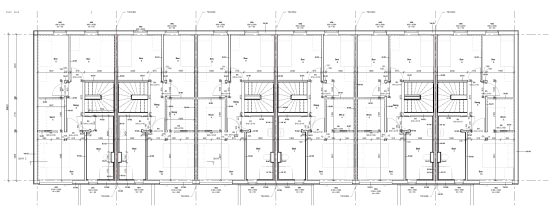
Those values have been chosen to fit the typical constraints of prefabricated wall manufacturers. In particular, those parameters are similar to the one used by the EON Element AS construction system [60], with the only exception of not considering the presence of corner elements here. More information regarding the EON construction system is given in Section 5.3.1.

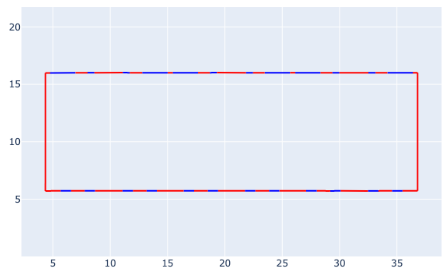
During the development of this thesis, other parameters have been tried. However, the differences in terms of the performance of the model were not significant. Using the model with other parameters from industry constraints gave similar results. Therefore, exploring different construction system parameters is not part of this series of tests.



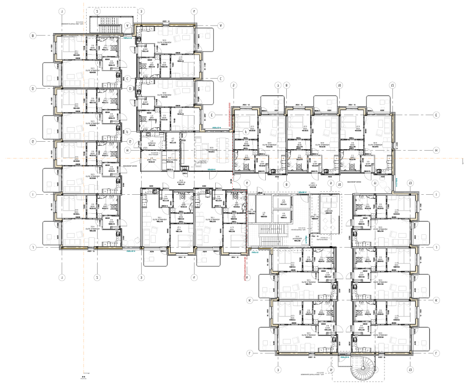


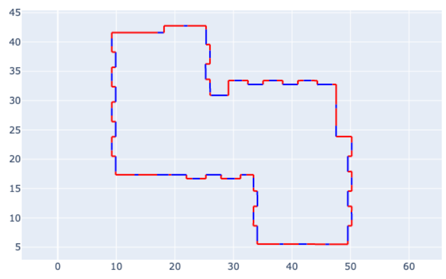
plan_16wall_044sub_15op





plan_04wall_052sub_25op





plan_68wall_127sub_42op

Figure 5.2: *Some examples of floor plans used for the testing. The column on the left reports the original floor plan created by architects and exported in PDF format. On the right column, for each floor plan, its representation of the outer perimeter used for the model testing. The plot is generated from the .csv representation of the floor plan. In blue are represented the positions of the openings; in red are the walls without openings. The floor plans are courtesy of Modulize AS [29].*

5.2.2. Exploration of the Pareto frontier

Having two objectives in the minimization formula, as explained in Section 3.2.4, the parameter λ has been used to weigh the contribution of each of the objectives to minimize. The first analysis explores the Pareto frontier with an approximated method based on plotting a graph with the two objectives. Figure 5.3 reports a graph where: the x-axis describes the total discrepancy of the output (in meters), and the y-axis the cost of the output. Those values have been obtained by running the optimization to all the different test files, repeating the optimization with λ values for 150 evenly-spaced samples of the range $[0 - 1]$. It can be noticed that the scatter plot has the same shape for all the instances.

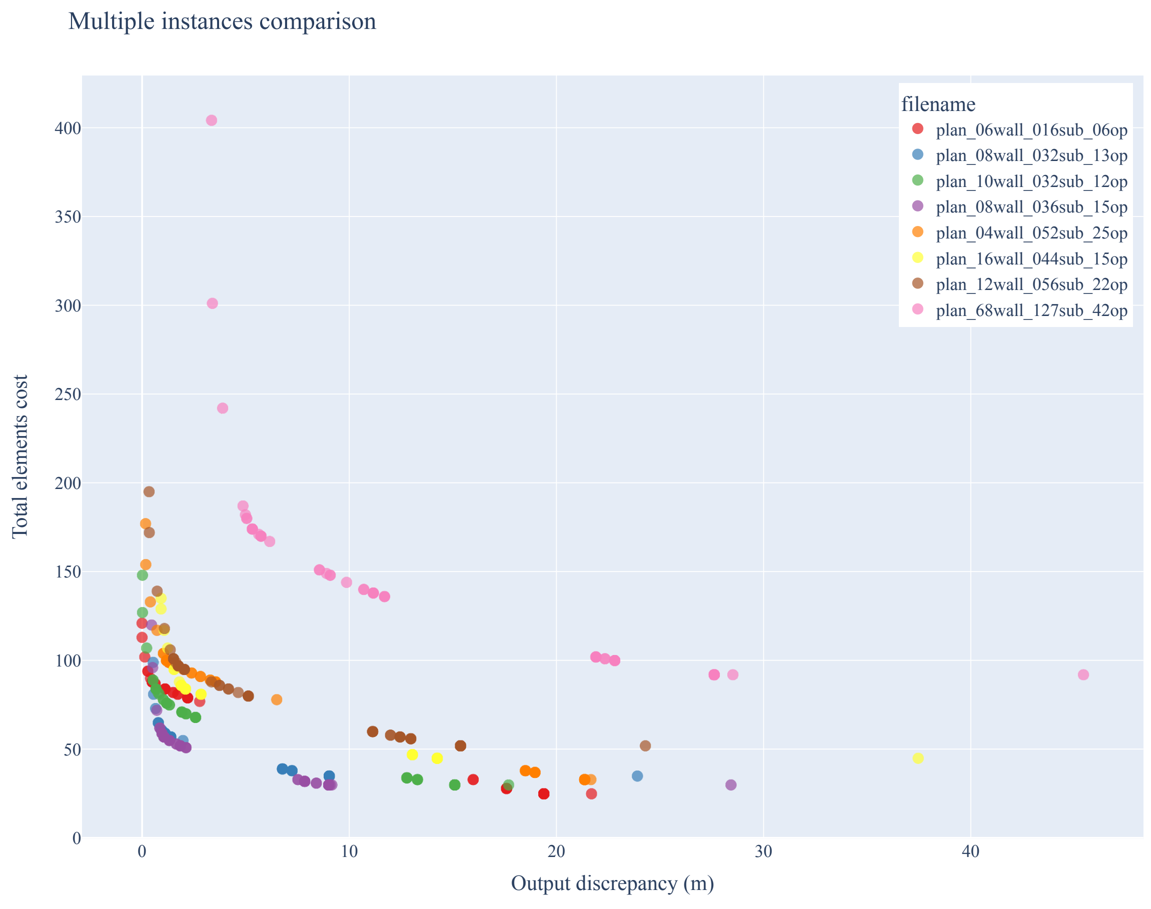


Figure 5.3: *Comparison of the two objective functions for all the test instances. The scatter plot is given for different lambda values.*

The previous graph has been split into different instances to understand better the effect of the choice of the λ parameter. Figure 5.4 shows the approximated Pareto frontier graph, divided by the test instance. In the x-axis is reported the objective function of minimizing the output discrepancy (in meters). In the y-axis, there is the second objective function of minimizing the cost of the solution in terms of a linear combination of the number of standard and custom elements used. The different sub-plots in Figure 5.4 do not have the same scale on the y-axis. This is because the test instances have different dimensions regarding the number of walls and openings. This makes the total cost have different values for different instances. Using not the same spacing in the y axis was done on purpose to provide a normalized view of the cost function. Another aspect to notice is that the cost functions had the same values for multiple lambda values. In other words, points in the scatter plot correspond to different λ values. For this reason, the dots in Figure 5.4 have been plotted with slight transparency to show better the overlaps.

Having multiple points overlapping means that linear changes of the λ parameter are not resulting in linear changes over the values of the two objectives. Figure 5.5 shows on the x-axis the λ parameter and on the y-axis the output discrepancy error (in meters). Looking at the graph, it is evident that the discrepancy error objective function is discrete. Specific λ threshold values can be identified, which change the discrepancy error objective for all instances. The most evident finding is that λ values in the range $\approx [0.1, 0.6]$ produce the same discrepancy in the output.

Number of elements

Since the objective function modelling the cost of the solution is a linear combination of the number of standard and custom elements used, it is relevant to see how different λ values affect the output model in this regard. Figure 5.7 displays the number of standard elements the optimization model picks for each test instance when the λ parameter is varying. In those graphs, the y-axis scale is not the same for all the different instances because those have different dimensions in terms of walls and subwalls. The findings are that a minimum baseline number of elements is needed to create the new geometry. For $\lambda \to 0$, the model is optimizing only to reduce the number of elements, leading to a massive discrepancy in the output, which is represented by the colour scale of the scatter plot. For $\lambda \to 1$, the model is optimizing more to reduce the discrepancy, so the number of standard elements increases, but not linearly. Then for $\lambda \gtrapprox 0.9$, the model starts to reduce the number of standard elements. This reduction can be explained by looking at the dual graphs in Figure 5.8, where the custom elements are on the y-axis. It can be noticed that when $\lambda \gtrapprox 0.9$, the model adds more custom elements to reduce the discrepancy further. Those custom elements are replacing the standard ones and increasing the cost function.

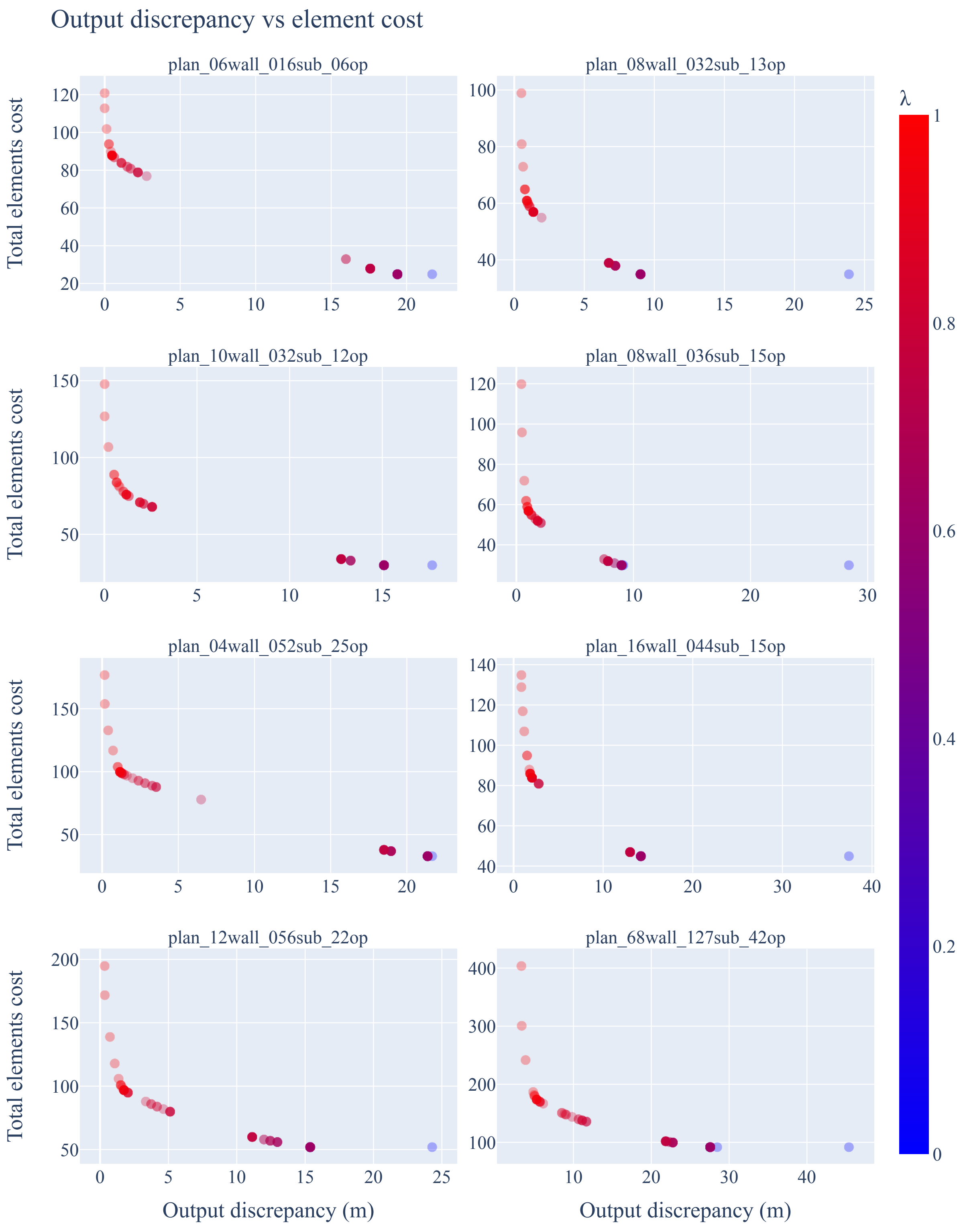


Figure 5.4: *Comparison of the two objective functions for all the test instances, split by each instance. The subplots for instances have different y-axis scale.*

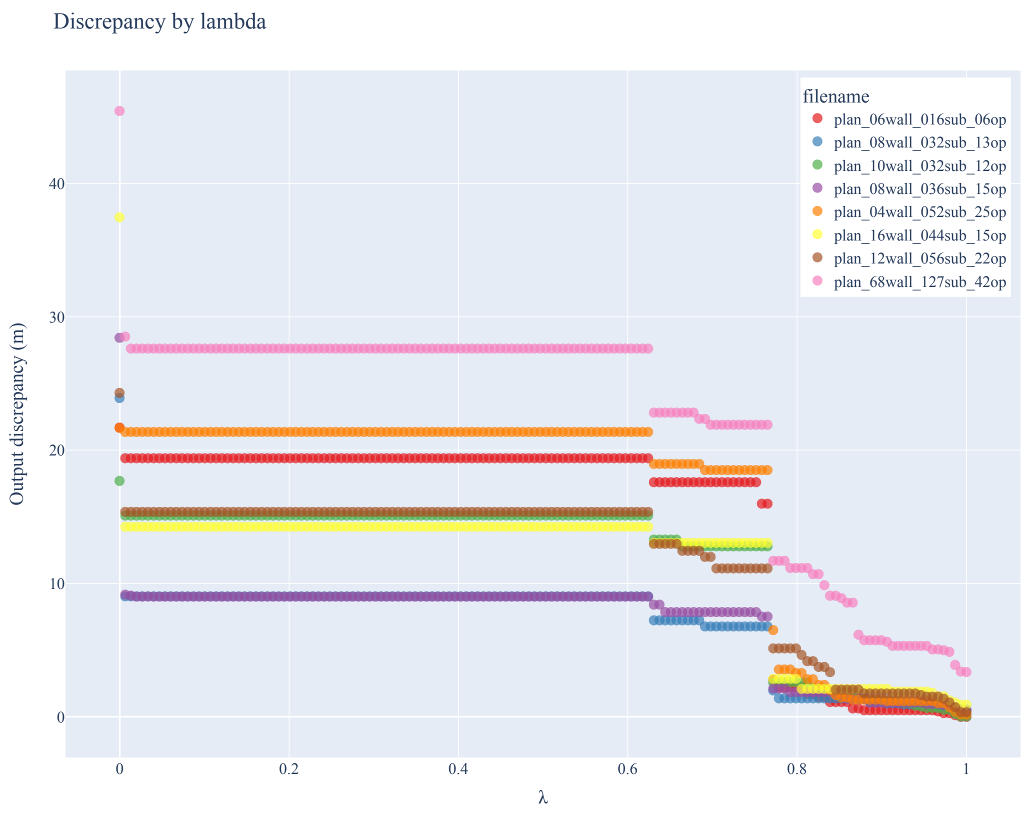


Figure 5.5: *Change of the total discrepancy of the output for different lambda values.*

5.2.3. Comparison of geometries

To conclude the analysis of the effects of the choice of the λ parameter, here follows a graphical comparison of the building geometries.

Figure 5.6 shows the impact of the parameter λ over the geometry of the building. The original geometry and the optimized one are displayed together. In black are the original walls in the input. In blue are the openings, for both the input and the output since the dimension of those is not changing. In red are the prefab elements used in the output. In green are the custom elements used in the output. One test case has been picked, and the optimization was performed for several λ values. Then, only some relevant values have been chosen to be displayed here. As already discussed, for low λ values, the output geometry has a huge discrepancy regarding the input. Whereas for $\lambda \gtrapprox 0.9$, the output discrepancy is almost non-noticeable.

Section 6.1 discusses the possible future improvement of considering the discrepancy in the difference in the area of the two perimeters.

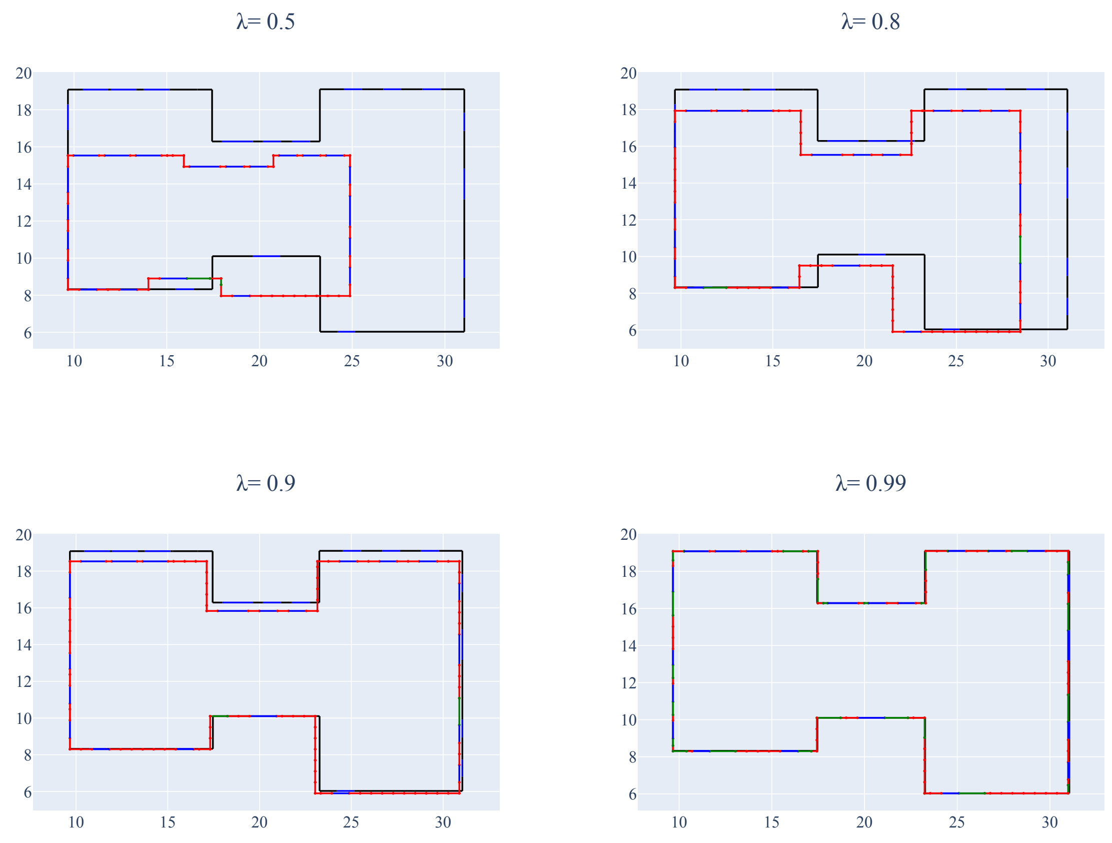


Figure 5.6: *Input vs output floor plan geometry for one specific test instance. In black are the original walls in the input. In blue are the openings, for both the input and the output, since the dimension of those is not changing. In red are the prefab elements used in the output. In green the custom elements used in the output. The axes dimensions of the floor plans are in meters.*

Conclusions on the choice of λ The choice of the λ parameter must be made by the end-users, depending on what aspect is most valued to be optimized. After analyzing the test results, it can be suggested to pick a value close to 0.9, which provided low discrepancy and reasonable cost without needing a massive number of custom elements.

5.2.4. Execution time performance

After running the optimization model with different instances with the GUROBI solver, the performance in terms of execution time has not changed significantly. Whereas, using the CBC solver, it can be noticed that the execution time increases when expanding the model dimensions in the number of rows and columns in input.

Table 5.2 reports a summary of the execution time for the different instances with the two solvers tested.

		optimization time (s)	CBC	Gurobi
filename	model_cols	model_rows		
plan_06wall_016sub_06op	66	46	0.042	0.029
plan_10wall_032sub_12op	126	86	0.056	0.069
plan_08wall_036sub_15op	132	90	0.630	0.010
plan_04wall_052sub_25op	168	114	0.612	0.019
plan_16wall_044sub_15op	180	122	0.086	0.027
plan_12wall_056sub_22op	210	142	0.577	0.156
plan_68wall_127sub_42op	504	338	26.186	0.180

Table 5.2: *Comparison of execution time of optimization for two solvers: Gurobi and CBC. The instances are ordered by increasing model rows. The execution time is in seconds.*

5.3. Case study

This section analyzes one case study, where an IFC building model was optimized using all the methods discussed in this thesis and, in parallel, manually redrawn by an employee of Eon Element AS, a prefab wall manufacturer based in Norway [60].

Figure 5.10 is a graphical visualization of the input building model. The target of this case study is a 2-storey residential family house building. The IFC file was given by gentle concession from Modulize AS [29].

5.3.1. Construction system specifications

EON Element is a patented environmentally friendly building system made of wood elements. The elements are complete wall parts with a framework, wind and vapour barrier and insulation. The elements are assembled with bolts in pre-drilled holes and ready for interior and exterior cladding. It can be approved as a passive house wall [60]. In a passive house, "the heat requirement is so low that a separate heating system is not necessary and there is no loss of comfort" [61]. There are different standards and certifications regarding passive houses, such as the *German Passive House Standard* and the *Norwegian criteria NS3700:2010*. Those certifications aim to establish benchmarks to measure energy efficiency.

In this thesis, the focus is on the geometry of the construction system. The following parameters define the Eon Element AS design:

- Standard wall elements of length either 0.3m or 0.6m;
- Thickness: 0.3 m;
- Height: from 0.3m to 3m in steps of 0.3m;
- Minimum allowed custom element length 0.2m;
- Maximum allowed custom element length 1.5m;
- Corner elements can be inner or outer. Both are 90° corners. The height can be 2.4m, 2.7m or 3m.

The above parameters are similar to those used to perform all the previous tests of the thesis. The presence of corner elements slightly changed the model in the output but without changing the performance. Figure 5.11 shows no significant differences in optimization elapsed time when adding some fixed corners in the output. Automatic corner identification has not been considered part of this work, therefore the corners were manually detected.

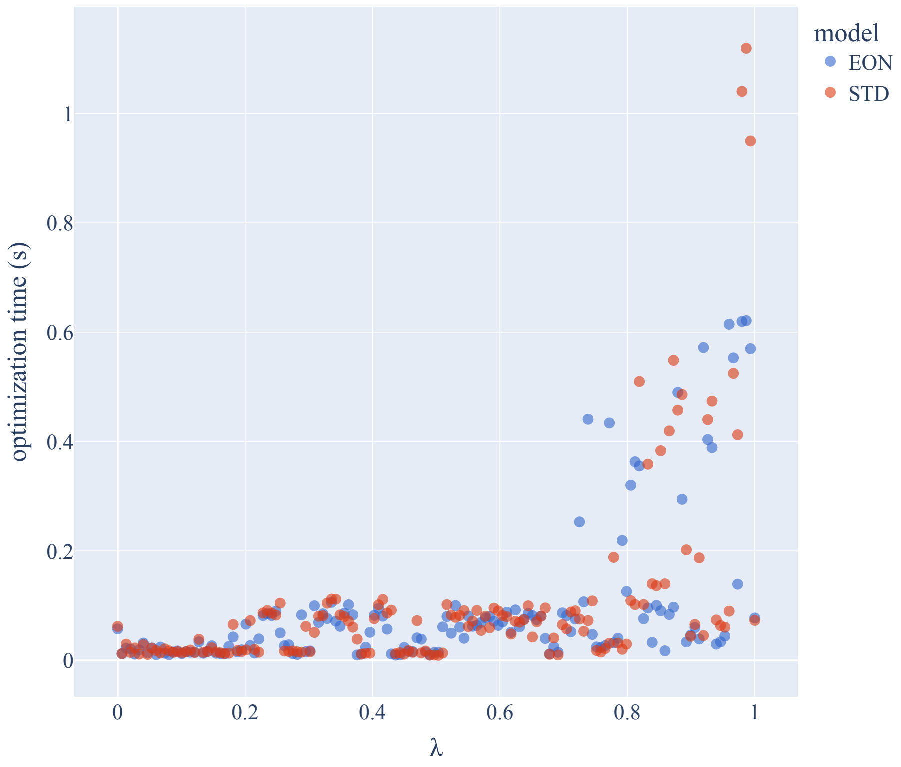


Figure 5.11: *Comparison of execution time of the standard model (STD) and the same with fixed corners (EON).*

5.3.2. Comparison with the redrawing process in industry

The final part of this Chapter compares and validates this thesis method with the current methods adopted in the industry for redrawing building projects. Figure 5.12 shows the steps to manually redraw the case study building geometry using the EON construction system. The manual redraw process uses a CAD tool to create new geometry in 3D. Direct interviews with workers in the industry estimated the average work needed to redraw a medium-sized house to have a minimum time of 1h and range till one day of work. The instance object of this case study was considered “very design-friendly for our system” by the employees of Eon Element and took around one hour of work to redraw.

On the other hand, when using this thesis method, the total redrawing time was about one minute, considering the complete execution of all scripts, both optimization and IFC data extraction. This thesis method produced a 2D view of the redrawing geometry, whereas the drawing obtained by the industry experts consists of a 3D representation. However, a 2D plan contains still all the relevant information regarding the number of prefab elements and which part of the building must be fit. Therefore the manual work performed by EON Elements has been sliced into a floor plan to be compared with the result produced by this thesis approach. A 2D visualization makes it possible to compare the two results, as shown in Figure 5.14.

Regarding how this thesis outputs the results, Listing 42 shows the optimization output for the current case study in text format. It consists of a breakdown for each wall of the input project with a list of elements needed to assemble it. This sort of output is particularly relevant for the manufacturer for two reasons: first can provide information to the factory on how many elements to produce, and secondly, it can be used to produce a very detailed price quote.

```
** For wall #12358 of len 4.355m *************************
    Adding starting corner of len 0.15m
    For subwall #25 of len 4.355m
       Use 7 prefab walls of 0.6m
    Adding ending corner of len 0.15 m
Output len is 4.5 m, discrepancy: 0.145m

** For wall #12473 of len 2.155m *************************
    For subwall #22 of len 0.155m
       Use 1 prefab walls of 0.3m
    Opening 23 keeping 1.0m
    For subwall #24 of len 1.0m
       Use 1 prefab walls of 0.6m
       Use 1 prefab walls of 0.3m
    Adding ending corner of len 0.15 m
Output len is 2.35 m, discrepancy: 0.195m

** For wall #26008 of len 6.21m *************************
    Adding starting corner of len 0.15m
    For subwall #0 of len 0.207m
       Use 1 prefab walls of 0.3m
    Opening 1 keeping 1.5m
    For subwall #2 of len 2.355m
       Use 1 prefab walls of 0.6m
       CUSTOM element of 1.25m
    Opening 3 keeping 1.8m
    For subwall #4 of len 0.348m
       Use 1 prefab walls of 0.3m
    Adding ending corner of len 0.15 m
Output len is 6.05 m, discrepancy: -0.16m

** For wall #30962 of len 3.8m *************************
    Adding starting corner of len 0.15m
    For subwall #5 of len 3.8m
       Use 6 prefab walls of 0.6m
Output len is 3.75 m, discrepancy: -0.05m
...

Total number of custom elem:  2
Standard elem: 7 of 0.3m,  41 of 0.6m, total 48 elements
```

Listing 5.1: *Breakdown of the floor plan into a prefab list of elements. The lengths have been rounded to the third decimal. Some more walls have been omitted for display purposes.*

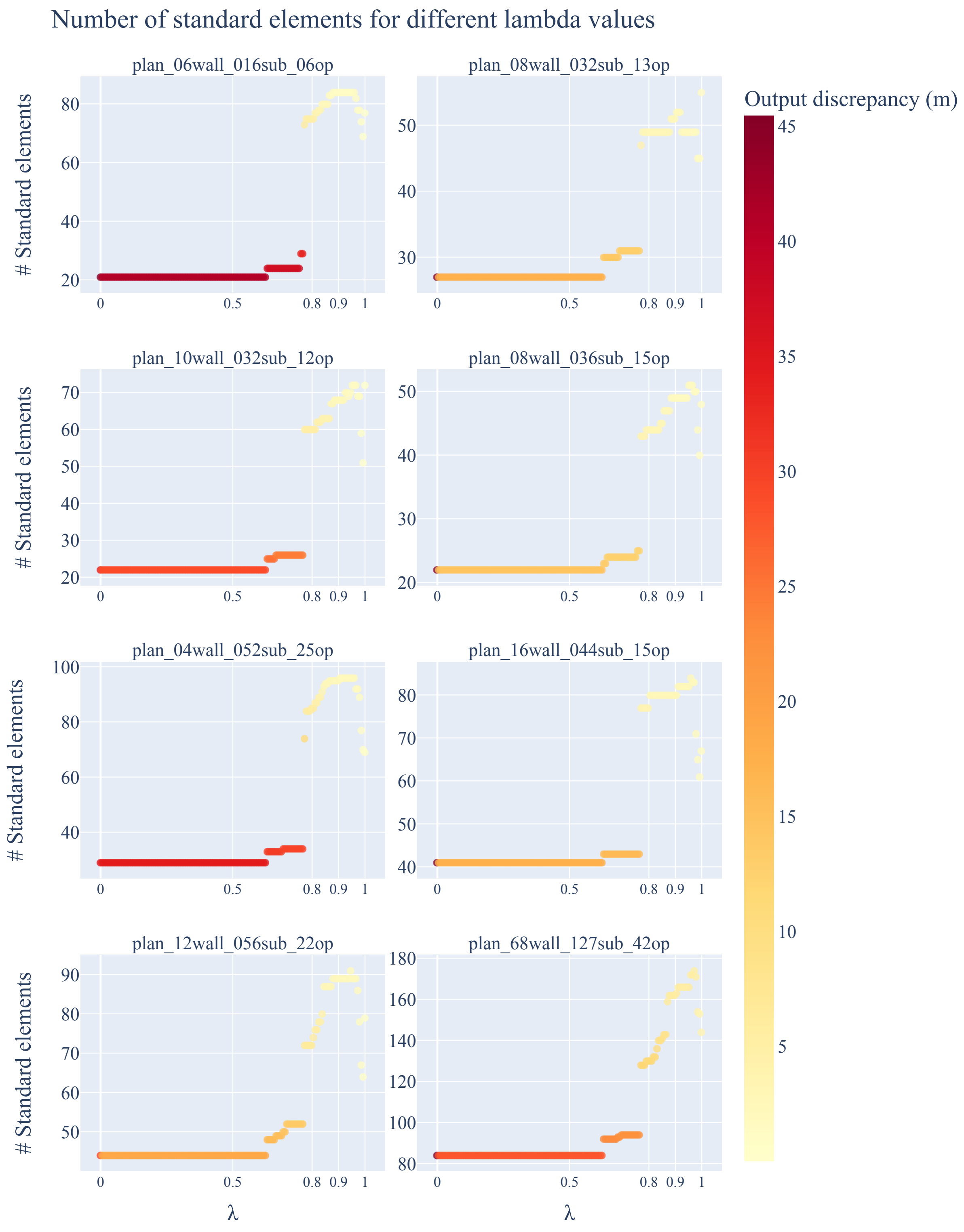


Figure 5.7: *Number of standard elements the optimization model picks, divided by test instance, by varying the λ parameter. The scale on the y-axis for the different subplots is not the same to normalize the result and make them comparable. The colour of the points indicates the total discrepancy of the output.*

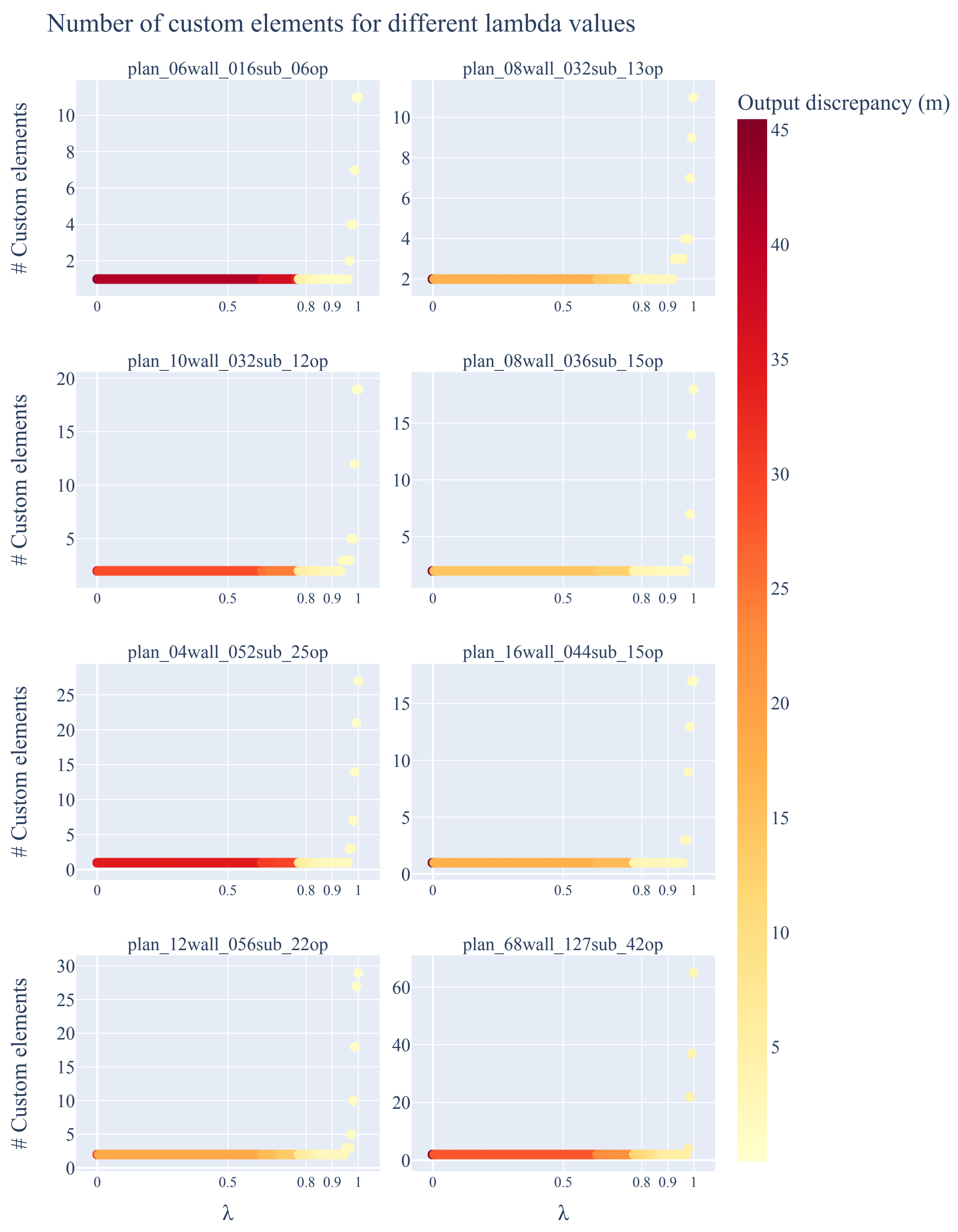


Figure 5.8: *Number of custom elements the optimization model picks, divided by test instance, by varying the λ parameter. The scale on the y-axis for the different subplots is not the same to normalize the result and make them comparable. The colour of the points indicates the total discrepancy of the output.*

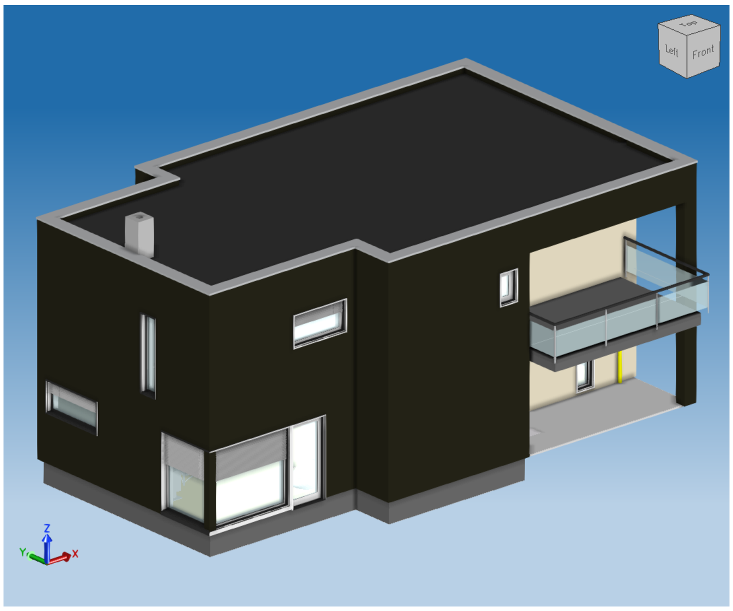


Figure 5.9: *Example of visualization of the case study IFC file, using the open source OpenIFCViewer.*

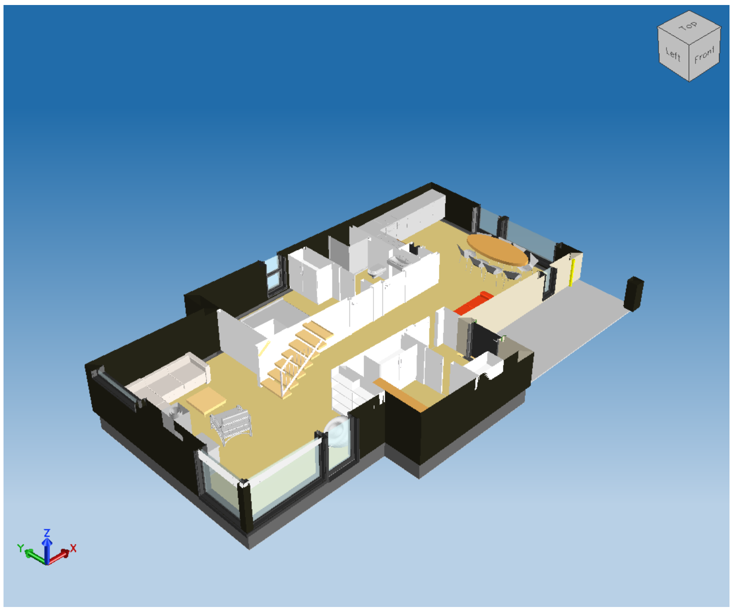


Figure 5.10: *Same model with section over the storey considered in this case study.*

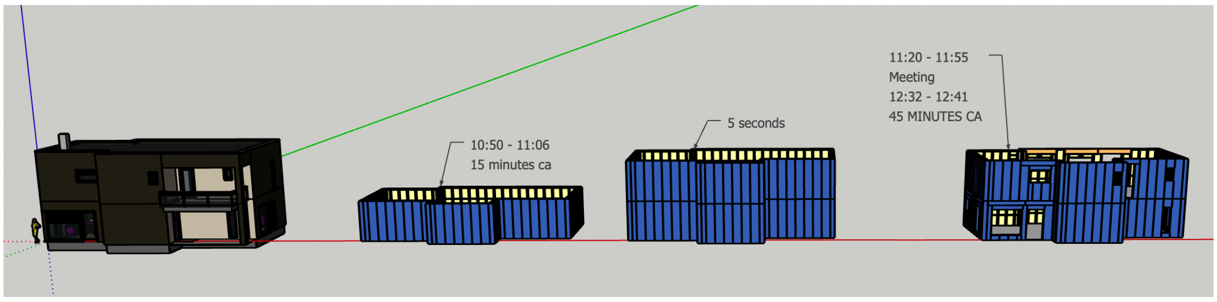


Figure 5.12: *Work breakdown of manual redraw made with a CAD software by employees of Eon Element [60]. Reproduced with permission.*

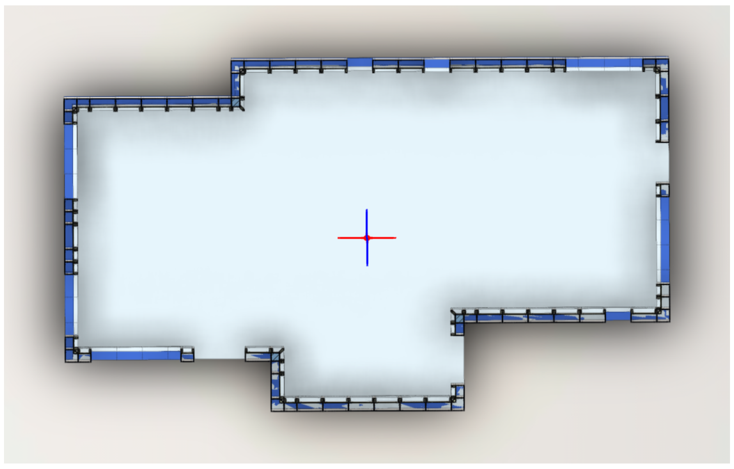


Figure 5.13: *2D floor plan visualization of the output created by EON Elements employees.*

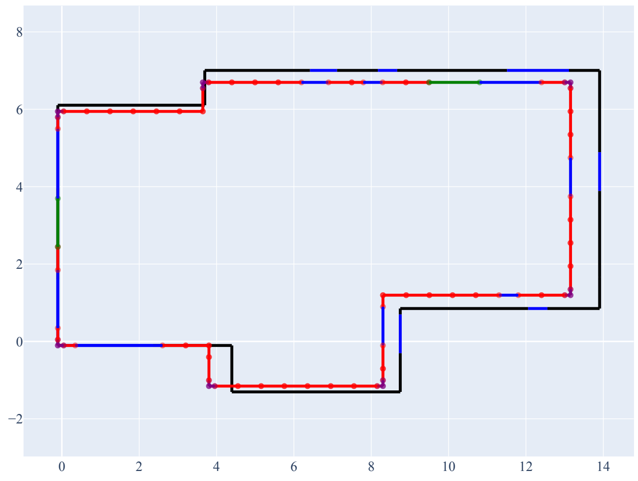


Figure 5.14: *2D floor plan visualization of the output generated with this thesis method. In red are the prefab wall elements used in output. In blue the openings, in green the custom elements and in purple the corners.*

6 | Conclusions

This chapter examines the results of this thesis, possibilities of applications and further improvements.

This thesis aimed to automate the process of redrawing building designs using prefabricated components. The novel approach proposed in the thesis was split into two methods to achieve this goal. The first consisted of a MIP optimization model and the second of a computational geometric extraction process from 3D IFC files.

This thesis results demonstrated that under some assumptions on the geometry of the input, it is possible to build almost any geometry floor plan using standardized elements with slight changes in the original design. The primary assumption was on the orthogonality of joints of walls in the original input design. Moreover, the results were optimized to reduce the cost of manufacturing. This means that a prefab wall manufacturer can reduce the costs of building projects and guarantee limited changes over the original design geometry not intended for prefabrication.

The optimization model was tested using eight test instances and a final case study to assess its performance. The performance evaluated were: total element cost and entire length discrepancy input versus output. Optimization time was analyzed for different solvers and instances. The results were fast in execution, with an average time of a few seconds for the test cases. An approximated subset of the Pareto frontier was given as a scatter plot view of the objective space to deal with the presence of two main objectives.

Finally, a complete example of applying this thesis's method to a building project was compared with the manual process currently used in the industry. The employees of a manufacturing company of prefabricated wall panels considered the results very promising since this thesis could decrease the time spent on the task of redrawing, from hours of human work to minutes of automatic execution of the scripts.

6.1. Future works and improvement

This section discusses some possible ways to improve the current work. The improvements are divided into two parts, one relative to the optimization model and one to the geometric processing from IFC files.

6.1.1. MIP model

Increasing the scope of the model The model presented in this thesis was developed with some prerequisites that made it simpler to handle in the first instance. Relaxing some of the requisites on the input can be helpful to scale the model to be adapted to more building projects. The model's first and most significant expansion can be allowing walls not orthogonal to each other. This is because some building drawings use different angles of wall junctions which are not always 90 degrees. The subdivision into cycles of wall directions must therefore be updated to consider the freedom in the connection angle between walls.

Secondly, this thesis assumed that the project opening dimensions were correct and corresponded to the actual dimensions of the doors and windows to be installed in the building. However, it can happen to have input data where the opening dimensions are incorrect or do not fit with the requirements of the manufacturing with prefabrication. In this case, if a database of opening elements is also available, the model can be expanded by adding a decision variable to replace the input opening with an opening from the database.

Finally, this thesis did not consider the position of the elements in each output subwalll because it was not relevant for the on-site assembly. However, some construction systems could have added constraints in positioning some elements. For example, Eon Element uses particular corner elements, distinguishing inner and outer corners. Expanding the model to work with other manufacturers' constraints is a possible future work.

Alternative constraints In this thesis the main constraints on the geometry are the cycle ones, used to preserve the closed polygonal In this thesis, the main constraints on the geometry are the cycle ones used to preserve the closed polygonal perimeter of the walls. Then, one objective is to minimize the discrepancy of the output redrawn model compared to the input one. However, this discrepancy can make the output building project too big to fit the requirements of a construction site. Therefore, some property developers might want to add a constraint on the maximum allowed area of a building. This necessity comes because the terrain's dimensions can have specific limits, which vary for each project. To avoid adding a non-linear constraint, like the area calculation, this can be simplified, splitting it down into two constraints, one for each of the two dimensions of the floor plan of the building. When adding the above constraints, knowledge of the land area measurements is necessary.

More detailed cost estimation This thesis assumed that the cost of elements is unitary and the cost of custom elements to be four times the standard ones, as suggested by domain experts of one manufacturing company. When dealing with off-site construction, an essential part of the cost to be considered is the cost of transporting elements. This cost heavily depends on the site's location and other factors that were out of the scope of this thesis. Therefore, knowing an exact cost function for each element in the prefab database can better estimate the cost function to minimize.

The output discrepancy was not counted in monetary cost in this thesis work. If given a cost function per square meter of the increased surface of a building, the discrepancy could be better used in the objective function to make it homogeneous.

Testing This thesis method was tested with eight real building projects. The test cases were picked to represent diverse geometries and dimensions of floor plans. The results were consistent; consequently, it was not needed to use more test instances. However, to test the limits of the model, it can be considered as future work the implementation of automatic generation of geometries as test cases.

Furthermore, this thesis used an approximated method to explore the Pareto frontier. A possible future work can include obtaining an exact Pareto frontier using a different solver.

6.1.2. IFC processing

This thesis provided a method to automate the extraction of quantities and geometry data from an IFC file. The IFC files have been carefully picked during this thesis work to satisfy all the requirements for good quality. However, as discussed in 2.2, IFC files are not well standardized, which introduces the need to consider many edge cases. Dealing with malformed files can lead to complete automation of this task in the future.

Future works can investigate the addition of pre-processing of IFC files to create a pipeline of automated checking of the designs before applying the method for geometric extraction given in this thesis. Examples of automation tasks include adding a collision checking phase, classifying IFC entities, and automatic wall splitting.

6.2. Possibilities of application

The software developed during this thesis work is considered a significant step forward in automating the redrawing process for the prefabrication of buildings. Direct interviews with experts in the field of prefab wall manufacturing evaluated this work as a promising step toward automating the redrawing task they currently need to perform entirely manually.

This thesis is part of a project Modulize AS is currently bringing forward to help manufacturers of prefab element to achieve their full potential creating a software solution for offsite construction, optimizing design, planning, and procurement.

Bibliography

[1] A. G. F. Gibb, “Standardization and pre-assembly- distinguishing myth from reality using case study research,” *Construction Management and Economics*, vol. 19, no. 3, pp. 307–315, Apr. 2001, ISSN: 0144-6193, 1466-433X. DOI: 10.1080/01446190010020435. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/01446190010020435 (visited on 05/05/2022).

[2] B. Ginigaddara, S. Perera, Y. Feng, and P. Rahnamayiezekavat, “Typologies of offsite construction,” 2019, Accepted: 2019-10-31T04:21:21Z. DOI: 10.31705/WCS.2019.56. [Online]. Available: http://dl.lib.uom.lk/handle/123/15373 (visited on 04/14/2022).

[3] N. Bertram, S. Fuchs, J. Mischke, R. Palter, G. Strube, and J. Woetzel, “Modular construction: From projects to products,” McKinsey & Company, Jun. 2019, p. 34. [Online]. Available: https://www.mckinsey.com/business-functions/operations/our-insights/modular-construction-from-projects-to-products.

[4] S. Lehmann, “Low carbon construction systems using prefabricated engineered solid wood panels for urban infill to significantly reduce greenhouse gas emissions,” *Sustainable Cities and Society*, vol. 6, pp. 57–67, Feb. 2013, ISSN: 22106707. DOI: 10.1016/j.scs.2012.08.004. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S221067071200056X (visited on 04/13/2022).

[5] D. Steinhardt, K. Manley, L. Bildsten, and K. Widen, “The structure of emergent prefabricated housing industries: A comparative case study of australia and sweden,” *Construction Management and Economics*, vol. 38, no. 6, pp. 483–501, Jun. 2, 2020, ISSN: 0144-6193, 1466-433X. DOI: 10.1080/01446193.2019.1588464. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/01446193.2019.1588464 (visited on 05/05/2022).

[6] C. Rausch, “Algorithms for geometric optimization and enrichment in industrialized building construction,” Ph.D. dissertation, University of Waterloo, Mar. 1, 2021.

[7] T. Salama, O. Moselhi, and M. Al-Hussein, “Overview of the characteristics of the modular industry and barriers to its increased market share,” *International Journal of Industrialized Construction*, vol. 2, no. 1, pp. 30–53, Jul. 30, 2021, Num-

ber: 1, ISSN: 2563-5034. DOI: 10.29173/ijic249. [Online]. Available: https://journalofindustrializedconstruction.com/index.php/jic/article/view/249 (visited on 05/11/2022).

[8] N. Liang and M. Yu, "Research on design optimization of prefabricated residential houses based on BIM technology," *Scientific Programming*, vol. 2021, e1422680, Nov. 9, 2021, Publisher: Hindawi, ISSN: 1058-9244. DOI: 10.1155/2021/1422680. [Online]. Available: https://www.hindawi.com/journals/sp/2021/1422680/ (visited on 05/04/2022).

[9] F. Pittau, L. E. Malighetti, G. Iannaccone, and G. Masera, "Prefabrication as large-scale efficient strategy for the energy retrofit of the housing stock: An italian case study," *Procedia Engineering*, vol. 180, pp. 1160–1169, 2017, ISSN: 18777058. DOI: 10.1016/j.proeng.2017.04.276. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1877705817317836 (visited on 04/13/2022).

[10] I. Y. Wuni and G. Q. P. Shen, "Holistic review and conceptual framework for the drivers of offsite construction: A total interpretive structural modelling approach," *Buildings*, vol. 9, no. 5, p. 117, May 2019, Number: 5 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2075-5309. DOI: 10.3390/buildings9050117. [Online]. Available: https://www.mdpi.com/2075-5309/9/5/117 (visited on 05/28/2022).

[11] S. Kubba, "Chapter 5 - building information modeling," in *Handbook of Green Building Design and Construction*, S. Kubba, Ed., Boston: Butterworth-Heinemann, Jan. 1, 2012, pp. 201–226, ISBN: 978-0-12-385128-4. DOI: 10.1016/B978-0-12-385128-4.00005-6. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780123851284000056 (visited on 05/26/2022).

[12] T. D. Oesterreich and F. Teuteberg, "Behind the scenes: Understanding the socio-technical barriers to BIM adoption through the theoretical lens of information systems research," *Technological Forecasting and Social Change*, vol. 146, pp. 413–431, Sep. 1, 2019, ISSN: 0040-1625. DOI: 10.1016/j.techfore.2019.01.003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0040162518304852 (visited on 05/26/2022).

[13] ISO, "16739-1: 2018: Industry foundation classes (IFC) for data sharing in the construction and facility management industries—part 1: Data schema," *International Organisation for Standardisation: Geneva, Switzerland*, 2018.

[14] buildingSMART. "Industry foundation classes (IFC)," buildingSMART International. (2022), [Online]. Available: https://www.buildingsmart.org/standards/bsi-standards/industry-foundation-classes/ (visited on 05/26/2022).

[15] *IFC engine.* [Online]. Available: `http://rdf.bg/product-list/ifc-engine/` (visited on 05/07/2022).

[16] ISO, "ISO 10303-21:2016," 2016. [Online]. Available: `https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/06/31/63141.html` (visited on 04/14/2022).

[17] A. Borrmann, J. Beetz, C. Koch, T. Liebich, and S. Muhic, "Industry foundation classes: A standardized data model for the vendor-neutral exchange of digital building models," in *Building Information Modeling: Technology Foundations and Industry Practice*, A. Borrmann, M. König, C. Koch, and J. Beetz, Eds., Cham: Springer International Publishing, 2018, pp. 81–126, ISBN: 978-3-319-92862-3. DOI: `10.1007/978-3-319-92862-3_5`. [Online]. Available: `https://doi.org/10.1007/978-3-319-92862-3_5` (visited on 05/26/2022).

[18] J. Zhu, P. Wu, M. Chen, M. J. Kim, X. Wang, and T. Fang, "Automatically processing IFC clipping representation for BIM and GIS integration at the process level," *Applied Sciences*, vol. 10, no. 6, p. 2009, Mar. 15, 2020, ISSN: 2076-3417. DOI: `10.3390/app10062009`. [Online]. Available: `https://www.mdpi.com/2076-3417/10/6/2009` (visited on 04/13/2022).

[19] I. Stroud and H. Nagy, *Solid modelling and CAD systems: how to survive a CAD system.* London ; New York: Springer, 2011, 689 pp., OCLC: ocn701813173, ISBN: 978-0-85729-258-2 978-0-85729-259-9.

[20] H. Liu, M. Lu, and M. Al-Hussein, "Ontology-based semantic approach for construction-oriented quantity take-off from BIM models in the light-frame building industry," *Advanced Engineering Informatics*, vol. 30, no. 2, pp. 190–207, Apr. 2016, ISSN: 14740346. DOI: `10.1016/j.aei.2016.03.001`. [Online]. Available: `https://linkinghub.elsevier.com/retrieve/pii/S1474034616300246` (visited on 06/18/2022).

[21] M. Venugopal, C. M. Eastman, and J. Teizer, "An ontology-based analysis of the industry foundation class schema for building information model exchanges," *Advanced Engineering Informatics*, vol. 29, no. 4, pp. 940–957, Oct. 2015, ISSN: 14740346. DOI: `10.1016/j.aei.2015.09.006`. [Online]. Available: `https://linkinghub.elsevier.com/retrieve/pii/S1474034615001019` (visited on 04/13/2022).

[22] T. Krijnen and M. Tamke, "Assessing implicit knowledge in BIM models with machine learning," Jan. 1, 2015, pp. 397–406, ISBN: 978-3-319-24206-4. DOI: `10.1007/978-3-319-24208-8_33`.

[23] F. C. Collins, A. Braun, M. Ringsquandl, D. M. Hall, and A. Borrmann, "Assessing IFC classes with means of geometric deep learning on different graph encodings," presented at the 2021 European Conference on Computing in Construction, Jul. 26,

2021, pp. 332–341. DOI: 10.35490/EC3.2021.168. [Online]. Available: https://ec-3.org/publications/conferences/2021/paper/?id=168 (visited on 04/13/2022).

[24] S. Gijezen, T. Hartmann, K. T. Veenvliet, H. Hendriks, and N. Buursema, "Organizing 3d building information models with the help of work breakdown structures to improve the clash detection process,"

[25] S. Daum and A. Borrmann, "Checking spatio-semantic consistency of building information models by means of a query language," *Proceedings of the 13th International Conference on Construction Applications of Virtual Reality*, 2013. [Online]. Available: https://www.semanticscholar.org/paper/Checking-spatio-semantic-consistency-of-building-by-Daum-Borrmann/3475f6bf35d010cff6c73fa8424e15525364137f (visited on 05/24/2022).

[26] S. Daum and A. Borrmann, "Processing of topological BIM queries using boundary representation based methods," *Advanced Engineering Informatics*, vol. 28, no. 4, pp. 272–286, Oct. 2014, ISSN: 14740346. DOI: 10.1016/j.aei.2014.06.001. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1474034614000391 (visited on 05/24/2022).

[27] J. Luttun and T. F. Krijnen, "An approach for data extraction, validation and correction using geometrical algorithms and model view definitions on building models," *Proceedings of the 18th International Conference on Computing in Civil and Building Engineering (ICCCBE 2020)*, vol. 98, 2020, Publisher: Springer, ISSN: 2366-2557. DOI: 10.1007/978-3-030-51295-8_38. [Online]. Available: https://repository.tudelft.nl/islandora/object/uuid%3Af26ec1dc-5cac-42e0-81e2-0ae84cb6d128 (visited on 04/14/2022).

[28] D. Jankovics, "Customized topology optimization for additive manufacturing," Ph.D. dissertation, Aug. 1, 2019.

[29] "Modulize AS." (2022), [Online]. Available: https://www.modulize.io/ (visited on 04/14/2022).

[30] V. Machairas, A. Tsangrassoulis, and K. Axarli, "Algorithms for optimization of building design: A review," *Renewable and Sustainable Energy Reviews*, vol. 31, pp. 101–112, Mar. 1, 2014, ISSN: 1364-0321. DOI: 10.1016/j.rser.2013.11.036. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364032113007855 (visited on 06/13/2022).

[31] D. Yang, S. Ren, M. Turrin, S. Sariyildiz, and Y. Sun, "Multi-disciplinary and multi-objective optimization problem re-formulation in computational design exploration: A case of conceptual sports building design," *Automation in Construction*, vol. 92, pp. 242–269, Aug. 1, 2018, ISSN: 0926-5805. DOI: 10.1016/j.autcon.2018.03.023.

[Online]. Available: https://www.sciencedirect.com/science/article/pii/S0926580517309317 (visited on 05/23/2022).

[32] H. Liu, B. Holmwood, C. Sydora, G. Singh, and M. Al-Hussein, "Optimizing multi-wall panel configuration for panelized construction using BIM," *Proceedings of International Structural Engineering and Construction*, vol. 4, no. 1, E. Pellicer, J. M. Adam, V. Yepes, A. Singh, and S. Yazdani, Eds., Jul. 2017, ISSN: 2644-108X. DOI: 10.14455/ISEC.res.2017.15. [Online]. Available: https://www.isec-society.org/ISEC_PRESS/ISEC_09/html/C-19.xml (visited on 04/13/2022).

[33] W. Lu, C. Webster, K. Chen, X. Zhang, and X. Chen, "Computational building information modelling for construction waste management: Moving from rhetoric to reality," *Renewable and Sustainable Energy Reviews*, vol. 68, pp. 587–595, Feb. 1, 2017, ISSN: 1364-0321. DOI: 10.1016/j.rser.2016.10.029. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364032116306797 (visited on 05/04/2022).

[34] M. Zawidzki and J. Szklarski, "Multi-objective optimization of the floor plan of a single story family house considering position and orientation," *Advances in Engineering Software*, vol. 141, p. 102 766, Mar. 2020, ISSN: 09659978. DOI: 10.1016/j.advengsoft.2019.102766. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0965997819301310 (visited on 04/13/2022).

[35] W. Para, P. Guerrero, T. Kelly, L. Guibas, and P. Wonka, "Generative layout modeling using constraint graphs," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, Montreal, QC, Canada: IEEE, Oct. 2021, pp. 6670–6680, ISBN: 978-1-66542-812-5. DOI: 10.1109/ICCV48922.2021.00662. [Online]. Available: https://ieeexplore.ieee.org/document/9710899/ (visited on 04/13/2022).

[36] C. Khosakitchalert, N. Yabuki, and T. Fukuda, "BIM-based wall framing calculation algorithms for detailed quantity takeoff," Nov. 8, 2019.

[37] Z. Xu, J. Abualdenien, H. Liu, and R. Kang, "An IDM-based approach for information requirement in prefabricated construction," *Advances in Civil Engineering*, vol. 2020, e8946530, Jan. 27, 2020, Publisher: Hindawi, ISSN: 1687-8086. DOI: 10.1155/2020/8946530. [Online]. Available: https://www.hindawi.com/journals/ace/2020/8946530/ (visited on 06/04/2022).

[38] E. Tauscher, H.-J. Bargstädt, and K. Smarsly, "Generic BIM queries based on the IFC object model using graph theory," p. 8,

[39] E. M. L. Beale, "Integer programming," in *Computational Mathematical Programming*, K. Schittkowski, Ed., ser. NATO ASI Series, Berlin, Heidelberg: Springer, 1985, pp. 1–24, ISBN: 978-3-642-82450-0. DOI: 10.1007/978-3-642-82450-0_1.

[40] D. Bienstock and G. L. Nemhauser, Eds., *Integer programming and combinatorial optimization: 10th International IPCO Conference, New York, NY, USA, June 7-11, 2004: proceedings*, Lecture notes in computer science 3064, Meeting Name: Conference on Integer Programming and Combinatorial Optimization, Berlin ; New York: Springer-Verlag, 2004, 443 pp., ISBN: 978-3-540-22113-5.

[41] K. Deb and K. Deb, "Multi-objective optimization," in *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, E. K. Burke and G. Kendall, Eds., Boston, MA: Springer US, 2014, pp. 403–449, ISBN: 978-1-4614-6940-7. DOI: `10.1007/978-1-4614-6940-7_15`. [Online]. Available: `https://doi.org/10.1007/978-1-4614-6940-7_15` (visited on 05/28/2022).

[42] "Multi-objective optimization," in *Multi-objective Management in Freight Logistics: Increasing Capacity, Service Level and Safety with Optimization Algorithms*, M. Caramia and P. Dell'Olmo, Eds., London: Springer, 2008, pp. 11–36, ISBN: 978-1-84800-382-8. DOI: `10.1007/978-1-84800-382-8_2`. [Online]. Available: `https://doi.org/10.1007/978-1-84800-382-8_2` (visited on 05/28/2022).

[43] A. A. Khan, A. Donaubauer, and T. H. Kolbe, "A multi-step transformation process for automatically generating indoor routing graphs from existing semantic 3d building models," p. 20,

[44] J. Reback, Jbrockmendel, W. McKinney, J. Van Den Bossche, T. Augspurger, M. Roeschke, S. Hawkins, P. Cloud, Gfyoung, Sinhrks, P. Hoefler, A. Klein, T. Petersen, J. Tratner, C. She, W. Ayd, S. Naveh, J. Darbyshire, M. Garcia, R. Shadrach, J. Schendel, A. Hayden, D. Saxton, M. E. Gorelli, F. Li, M. Zeitlin, V. Jancauskas, A. McMaster, T. Wörtwein, and P. Battiston, *Pandas-dev/pandas: Pandas 1.4.2*, version v1.4.2, Apr. 2, 2022. DOI: `10.5281/ZENODO.3509134`. [Online]. Available: `https://zenodo.org/record/3509134` (visited on 06/21/2022).

[45] H. Hecht and Š. Jaud, "TUM OpenInfraPlatform: The open-source BIM visualisation software," p. 8, 2019.

[46] E. Valero, D. D. Mohanty, and F. Bosche, "Development of an open-source scan+BIM platform," presented at the 37th International Symposium on Automation and Robotics in Construction, Kitakyushu, Japan, Oct. 14, 2020. DOI: `10.22260/ISARC2020/0033`. [Online]. Available: `http://www.iaarc.org/publications/2020_proceedings_of_the_37th_isarc/development_of_an_open_source_scanbim_platform.html` (visited on 05/07/2022).

[47] *IfcOpenShell.* [Online]. Available: `https://github.com/IfcOpenShell/IfcOpenShell` (visited on 06/22/2022).

[48] A. Viegas. "IFC.js." (2022), [Online]. Available: `https://IFCjs.github.io/info/` (visited on 05/07/2022).

[49] *Xbim toolkit.* [Online]. Available: `https://docs.xbim.net/` (visited on 05/07/2022).

[50] S. Lockley, C. Benghi, and M. Černý, "Xbim.essentials: A library for interoperable building information applications," *Journal of Open Source Software*, vol. 2, no. 20, p. 473, Dec. 7, 2017, ISSN: 2475-9066. DOI: `10.21105/joss.00473`. [Online]. Available: `https://joss.theoj.org/papers/10.21105/joss.00473` (visited on 05/07/2022).

[51] *IFC++*, original-date: 2015-04-01T13:53:19Z, Jun. 21, 2022. [Online]. Available: `https://github.com/ifcquery/ifcplusplus` (visited on 06/22/2022).

[52] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, C. Willing, and J. d. team, "Jupyter notebooks - a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, F. Loizides and B. Scmidt, Eds., Netherlands: IOS Press, 2016, pp. 87–90. [Online]. Available: `https://eprints.soton.ac.uk/403913/`.

[53] *Open CASCADE technology - open cascade.* [Online]. Available: `https://www.opencascade.com/open-cascade-technology/` (visited on 04/14/2022).

[54] T. Paviot, *pythonOCC*, May 1, 2022. [Online]. Available: `https://github.com/tpaviot/pythonocc` (visited on 05/04/2022).

[55] H. G. Santos, T. Toffolo, Tommy, S. Spoorendonk, P. Larsen, M. Jurasovic, S. Brito, S. Heger, A. Marvin, F. Bonelli, I. Larchenko, L. Singer, Q. Fortier, S. Vigerske, X. Chen, luvik, tekgrizzly, A. Phillips, D. Zhou, brmanuel, P. S. Lopes, P. Lietz, R. P. A, and pabloazurduy, *Coin-or/python-mip: 1.14.0*, Jun. 17, 2022. DOI: `10.5281/zenodo.6657233`. [Online]. Available: `https://zenodo.org/record/6657233` (visited on 06/22/2022).

[56] *Cbc.* [Online]. Available: `https://github.com/coin-or/Cbc` (visited on 05/07/2022).

[57] *Gurobi optimizer reference manual.* [Online]. Available: `https://www.gurobi.com/` (visited on 05/07/2022).

[58] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using NetworkX," in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11–15.

[59] *Collaborative data science*, 2015. [Online]. Available: `https://plot.ly`.

[60] "EON element AS." (2022), [Online]. Available: `https://www.eonelement.com/en/` (visited on 05/30/2022).

[61] W. Feist, "Life-cycle energy analysis: Low-energy house, passive house, self-sufficient house," *International Symposium of CIB*, pp. 183–190, Jan. 1, 1997.

List of Figures

List of Tables

Acronyms

AEC Architecture, Engineering and Construction. 3, 5, 9, 10

BIM Building Information Modeling. 3, 9, 19, 46

B-rep boundary representation. 13

CAD Computer-Aided Design. 3, 19

CLT Cross-laminated timber. 6

IFC Industry Foundation Classes. 3, 10, 13, 20, 35–38, 40, 41, 44, 46, 77

LGPL GNU Lesser General Public License. 46

MIP Mixed-Integer Programming. 3, 27, 47

OSC off-site construction. 2, 5, 6, 8, 9

Acknowledgements

The author thanks the advisors, Prof. Federico Malucelli and Prof. Pietro Luigi Belotti, for their constant presence at meetings to discuss and refine the optimization model and the computational geometry part.
The author thanks Modulize AS for providing the database of IFC files and PDF floor plans used for this thesis and for providing the initial thesis question topic.
The author thanks Eon Element AS for giving detailed information regarding their construction system and for validating the thesis work with their approach.
Finally, the author is thankful to Amandine Elise Maria Toso, Lucas Carstens and Brian Fitzpatrick, who read this thesis and helped to refine the writing part of the thesis.