TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

DIPLOMARBEIT

A parametric floor plan representation for interactive room editing

unter der Leitung von

Ao. Univ. Prof. Dipl.-Arch. Dr. phil. Georg Suter

E 259-1 Abteilung für Digitale Architektur und Raumplanung

Institut für Architekturwissenschaften

eingereicht an der

Technischen Universität Wien

Fakultät für Architektur und Raumplanung

von

Rungrat Laohavichitsak

01127531

Wien, November 2018

ABSTRACT

The application of parametric modelling methods has been used in architecture to handle a high level of complexity in architecture projects to achieve better data managing and time saving in the floor plan layout-modification process. The parametric design application which has not been well documented in architecture literature is the so-called floor plan representation which have been developed for space layout planning. This process of constructing and composing a floor plan including a related data-structure based graph theory will be introduced in this study to serve as a tool for architectural design methodology.

Parametric design is as medium to map the topology of a template floor plan into different geometric shapes of target unit boundaries. This helps in the drawing process of a project, especially for apartment buildings which have repetitive floor plans for a big number of apartment units and for which they have the same floor plan topology but have small differences in the dimensions of units and rooms which have different shapes. This difference may be caused by the limitation of the building envelop or the building site of the project as well as the size of each individual floor plan. This study will attempt to address these issues and to assist the designer so that he will be no longer obliged to draw or make modifications manually for each unit, one by one, and to be more in control of the modifications of the floor plans.

The mapping and configuration of the floor plans will be based on the constraint system. The constraint system and graph base floor plan representation are the principle element to ensure that the floor plan attribute, such as rooms adjacency, orientation and dimension are not violating the given designer criteria/reference after the floor plan modification and the mapping process have been done. The study of the floor plan representation based on the graph theory is the fundament for the data structure and for the constraint system of the floor plan. The result is that the target floor plan and the modifications of the floor plan will comply with the constraint.

Keywords

Floor plan representation, Parametric modelling, Architectural design

ACKNOWLEDGEMENT

I would like to express my appreciation to Prof. Georg Suter for all the advice and the generous amount of time spent for correction and improvement, which is the most vital contribution to this work.

I would like to express my thanks and appreciation to my best friend Peter and family Sofia and Aquila who always provided support and motivation to pursue my theses to this point.

I would like to thank King Mongkut University of Technology, Thonburi, Thailand, School of Architecture and Design, which where I started my studies of Architecture.

And I would like to thanks to my Dad, my Mom, my sister and brothers and who are in the background giving me always a good wish.

Many thanks

CONTENTS

1 INTRODUCTION

There is a rising demand for computer-based space layout and planning. There are several studies and research reports on space configuration based on graph theory (Nourian, 2016), automated facility layout, space allocation optimization (Liggett, 2000), space layout and planning and architecture morphology (Steadman, 1983).

One element of the study of architecture is to improve the function of organizing space and aesthetic design. This study will focus on the scope of space planning, floor plan layout and computer-based floor plan representation which would help designers to better understand the underlying data structure. This will help the designer to better monitor the space designing process and space management and the configuration of the space within a given limitation and to solve spatial design problems.

There is a high potential for using computer based parametric design and graph-based floor plan representation to optimize the designing and floor plan constructing of apartment buildings, which have a great number of similar floor plans units but have differences geometries and dimensions in both the unit dimensions and its rooms' dimensions within.

The traditional way of drawing floor plans is to draw element by element, copy one to another and changing must be done manually, this could require a lot of work and time specially for complex apartment buildings projects. Such timeframe is often not acceptable for a designer's office. This study will address this issue by using the parametric approach with a graph-based floor plan representation to model a template representing all other apartment units.

The space allocation defined by adjacency is considered in architecture as important and any configuration changes which are made must be limited by this criterion. The building or floor plan criteria such as light, ventilation and pedestrian circulation could be optimized by the floor plan layout editing under the constraints which will be defined by the graph-based floor plan representation.

There are some studies relating to floor plan representation which only address the floor plan with convex rectilinear face representing room, thus limiting the flexibility of the designer. This study will also focus on developing a method of modelling the graph-based floor plan representation with non-convex rectilinear faces.

2 RELATED WORK

The concept of automated layout planning originally comes from the idea of finding a solution for space layout problems. Liggett (2000) states that facility layout problems exist in different scope, as examples of assigning activities and objects in a building. Facility layout problems appear in the process of rearranging spaces in an existing building as well as the space allocation process in a new building, especially in the planning phase. In the conceptual design phase, the space allocation could be made in different variations for the building configuration to find the best solution to design a project in terms of the efficiency of use of space in a building. The layout application can be used to handle the space management problems by increasing and decreasing the number of people and objects and with it changing the space needed. There are several programs that have been developed to automate the space layout and to find solutions for the space layout problems. Such methods attract a lot of attention, especially from architects and interior designers which are interested in facility management and the assigning or reassigning of space within a building, such as offices, hospitals, universities and department stores. Since 1960 there are several researchers which have developed different approaches and solutions of automated generation of plans or layout with the help of algorithm-based applications. The graph theory has been mentioned frequently. This approach will generate a layout based on a node which represents an activity which is placed in certain locations. Two nodes are connected to each other with an edge representing the adjacency between spaces, this is a so-called “Adjacency Graph” (Liggett, 2000). Another approach is that the object will be placed in a way that it satisfies certain given constraints and factors such as distance, position, orientation, path views and adjacency. The Eastman’s General Space Planner method (Eastman 1973).

The book of Woodbury (2010) describes the difference between conventional design and parametric design. He describes the creation of a model in a conventional way which is to be simple assembling parts together by adding elements step by step. In this way the design solution has been made by direct manipulation. The weak point of this design process is that it is difficult to change. For changing one part it needs a lot of effort to adjust many other parts manually. The more complex the model is the more work is required in the changing process. This can hinder the level of innovation within the design process. Parametric design is addressing this issue and increases innovation. Deleting part of the model or fixing the model in a conventional system is easy, because the parts are independent. In contrast parametric design makes it easy because they are interrelated. Parametric modelling is also known as constraint modelling. The design through this system has been made standard for creating and editing relationships as it gives possibility to discover new idea of designing a model and decrease the timeframe of rebuilding the model. The weak point of the parametric system is that it requires skill and time to think and construct the system (Woodbury, 2010).

The graph theory has been introduced as an approach for this study. The Graph theory has been used as a step to solve any problem with the help of graphs and to represent a given situation, for example the situation of finding the path of crossing the bridge in Konigsberg city. The problem is to find a path to cross each bridge only once and return to the starting point. The situation of city of Konigsberg has been illustrated in the form of graph in order to find the solution to the problem (Aldous, 2005).

The study “Turning a graph into a Rectangle Floor Plan” are sets of input data requirements such as lists of faces representing rooms and the adjacencies between them so-called Adjacency-matrix. These data will be turned into an orthogonal floor plan throughout various steps, as example, computing an Adjacency Graph (AG) from an Adjacency Matrix. AG is a graph representing the adjacency between faces representing rooms, vertices representing rooms and edges represent adjacency between them. The two sets of edges in AG are coloured in two colours and assigning two directions. This is called “Coloured Directed Graph” (CDG). CDG is a AG with assigned arrows (directed edges) in two directions are representing adjacencies between rooms through walls in both in x and y direction, vertices represent faces and edges represent adjacency (Roth, Hashimshony, & Wachman, 1982). CDG then is turned into Directed Coloured Sub-Graphs (CDSGs), then these graphs will be turned into Dimension Sub-Graphs (DSGs). CDSG is a subgraph of the CDG, representing adjacencies between rooms through walls only in one direction either x or y, vertices represent faces and edges represent

adjacency between them. DSG is a graph which is converted from CDSG by cutting procedure (Roth et al., 1982).

This graph theory will be introduced in this study for developing a new method to modify floor plans with non-convex cells based on graph representation. Cell is a term which is used in the Roth study and it refers to a room or a small unit of a room which is part of the unit plan. In this study the term "face" is equivalent to the term "cell" and will be used to refer to the graph theory.

The dimensions will be assigned to each face, beginning first with the faces which share the same common wall which is the external wall. A common wall is a wall that divides two faces which are joined together and it belongs to both faces. The assigned dimension is based on a range of min-value and max-value. The designer needs to be concerned about assigning values between min and max value to the face in a way that the adjacency of the floor plan will satisfy the constraint given by the adjacency-matrix. There are no concrete solutions towards assigning the dimensions to the faces in the way that all the faces will maintain adjacency between them. The process of turning a graph into a orthogonal floor plan only addresses the orthogonal convex floor plan boundaries, faces representing rooms are only limited by the convex geometry (Roth et al., 1982).

The study "Generating Layouts with Non-convex unit boundary" is addressing the problem of space allocation in architectural planning. In the study the floor plan layout, which has an orthogonal non-convex boundary is generated from the space allocation program. The study did not mentioned about the non-convex face (Roth, Hashimshony, & Wachman, 1985).

Architectural morphology or configuration is the restriction of geometry could take place on the possible boundary shape. In the book "Architectural morphology" it is recommended that the concept of architectural morphology is to be understood by all those which are involved with architectural design such as building scientist, theoreticians, architecture historians and even anybody who uses/develops computer aids for representation and configuration of building design (Steadman, 1983). This is similar to "the theory of transformation or comparison of related forms" of D'Arcy Wentworth Thompson (1960) which defines animal species by comparing their shape. They could be related to each other based on the significance of geometric transformation. This approach could be used to explain the shape of an architecture floor plan (compare Eastman, 1970; March, 1972).

Steadman mentions in his book that "*The use of the term 'morphology' alludes then to Goethe's original notion, of a general science of possible forms, covering not just form in nature, but forms in art, and specially the form of architecture*" (Steadman 1983, preface).

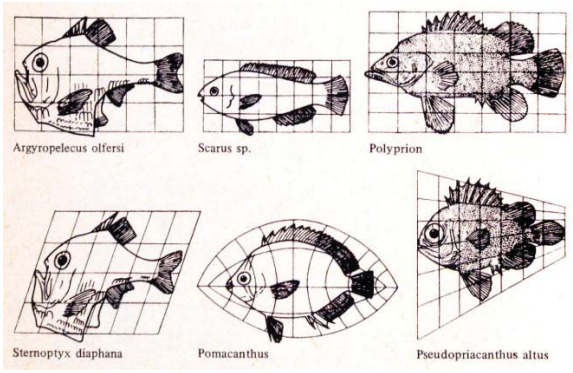


Figure 1 The relationship between the shapes between fish based on 'coordinated method' (from Thompson, 1961, figures 146 to 151).

In the study "Operations on network-based space layouts for modelling multiple space views of buildings", the modelling of different views has been generated based on network space layouts as a purpose to solve the issue of reusing space layouts. Some of the layouts generate a spatial view of circulation, the shortest path or a natural ventilation and access control (Suter, Petrushevski, & Šipetić, 2014).

The modelling of different specific views from the space layout has a very high potential of helping the evaluation of an architecture floor plan in different aspects, such as light and ventilation. This helps a designer for earlier states of the designing process. The parametric design based on graph representation will help architect and save the time for remodelling of a complex building design with great number of floor plan units, which are different in term of floor plan unit boundary but have the same topology of spatial construction, as a result the workflow between configuration of the floor plan and evaluating the floor plan is feasible.

3 LAYOUT EDITING OPERATIONS

Layout editing operations could be defined into two main operations, the first one is "Insertion of rooms topology in a target boundary" where the designer would place preferred source unit plan into the target boundary, which has different boundary geometry and dimensions. Moreover, the designer has to determine dimension constraint for the target unit plan. Dimensional constraints will control the proportions of unit plan and room dimensions. There can be constraints of distances between objects, or between points of objects" (Autodesk, 2017).

The second operation is "Moving a wall". The designer may use this operation to configurate either the source unit plan or each individual target unit plan as an outcome from the "Insertion of rooms topology in a target unit" editing operation. The configuration of a source unit plan will apply and will update this change to all other target unit plans.

3.1 Insertion of rooms in a target unit boundary

The source unit plan with the dimension constraint as defined by the designer will place into different types of target unit boundaries.

The expected input and the expected output for each type of unit boundary is illustrated as shown in Figure 2 to 7, on the left side is the part of user input and on the right side is the output. The source unit plan which is the preference existing unit plan template (see Figure 2 marked in blue) and the target unit boundary.

The next step is the execution of the editing operation, as example shown in Figure 2, the room topology of the source unit will be placed on the target unit boundary, under the dimension constraint. The dimension constraint will define the dimension of each room in the target unit plan (see Figure 2). The dimension lines marked in green refer to the rooms which will not maintain the same dimension as in the source unit. The dimension lines marked in red refer to those rooms which will maintain their dimension.

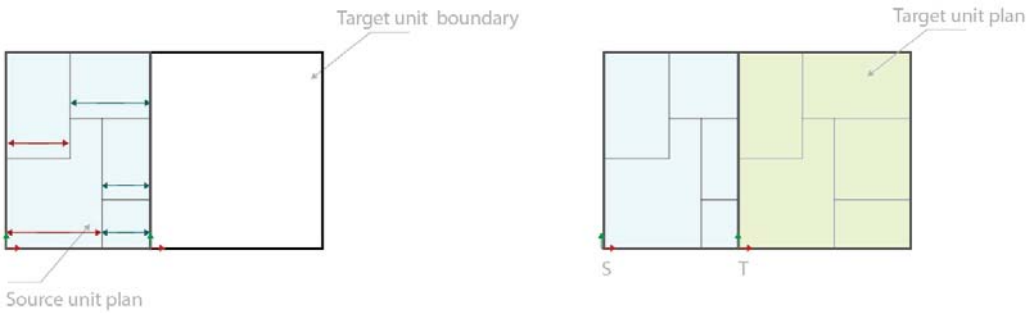
The expected output is a target unit plan (in Figure 2 marked in green) with the same topology, spatial structure and rooms adjacency as in the source unit under the dimension constraint. The constraint will be based on user preference and the type of source unit boundary.

All editing operations for each type of the unit boundary will have the same type of face as have the convex orthogonal and non-convex orthogonal geometry.

The different variations of insertion of rooms topology into a target boundary have in common that each type will refer to the geometry of the target unit boundary.

3.1.1 Rectangular unit boundary

Figure 2 is showing the input of the source unit plan with convex orthogonal boundary and the convex orthogonal target unit boundary. The dimension constraint will define some rooms which are oriented to one side of the coordinate plane and will maintain their original size. This will be decided by the designer through the process of placing the coordinate plane. The expected output is a target unit plan with convex orthogonal unit boundary with the same room topology as in the source unit plan but different in dimensions.



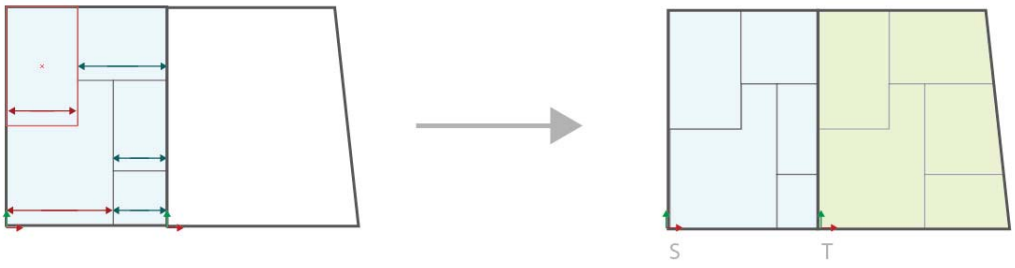
Lines marked in green have different dimension as in source unit plan

Lines marked in red have the same dimension as in source unit plan

Figure 2 Insertion of rooms in an orthogonal target unit boundary.

3.1.2 Quadrilateral unit boundary

Figure 3 is showing the expected input which is the source unit plan with convex orthogonal boundary and the quadrilateral target unit boundary. The dimension constraint is the same as in the previous variation. The expected output is a target unit plan with quadrilateral unit boundary with the same room topology as in the source unit plan but different in dimensions.



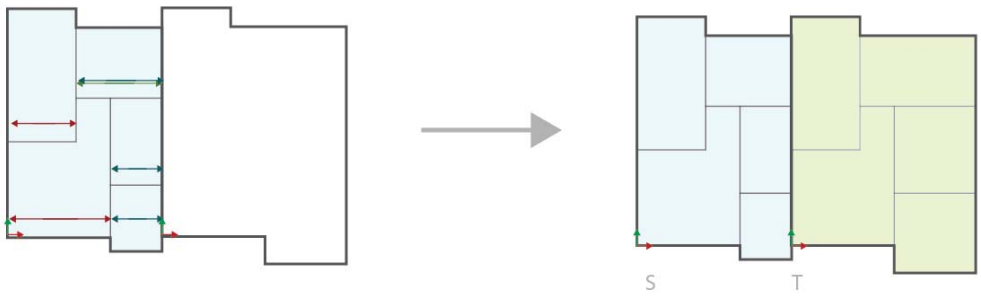
Lines marked in green have different dimension as in source unit plan

Lines marked in red have the same dimension as in source unit plan

Figure 3 Insertion of rooms in a quadrilateral unit boundary

3.1.3 Non-convex orthogonal unit boundary

Figure 4 is showing the expected Input which is the source unit plan with non-convex orthogonal boundary and the non-convex orthogonal target unit boundary. The dimension constraint in this variation is defined by the source unit boundary and the target unit boundary. The expected output is a target unit plan with non-convex orthogonal boundary with the same room topology as in the source unit plan but different in dimensions.



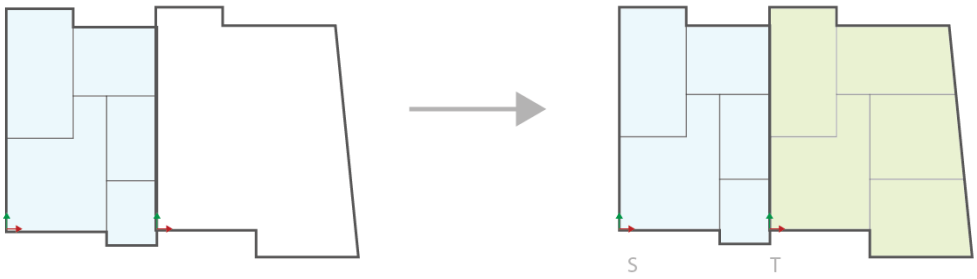
Lines marked in green have different dimension as in source unit plan

Lines marked in red have the same dimension as in source unit plan

Figure 4 Insertion of rooms in a non-convex orthogonal unit boundary.

3.1.4 Non-convex and non-orthogonal unit boundary

Figure 5 is showing the expected Input which is the source unit plan with non-convex and non-orthogonal unit boundary and the non-convex, non-orthogonal target unit boundary. The dimension constraint is the same as in previous variation. The expected output is a target unit plan with a non-convex non-orthogonal unit boundary with the same room topology as in the source unit plan but different in dimensions.



Lines marked in green have different dimension as in source unit plan

Lines marked in red have the same dimension as in source unit plan

Figure 5 Insertion of rooms in a non-convex non-rectilinear unit boundary.

3.2 Moving a wall

This chapter addressing the issue of over dimension constraint when assigning dimensions constraints to the intermediate room. If such constraints are applied, it will have as a result that there are many changes of dimensions of other rooms which are adjacent to this intermediate room. Monitoring the changes of all the other rooms and their constraints by the designer is not feasible (see the last section of “Rectangular unit boundary” in chapter 6.2.1.) Therefore for “Moving a wall” the manual re-assignment of the dimension constraint to specific rooms is the solution towards this issue.

The designers may optionally use this as unit plan modification to make some small changes manually under the constraint-based graph floor plan representation to optimize the unit plan based on unit plan criterion done by designer for example room area or access circulation. This operation could either be executed on the source unit plan or on one of the individual target unit plan (see Figure 6).

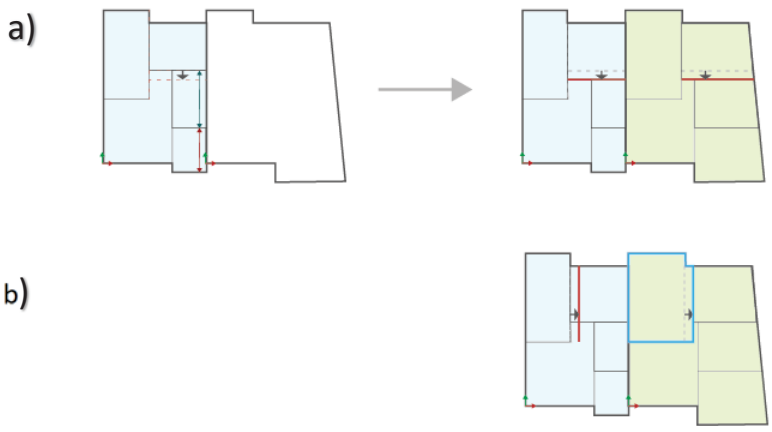


Figure 6 Moving a wall. (a) An example with expected input and expected output of moving a wall which is applying to all output in any target unit plan. (b) An example of moving interior walls of a room which is limited to the unit boundary.

In the case of the applying the source unit plan, the result will be that all other target unit plans will be updated according to only this one change.

There are two different types of dimension constraints concerning adjacent room. They will either maintain their size or they will not.

Figure 6(a) is showing the expected input is the target unit plan as output of any variation of previous “Insertion of rooms topology in a target unit” editing operation. The dimension

constraint is the dimension of the room which share the common wall with the modified room done by the designer and will either maintain its original size or not.

The operation is a manual change of room dimensions by moving a wall

Expected output is the result of the input target unit plan with the manual change of any wall junction defined by the designer and the changing of room dimensions as mentioned above in the dimension constraint.

Note: for non-convex orthogonal unit boundary and in case that if some part of a room attached to unit boundary and that the changing of unit boundary is not possible, then the self-defined dimension constraint of this face done by the designer might change the geometry of the room, for example rectangular room transformation to non-convex rectilinear room as shown in Figure 6(b) where the polygon is marked in a blue. If the shape of a room has been changed then the adjacency between this room and another room of which they share the x-wall and/or y-wall, then they will change and as a result the topology of this target unit plan in terms of adjacency will not be the same as in source unit plan. This issue may be a proposal for further study.

The orientation of the target units plans as a result of different editing operations of the “Insertion of rooms in a target ” at different positions of the building floor plan is shown in Figure 7.

The operation such as mirror and rotation are introduced to define the orientation of the target unit plan in the building. In the case that the target unit plan will have the same orientation as in the source unit plan, this step will be skipped, see Figure 7(b).

In Figure 7(c) the target unit plan (3) is the rotated version of the source unit plan (0) (coloured in blue). The orientation is defined by the coordinate axis x and y in the source unit plan and in the target unit plan.

In Figure 7(b) the target unit plan (7) is the output of the editing operation of “Insertion of rooms in a target ” with a Non-convex and non-orthogonal unit boundary.

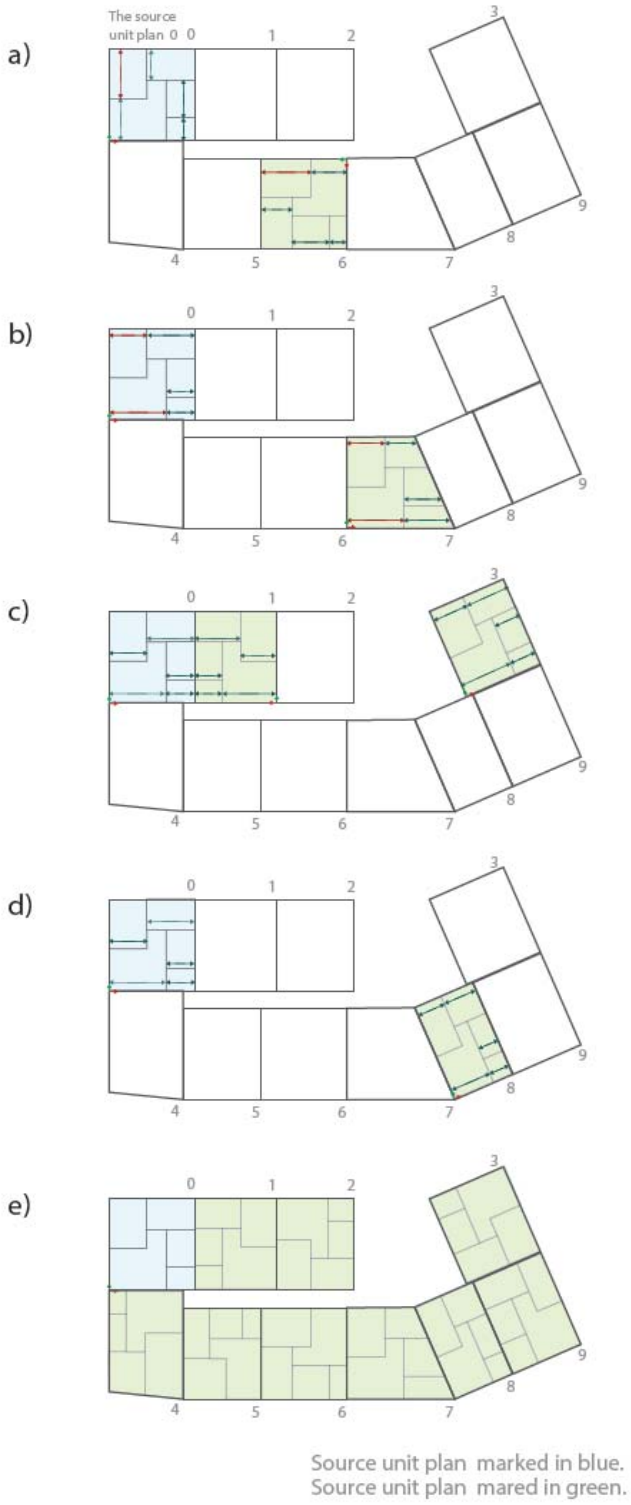


Figure 7 The orientation of target unit plans in a building plan corresponding to mirror or rotation operations of the source unit plan which is defined by the coordinate axis x and y, (a) to (d) The examples of different editing operations “Insertion of rooms in a target ” on the different positions of the building floor plan. (e) The complete of all “Insertion of rooms in a target ” of each target boundary in the building floor plan.

4 DERIVING A GRAPH-BASED FLOOR PLAN FROM A RECTANGULAR FLOOR PLAN

In the study “Turning a Graph into a Rectangular Floor Plan” (Roth et al., 1982), it is mentioned how to use existing data and their adjacencies to compute a orthogonal floor plan through the steps which are involving graphs as the basic information. In this section we will have a look in detail at this data information base.

In the work of Roth, it is demonstrated how to turn a graph into a rectangular floor plan. We will use the principle of Roth’s study to find the data structure representing a floor plan and its limitations towards an approach of mapping the floor plan through the different editing operations. Data structures will be used for the floor plan mapping and editing operations and the configuration according to the designer’s preference.

Referring to Roth’s study it shows step by step how to construct the floor plan from the beginning until the floor plan is accomplished. In this study we would construct a source unit plan from a complete existing floor plan template, with all the data for the dimensions. Therefore, we will focus only on the underlying data structure according to Roth’s principle and to achieve the procedure of “Deriving a graph-based floor plan from a rectangular floor plan”.

As mentioned in the chapter of related work, the Roth method only addresses the convex face, so the floor plan representation of Roth method is limited to convex faces, which might be not be feasible to our approach where we are dealing with an existing floor plan template. We will later develop the method to address this issue in a further chapter. In this chapter “Deriving a graph-based floor plan from a rectangular floor plan” will cover only the source unit plan with only convex faces.

We will use the Roth method and the data structure and its principle to better understand the graph-based floor plan representation by turning the existing floor plan into graphs and find a solution to the issue.

The “Deriving a graph-based floor plan from a rectangular floor plan” is the framework which will be applied to all “Layout editing operations”. This step is important for all “Layout editing operation”. The graph-based floor plan which has been generated from a source unit plan will be used to define the dimension constraint, the adjacency constraint of a target unit plan. The following is the overview of the required steps of “Deriving a graph-based floor plan from a rectangular floor plan”.

The steps must be made by the designer and are as follows.

1. Sketching of the source unit plan with dimensions where the polygon shape should be representing each a room. Labelling of all the rooms should be done by the designer.
2. The Plan Graph (PG) is generated from the designer input in step (1). PG is a graph representing unit plan.
3. The Adjacency Graph (AG) is generated representing the room adjacency. Dual Graph is a graph that has a vertex for each face of a Planar Graph G and Dual Graph has an edge whenever two faces of a graph G are separated from each other (Aldous, 2005)(Aldous, 2005)(Aldous, 2005). The term “Planar Graph” in graph theory refers to a graph where the edges are not intersect each other (Aldous, 2005), Planar Graph G equivalents to PG. Face is a close loop of Planar Graph in graph theory, the term will be used to represent a room and face is equivalent to face.
4. The Coloured Directed Graph (CDG) is an AG with assigning colours and arrows in two directions representing adjacencies between rooms through a wall both in x and y direction and vertices represent faces and edges represent adjacency.
5. The Coloured Directed Sub-Graphs (CDSGs) are generated. CDSG is a subgraph of the CDG, representing adjacency between rooms through walls only in one direction either x or y and vertices represent faces and edges represent adjacency between them.
6. The Dimension Sub-Graphs (DSGs) are generated from the CDSGs. Edge is representing the room dimension. Vertex represents the wall junction. These Graphs will be used as dimension and adjacency constraint for “Layout editing operation”.

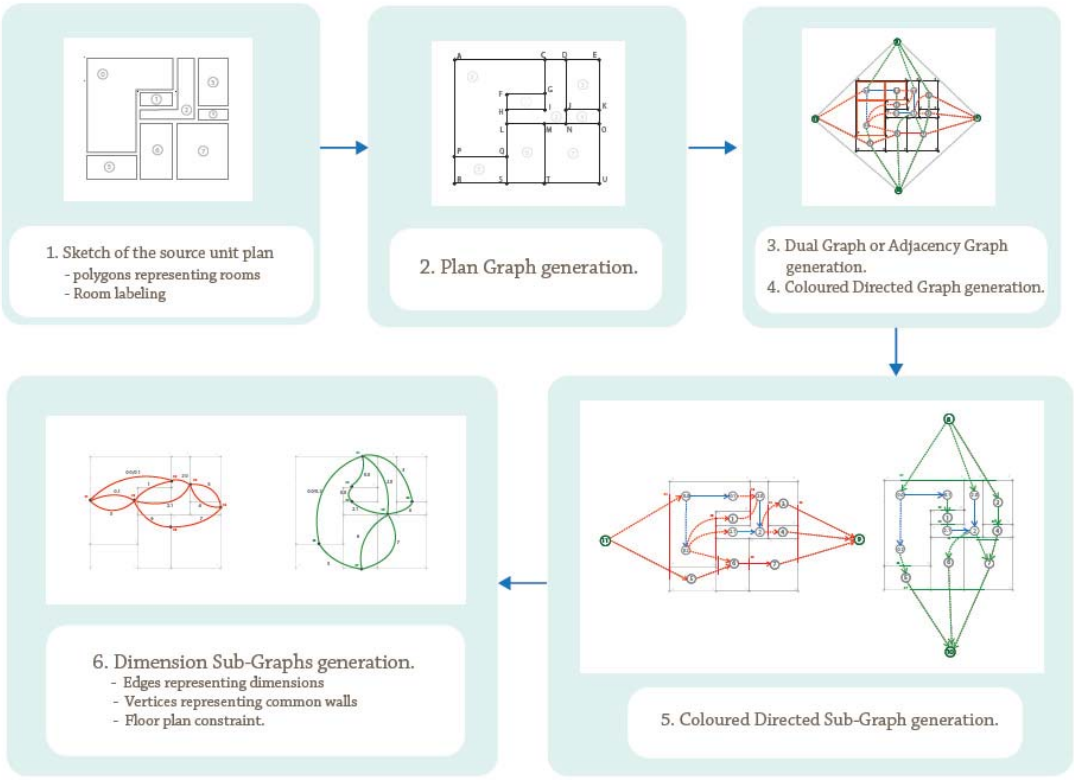


Figure 8 Overview of the steps to derive a graph-based floor plan from a rectangular floor plan.

Following chapters are steps for deriving a graph-based floor plan from a rectangular floor plan. Most of the steps are adapted from the previous work done by Roth, These steps are “Identify rooms and exterior space”, “Drawing an Adjacency Graph”, “Turning an Adjacency Graph to a Coloured Directed Graph”, and “Generating cut lines in a Coloured Dimension Sub-Graph” and “Turning Coloured Directed Sub-Graphs into Dimension Sub-Graphs”. Some of these steps are limited towards our approach.

4.1 Splitting a non-convex orthogonal face into convex orthogonal faces

In this section will demonstrate how to turn existing apartment unit plans with convex faces which have been split from non-convex orthogonal faces into graphs. The unit plan of housing apartment of the project Housing Complex by Hermann and Johannes Kaufmann architects will be used to demonstrate the steps of "Deriving a graph-based floor plan from a rectangular floor plan". Some steps will be based on the study of Roth's principle as mentioned in previous chapter.

In the study of Roth only convex orthogonal faces have been used to represent rooms. The AG for convex orthogonal faces is shown in Figure 9(a). In Figure 9(b) the Adjacency Graph for both convex and non-convex orthogonal faces are shown.

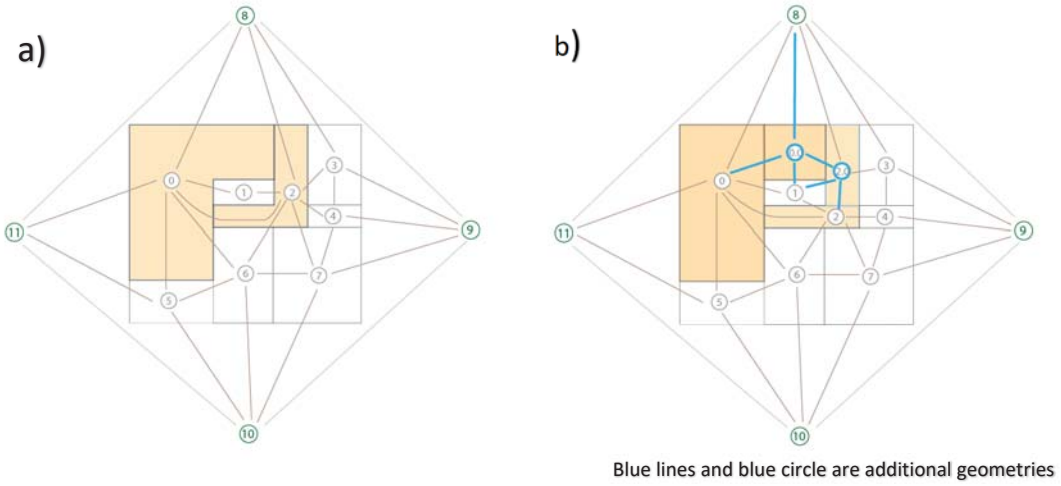


Figure 9 Two different AGs (a) AG for both convex and non-convex orthogonal faces. (b) AG for convex orthogonal faces base on study of Roth.

In the case of some unit plans, which might have many non-convex orthogonal faces it would require more work to turn all non-convex orthogonal faces to convex faces according to the Roth method. In Figure 9(b), the face (0.0) has been split out from non-convex orthogonal face (0), as shown in Figure 9(a), they are coloured in yellow. The work involved is to manage the labelling of all the split faces representing rooms and how we handle the additional edges between the split faces as edges representing walls. In Figure 9(b), there are two addition vertices, vertex (0.0) and vertex (2.0) represent faces are marked in blue as result of splitting the non-convex orthogonal faces (in Figure 9(a)). There are also other issues, as shown in Figure 10(a) to (b), for example there are many possible ways to split the face and as result some outcomes are redundant, in terms of geometrical elements such as faces and edges, as shown in Figure 10 (b) in the outcome (1). In this paper we will develop a new method to

address this issue. Before finding a solution, we would use the Roth method to understand more about the data structure and graph base floor plan representation.

This chapter is working on split convex faces which are part of a non-convex orthogonal face (see Figure 10). From a designer perspective, the optimal way of splitting faces may be to select one which has a lesser number of split faces and the way of splitting a face may be as simple as either splitting a face horizontally or vertically, as shown in Figure 10(b) type (2) and (3). It also might be used full for the architect to have different ways of splitting a face, as shown in Figure 10 the type (5) where the architect could have maximum area as a major function and another smaller area as a minor function As example this face may represent the open plan kitchen where the kitchen is integrated in to a living room and the major area would be the living part and minor area may be a kitchen and a storage space. The splitting of convex faces into non-convex orthogonal faces may be useful for the mixed used zone of a room but there is an issue of which the designer may be concerned, that is that the edge between two split faces represents a wall according to the Roth method.

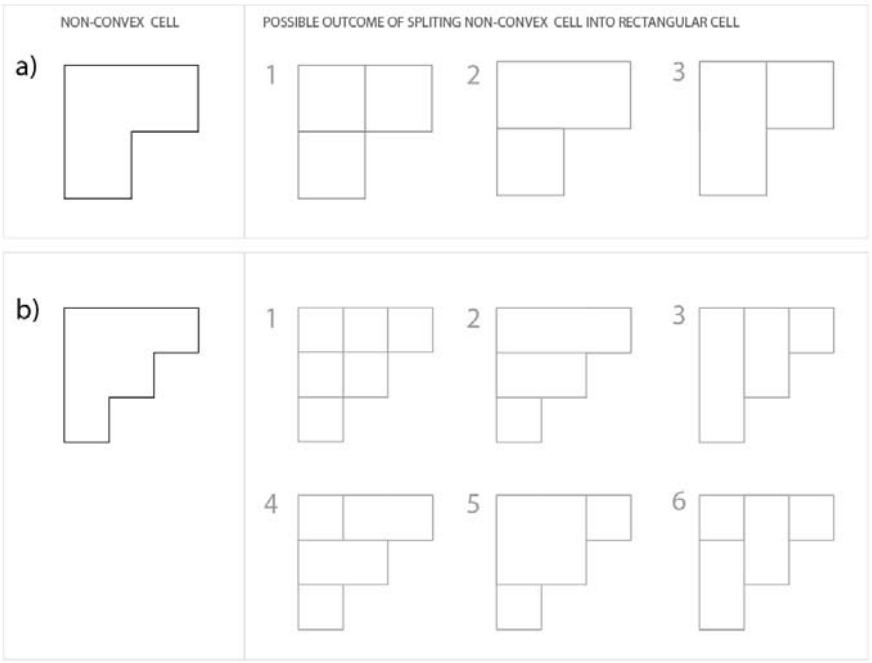


Figure 10 Example of ways toward splitting a non-convex orthogonal face to convex orthogonal faces. (a) The first example of non-convex orthogonal faces and its possible ways of splitting non-convex orthogonal faces to convex orthogonal faces. (b) The second example of non-convex orthogonal face with a greater number of vertices faces and its possible ways of splitting non-convex orthogonal faces to convex orthogonal faces.

4.2 Identify rooms and exterior spaces

First draw polygons representing each room or space. Next is to identify and label the name for each room on the source unit plan. As mentioned in the previous chapter of splitting a face (see Figure 11), the face (0.0) has been split from face (0) and represents an area of a room (0). In Figure 12, the numbers (0) to (7) have been given to the rooms in the source unit plan. The numbers (8) to (11) marked in green represent the four infinite external sides of the source unit plan. We did not use alphabet (N), (E), (S), (W) to represent North, East, South, West for external orientation side, because we want to avoid the confusion in the case of some target unit plans do not have the same orientation as the source unit plan (Roth et al., 1982).

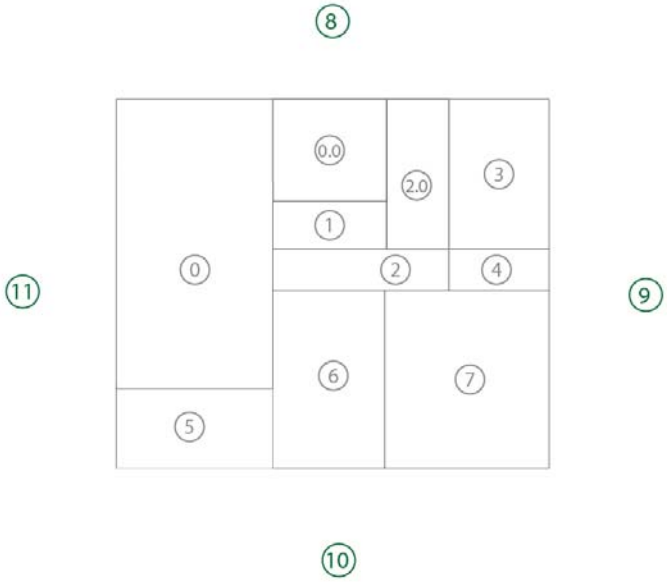


Figure 11 Drawing polygons representing rooms and labelling of each room.

4.3 Drawing a Plan Graph

Graphical representation is a method, which is often used to describe a situation to find steps for solving problems, for example to find the shortest way between two points on a subway network. The terminology associated with the term graph can be read in detail in the book "Graphs and applications an introductory approach" (Aldous, 2005). In this study we will use graphs to represent a floor plan and its dimension in order to construct a source unit plan. The graph consists of a set of points, called vertices, joined by lines, called edges; each edge joins exactly two vertices (Aldous, 2005).

The plan graph will be constructed representing source unit plan. This source unit plan will be used as an input for “Layout editing operation”. The vertices are representing the wall junctions. The edges are representing the wall segments.

We could define the wall segment into two subcategories, as shown in Figure 13 the x-wall coloured in pink and the y-walls are coloured in blue.

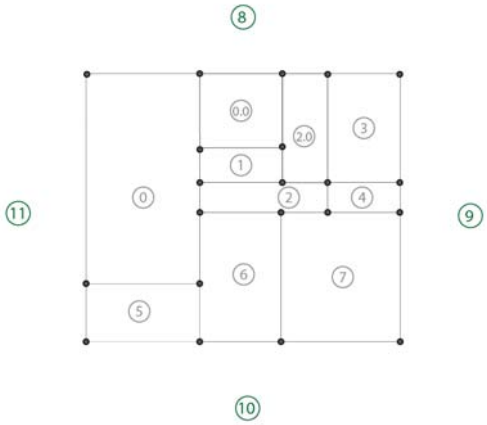


Figure 12 Plan Graph

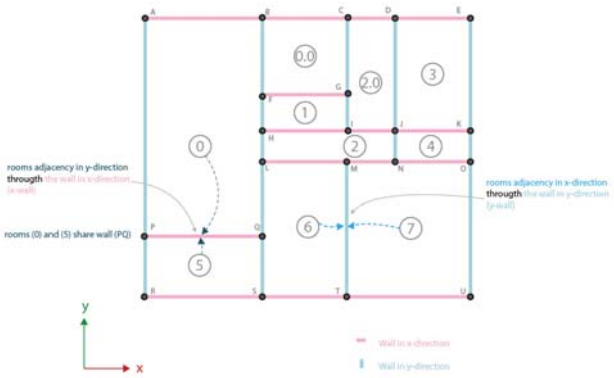


Figure 13 The adjacency between rooms in both x and y direction through the wall in y and x direction.

4.4 Drawing an Adjacency Graph

The AG consists of edges which represent the adjacencies between a pair of vertices representing a room in the source unit plan, as shown in Figure 14. The adjacency between rooms can be defined into two groups, the first one is the adjacency in y-direction through the x-wall and the second one is the adjacency in x-direction through the y-wall, as shown in Figure 13. The adjacency types are defined by the type of wall, x-wall or y-wall. Giving an example, vertex (0) and (5) are adjacent one to an another in y-direction through the x-wall, so an edge is drawn to connect these two vertices (Roth et al., 1982).

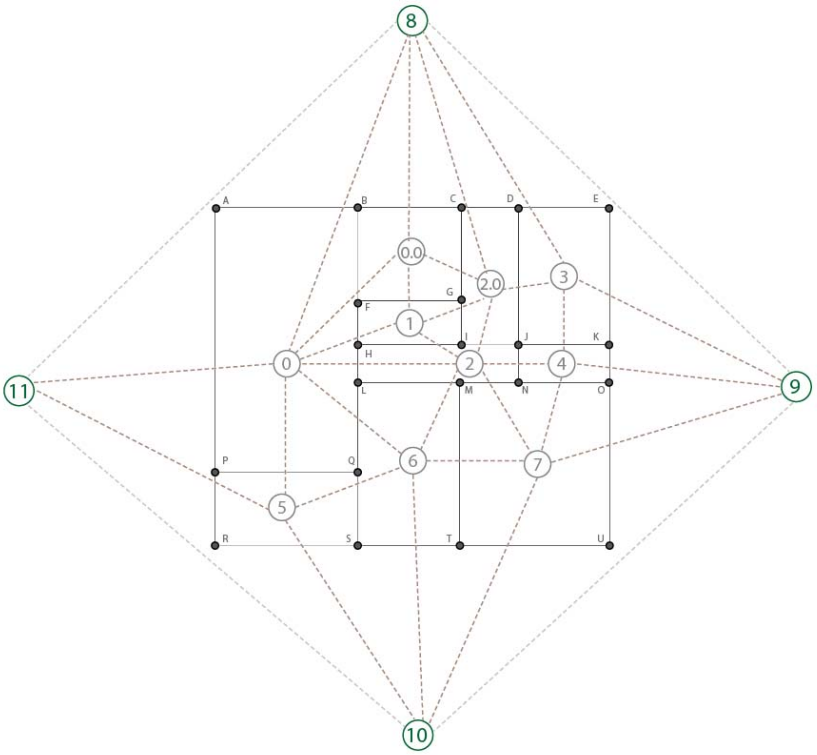


Figure 14 An Adjacency Graph representing the adjacency between rooms in a Plan Graph.

As mentioned earlier, the process of splitting a non-convex orthogonal face into convex orthogonal faces will have some issues not only of redundancy of the element in plan graph, but also the structure of the AG will change. The quantity of edges and vertices of the AG are also increased, as shown before in Figure 9(b), the additional edges marked with dash lines and vertices marked in blue. These elements are significant in terms of their representation in a unit plan e.g. common walls are represented by vertices in DSG and this as the result of CDSG and the AG in this chapter.

4.5 Turning an Adjacency Graph into a Coloured Directed Graph

All edges in an AG will be defined in two different sets and these two sets will be represented by two different colours (see Figure 15(a)). The two different sets are defined by the type of adjacency as mentioned previously, the adjacency in y-direction and the adjacency in x-direction. In Figure 15(a), the face (6) and (7) are adjacent in x-direction through the y-wall. This corresponds to adjacency edge (6, 7) of AG which intersects the y-wall. Red colour represents a set of adjacent edges in the x-direction. Two subgraphs are coloured in two

different colours according to the types of adjacency. Each subgraph consists of either a set of adjacent edges in the x-direction or adjacent edges in the y-direction (Roth et al., 1982).

Two sets of arrows, one has a direction from top to bottom, and the other one has a direction from left to right will direct each subgraph of CDG representing the direction and sequence of the edges in both subgraphs. The graph which consist of directed edges is called the directed graph or the digraph (Aldous, 2005). The two subgraphs of CDG is called CDSGs. CDSG for y-direction, coloured in green, will be directed by arrows from top to bottom. The CDSG for x-direction, coloured in red, will be directed by arrows from left to right. The coordinate plane will be constructed with x and y vectors to represent the directions for each CDSG (Roth et al., 1982).

The vertex (8) (see Figure 15(a)) is a starting vertex for the CDSG for the y-direction. We will call the vertex (8) the source and call vertex (10) the sink. And the vertex (11) and (9) will be source and sink for the subgraph for the x-direction. Figure 15(b) and Figure 15(c) are showing the CDSGs for both the y-direction and x-direction (Roth et al., 1982).

The vertical blue line (see Figure 15(c)) and the horizontal brown lines (see Figure 15(b)) in the CDSG for x and y direction are cut lines referring to Roth method, which then later represent the common walls in PG.

4.6 Generating cut lines in a Coloured Dimension Sub-Graph

The rules are that any edges coming out or entering the common vertex, generate a cut line. Figure 15(b), the edge (1, 2) and the edge (2.0, 2) are the edges that are entering the common vertex (2), so it creates a common cut line, and the edges (2, 6) and (2, 7) are the edges that are coming out from the common vertex (2) therefore a common cut line is generated. When there is an edge, which has been cut more than one time, these cut lines will be merged into one cut line (see Figure 15(b)),for example the edge (2,7) has more than one cut line and this cut lines are merged into one (see Figure 15(d)). This common cut lines in the CDSG are representing the common walls in the PG (Roth et al., 1982).

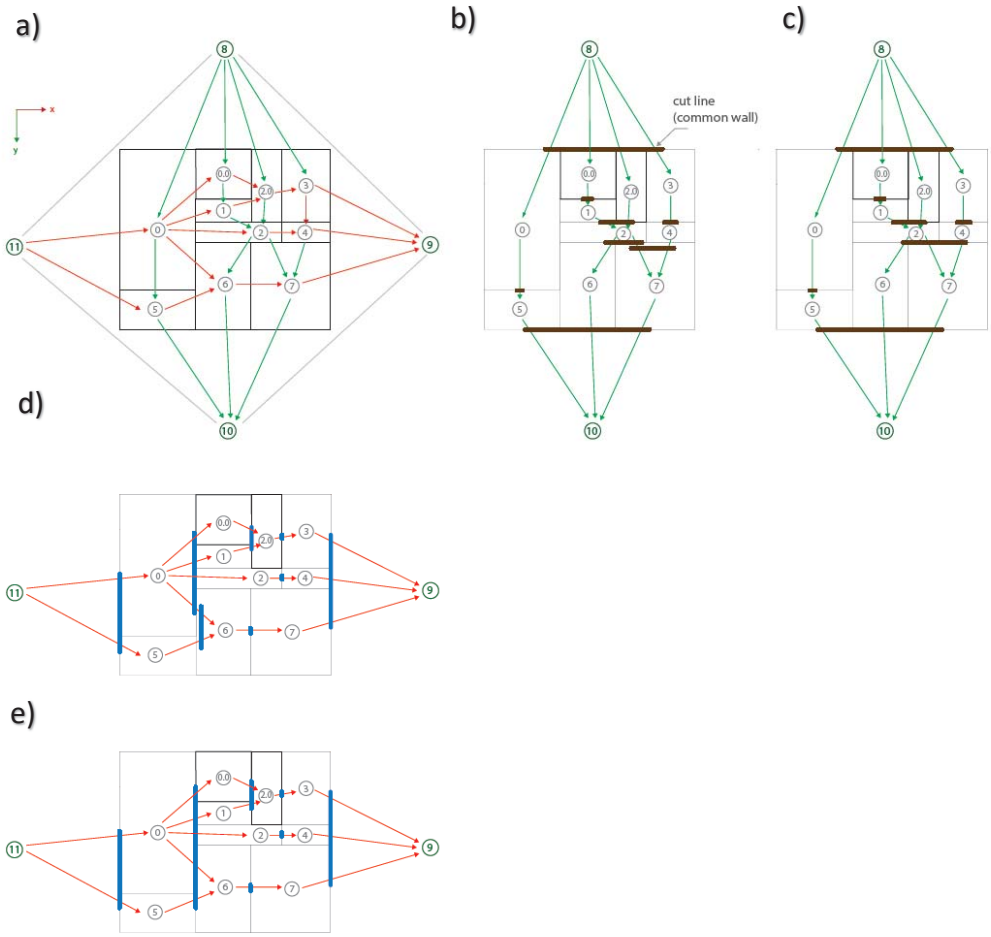


Figure 15 The graphs showing steps of generating cut lines in Coloured Directed Subgraph defining the common walls of the Plan Graph. (a) A Plan Graph and its CDG (b) CDSG for y-direction cut lines. (c) CDSG for y-direction with cut lines are merged. (d) CDSG for x-direction with cut lines. (e) CDSG for y-direction with some of cut lines are merged.

4.7 Turning Coloured Directed Sub-Graphs into Dimension Sub-Graphs

The cut lines in CDSGs will be turned into vertices in the DSGs and the vertices in CDSGs will be turned into edges in DSGs. In Figure 17(b), the vertex (0) of CDSG is in between the cut line (A_y) and (F_y) , these cut lines then turned into vertexes (A_y) and (F_y) in Figure 17(d) and they are connected by edge (0_y).

The vertices in the DSGs represent the common walls in the PG and the edges in the DSGs represent the dimensions for each face in the PG. Each face has two dimensions one in x and one in y direction and these can be read in the DSGs (for x-dimension or for y-dimension) (Roth et al., 1982).

DSGs will be used for Layout editing operation to define dimension and adjacency constraint.

4.8 Assigning dimensions and determine dimension constraints

The numerical value, or weight, assigned to each edge of the graph is called "Weighted Graphs". A DSG is a weighted graph as a numerical value is assigned to each edge.

In Figure 16(a) the width (in x-direction) of the room (7) and constraints on room dimensions, determine the dimension for each room that with an assigned dimension weight to each edge in a DSG. The width of room (7) equal to 5.14 will be assigned onto the edge (7_x) in DSG for x dimension, as shown in Figure 16(C). The width (in x-direction) of the unit boundary is the sum of all the weight values of the edges in any path of a DSG from source to sink. Source refers to the DSGs as a flow network, there is a single source, into which no edge enters, and sink refers to a DSGs as a flow network, there is a single sink from which no edge comes out (Roth et al., 1982). In Figure 16(c) the sum of the edges in path (0x, 0.0x 2.0x, 3x) will be the x-dimension of the source unit boundary. The same principle applies to the unit plan (see Figure 17(c)). "In graph theory, a path is a finite or infinite sequence of edges which connects a sequence of vertices which, by most definitions, are all distinct from one another" (Wikipedia, n.d.).

In Figure 16 (a), the height for the room (7) in the PG is corresponding to the weight of the edges (7_y) in the DSG (see Figure 16(b)).

Determining the dimension of each room is described in more detail in chapter 6.2.1 Rectangular unit boundary.

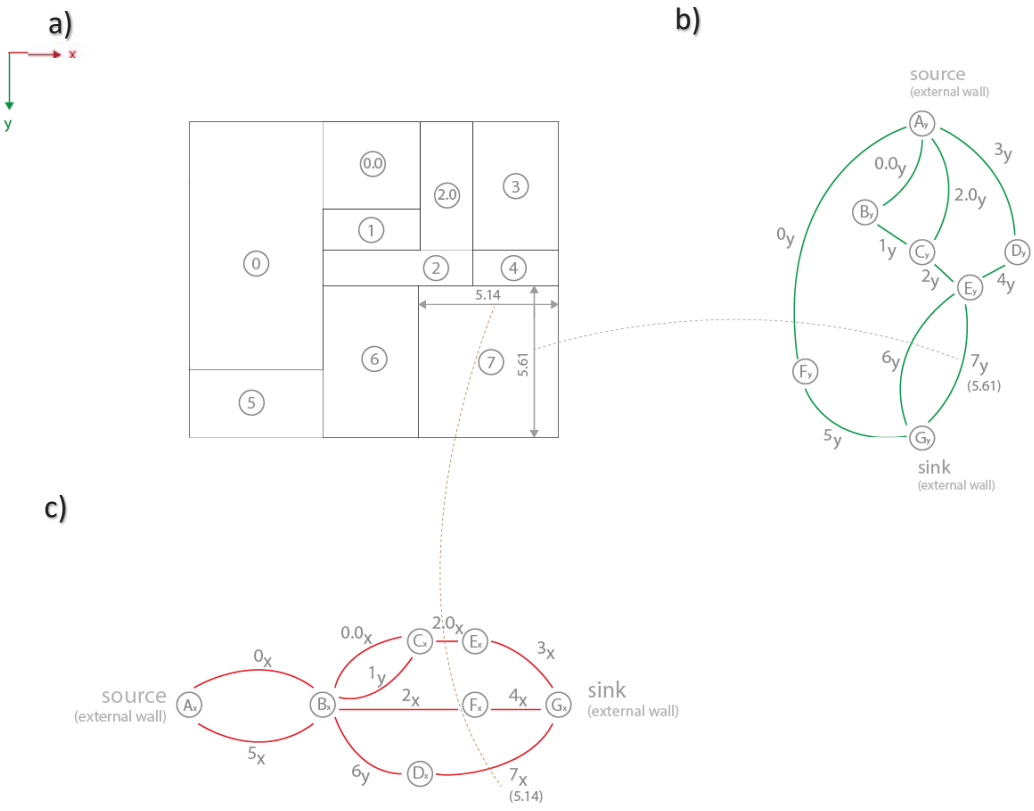


Figure 16 Assignment of room dimension constraint of room (7). (a) the PG with width and the height of room (7). (b) The DSG for y-dimension with height (y-dimension) value of room (7) assigning to the corresponding edge. (c) The DSG for x-dimension with width (x-dimension) value of room (7) assigning to the corresponding edge.

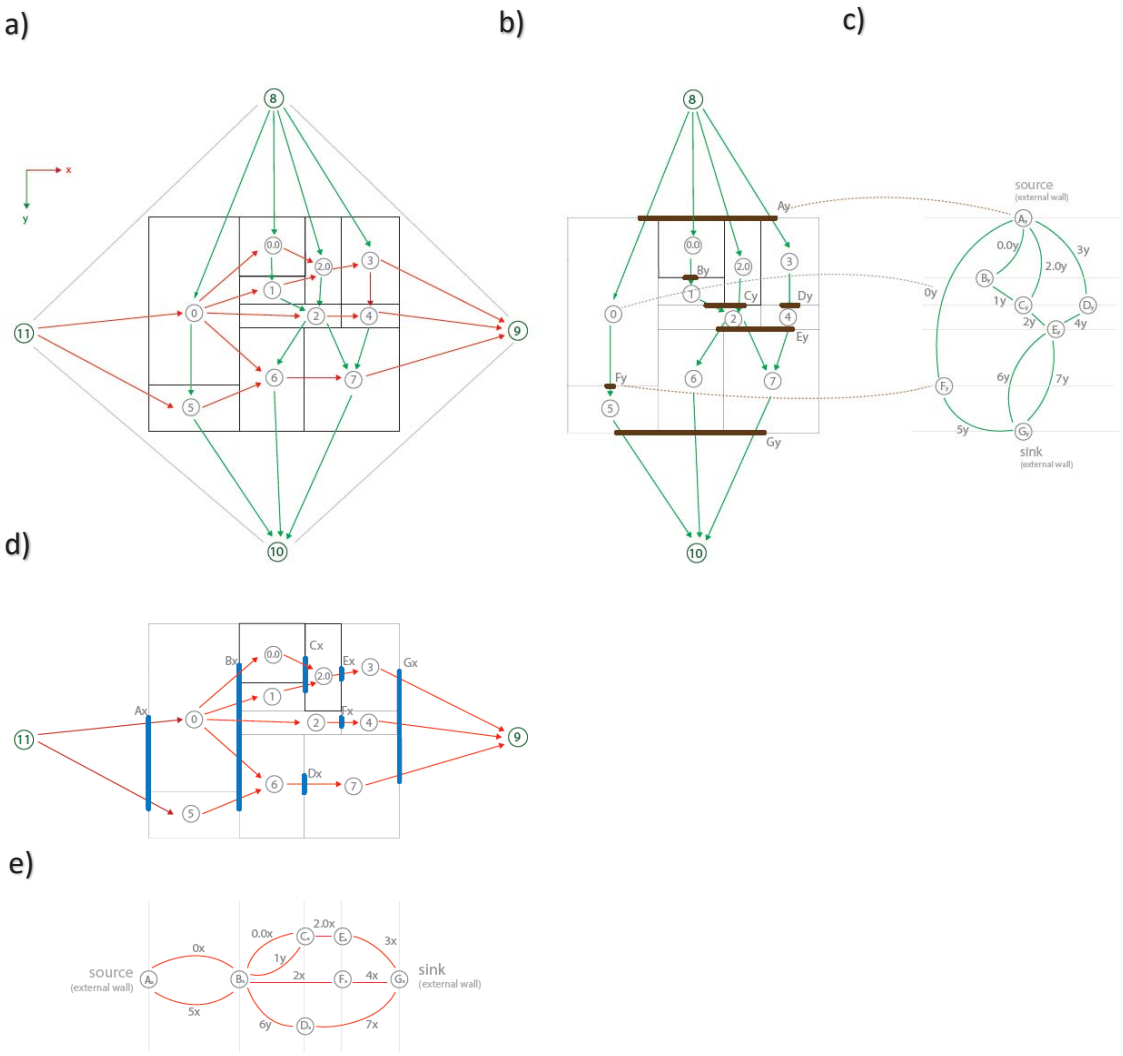


Figure 17 The relationship between CDSGs, DSGs, PG (a) A PG and its CDG (b) CDSG for y-direction with cut lines. (c) DSG for y-dimension. (d) CDSG for x-direction with cut lines. (e) DSG for x-dimension.

5 PLANS WITH NON-CONVEX ORTHOGONAL FACES

According to Roth, the graph-based floor plan representation address only orthogonal faces as shown in a previous chapter. Splitting a non-convex orthogonal face to convex orthogonal faces has several issues, for example when there are many non-convex orthogonal faces in the unit plan it would be more work to turn all non-convex orthogonal faces to convex faces. The work involved is to manage the labelling of all the split faces representing rooms and to manage the additional edges between the split faces as edges representing walls. There are many possibilities to split the face and as a result some unit plans can be redundant in terms of elements, such as faces and edges, as it is mentioned in a previous chapter. This issues with the Roth method are limited to our approach as mentioned earlier in a previous chapter. In this chapter it is shown now that the graph floor plan representation from the Roth method could not be used to represent the non-convex orthogonal face. The issue occurs in the procedure of generating cut-lines, therefore the Dual Graph is introduced to address the issue. The understanding of graphs will be used to develop a new method of turning a graph with non-convex orthogonal faces into a unit plan in the way that there are no additional edges representing walls in the PG. As result this will be the fundament for “Layout editing operations” of the source unit plan with non-convex orthogonal faces and the unit plan modifications under the graph constraint based on floor plan representation with convex and non-convex orthogonal faces.

5.1 Understanding the Dual Graph

Figure 18 shows the source unit plan, which is represented by the Planar Graph G (The Planar Graph G equivalents to PG). The black small circles represent vertices and each edge joins a pair of vertices, which are shown in solid lines. Each edge in a PG separates two faces, for example the edge (L, Q) separating face (0) and face (6), (Steadman, 1983).

To create the Dual Graph G’ from this Planar Graph G, we would place a vertex (big white circle) on each face of Planar Graph G including an infinite face which are the four exterior orientations represented by the vertices (8), (9), (10) and (11). The vertices of Dual Graph G’ corresponding to the faces in the Planar Graph G. We will draw an edge to join each pair of vertices of the Dual Graph G’, which are on the both sides of each edge of Planar Graph G. Each edge of the Planar Graph G will be intersected by each edge of Dual Graph G’(Aldous, 2005).

As example, (see Figure 18) the edge (B, F) in the Plan Graph G represent a wall segment between two faces, the face (0) and the face (0.0) of Plan Graph G., these faces correspond to vertex (0) and (0.0) of Dual Graph G', which are connected with the edge of the DG G'.

The graphs CDG, CDSG and DSG are based on this Dual Graph principle. There is a relationship between a Planar Graph G and its Dual Graph G'. In Figure 18, the vertex (0) in the Dual Graph G' has a degree of (7), this would mean that this face is composed by seven edges in the Plan Graph G. To summarize this face (0) is adjacent to faces through these seven edges.

The x-dimension of room (0) would be the distance between edge (A, P) and the merging of edge (B, F), (F, H), (H, L) and (L, G).

We will use the relationship between Planar Graph G and its Dual Graph G' to develop our new graph based on the floor plan representation for non-convex orthogonal faces.

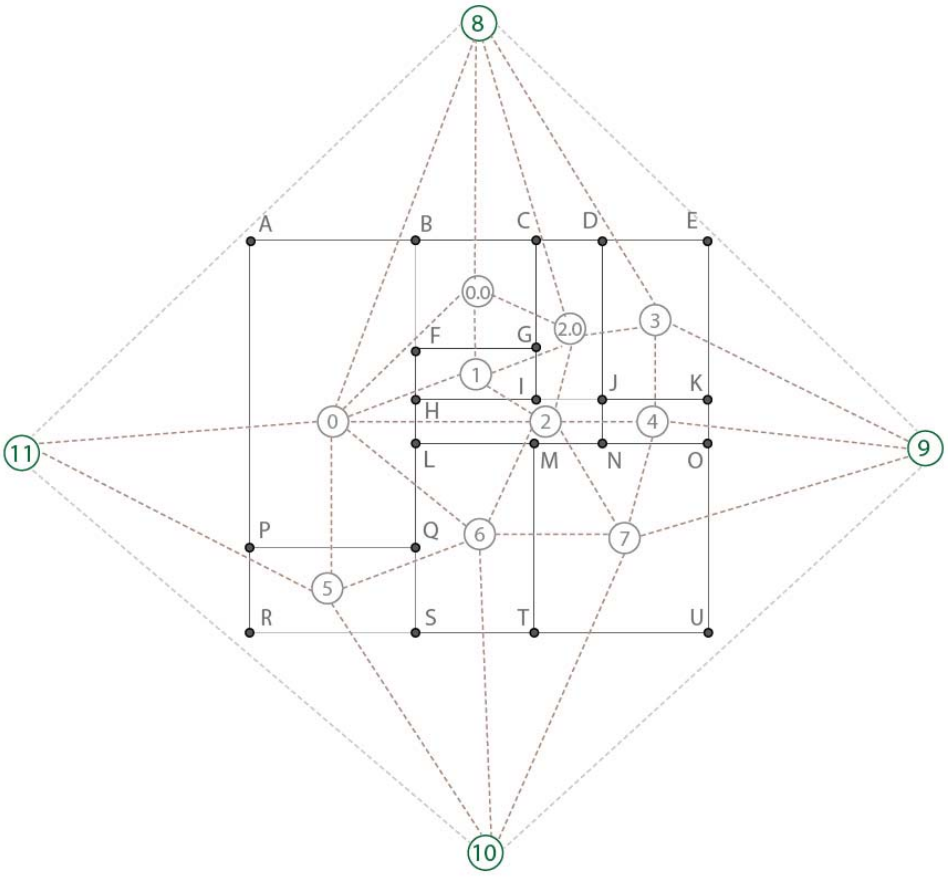


Figure 18 Graph G and its Dual Graph G' represent the floor plan with convex orthogonal faces.

5.2 Plan Graph, Dual Graph and Dimension Sub-Graphs

Figure 19 shows relationships between the Planar Graph, Dual Graph G' and DSGs. The vertex (1) of Dual Graph G' is placed on the face of Planar Graph G, Vertex (2), (3), (4) and (5) represents all other infinite faces which are adjacent to the face (1) and the dash lines represents the adjacencies between them. The vertex (1) is adjacent to the vertex (2) through the edges (A, B) and so on. The green and red cut lines are the cut lines which were generated according to the method of converting the CDSG (see Figure 19(a)) into a DSG (see Figure 19(b) and (c)) according to the Roth method. Dual Graph is equivalent to AG; therefore, Dual Graph could be used to develop CDG and CDSG for non-convex orthogonal face.

The distance between edge (A, B) and (D, C) is the y-dimension for the face (1) which is also shown in the DSG (see Figure 19(b)). The vertex (1) has a degree of four, which means it is adjacent to four different faces through four different edges of the Planar Graph G. The face (1) represents the convex orthogonal face. This face has two dimensions x and y, which is different from non-convex orthogonal face which has more than one dimension both in x and y direction.

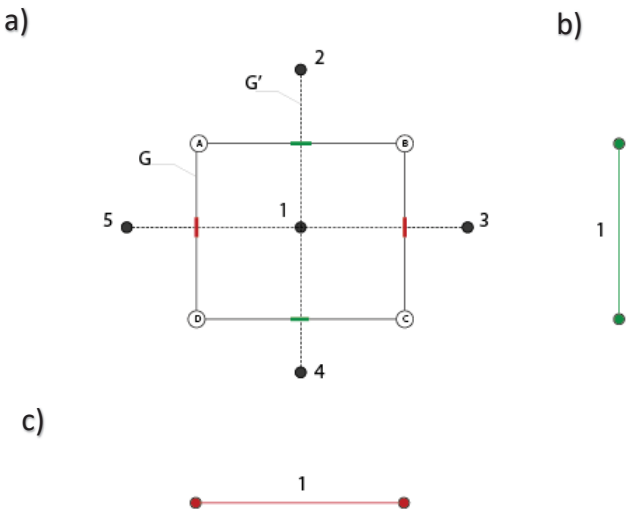


Figure 19 Interrelation between Plan Graph, Dual Graph and DSGs (a) Planar Graph G and its Dual Graph G' with the cut lines. (b) DSG for y-dimension. (c) DSG for x-dimension.

5.3 Turning non-convex orthogonal faces into Coloured Directed Sub-Graph based on the Roth method

Now we will turn non-convex orthogonal faces into graph according to the method of turning CDSGs into DSGs from the Roth method. For any edges which come out or enter a common vertex we will create a common cut line. In Figure 20(b) and (c), CDSGs and DSGs for the y-dimension were created following the Roth method.

There are two issues relating turning non-convex orthogonal faces into graph according to the Roth method.

The first issue is that, there is only an edge (1) in DSG (see Figure 20(c)) which represents only one dimension of the face (1) in the y-direction instead of two dimensions, as shown in Figure 21. These graphs could not represent the non-convex orthogonal face.

The reasons which cause this result is that cut lines are not based on the Dual Graph Principle, but rather according to the edges which enter or come out from the common vertex.

Note: In the Roth Method, generating cut lines in CDGSs is not done from an existing plan like in our approach of constructing a source unit plan and to find the underlying data structure to map and to configure the target unit plan.

In next chapter the Dual Graph is introduced for solving the above-mentioned issue.

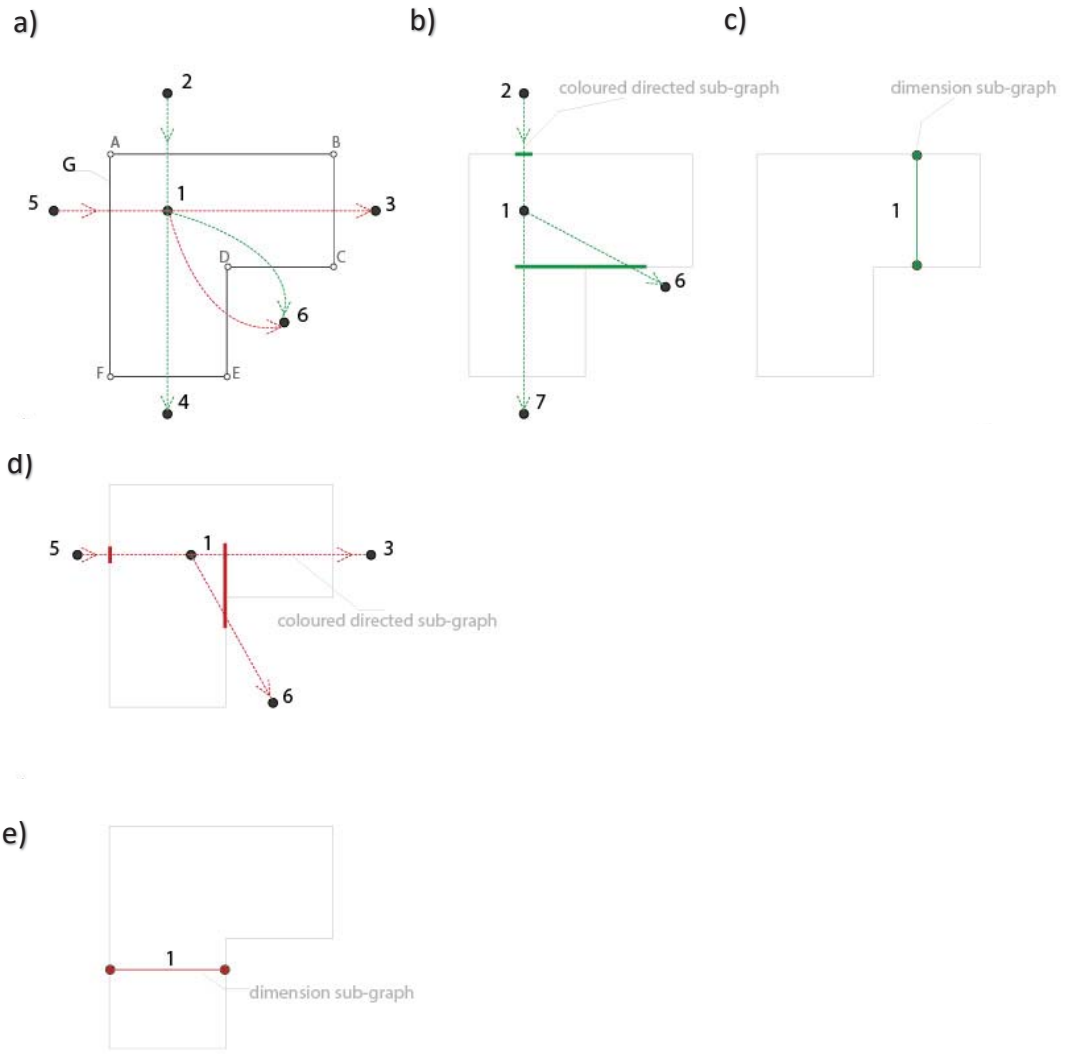


Figure 20 Turning non-convex orthogonal faces into graph based on the Roth method. (a) The Plan Graph G and its AG. (b) CDSG with cut lines for y-direction. (c) DSG for y-dimension. (d) CDSG with cut lines for x-direction. (e) DSG for x-dimension.

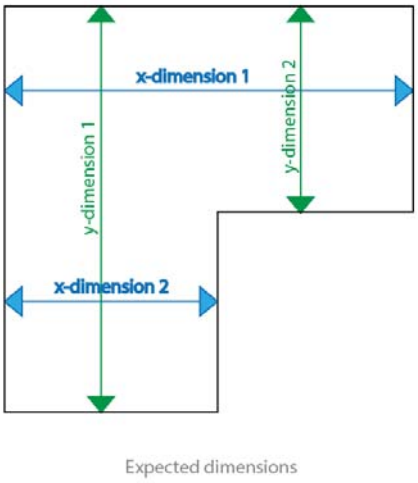


Figure 21 The expected dimensions for non-convex orthogonal face, two dimensions for x-direction and two dimensions for y-direction.

5.4 Turning non-convex orthogonal face into Coloured Directed Sub-Graph and Dimension Sub-Graph based Dual Graph method

In a previous chapter the issue of generating a graph representing non-convex orthogonal faces based on the Roth method was discussed. In this chapter the Dual Graph is used to develop graphs representing non-convex orthogonal face.

5.4.1 Generating cut lines representing common walls

In the Dual Graph G' (see Figure 22(a)) vertex (1) Adjacent to different other faces through six edges of the Planar Graph G, each edge of Dual Graph G' intersect exactly one edge of Planar Graph G. This intersection will be used to generate cut lines in the CDSGs (see Figure 22(b) and (c)).

The cut lines correspond to the edges which are composing the face (1) in Planar Graph G, as shown in Figure 22(a) edge (A, B), they are marked in green and correspond to a cut line in CDSG in Figure 22(b). We can use this edge of the Planar Graph respectively for the Dual Graph method and to generate cut lines for the non-convex orthogonal faces rather than J. Roth's method. As result, two edges in DSG represent two y-dimensions of face (1) which have been created, as shown in Figure 22(c).

The next issue relating to the Dual Graph method is that there is only vertex (1) in CDSG (see Figure 22(c)) representing two different unique dimensions in DSG (see Figure 22(d)).

If we use the Dual Graph to create cut lines (see Figure 22(b) and (d)), we will solve the first issue but not the second issue, where one edge represents two different dimensions in the DSGs.

The main objective is finding a solution which will address this problem. The solution is to create two identical names representing two different dimensions in DSG and the name should correspond to the vertices in CDSG.

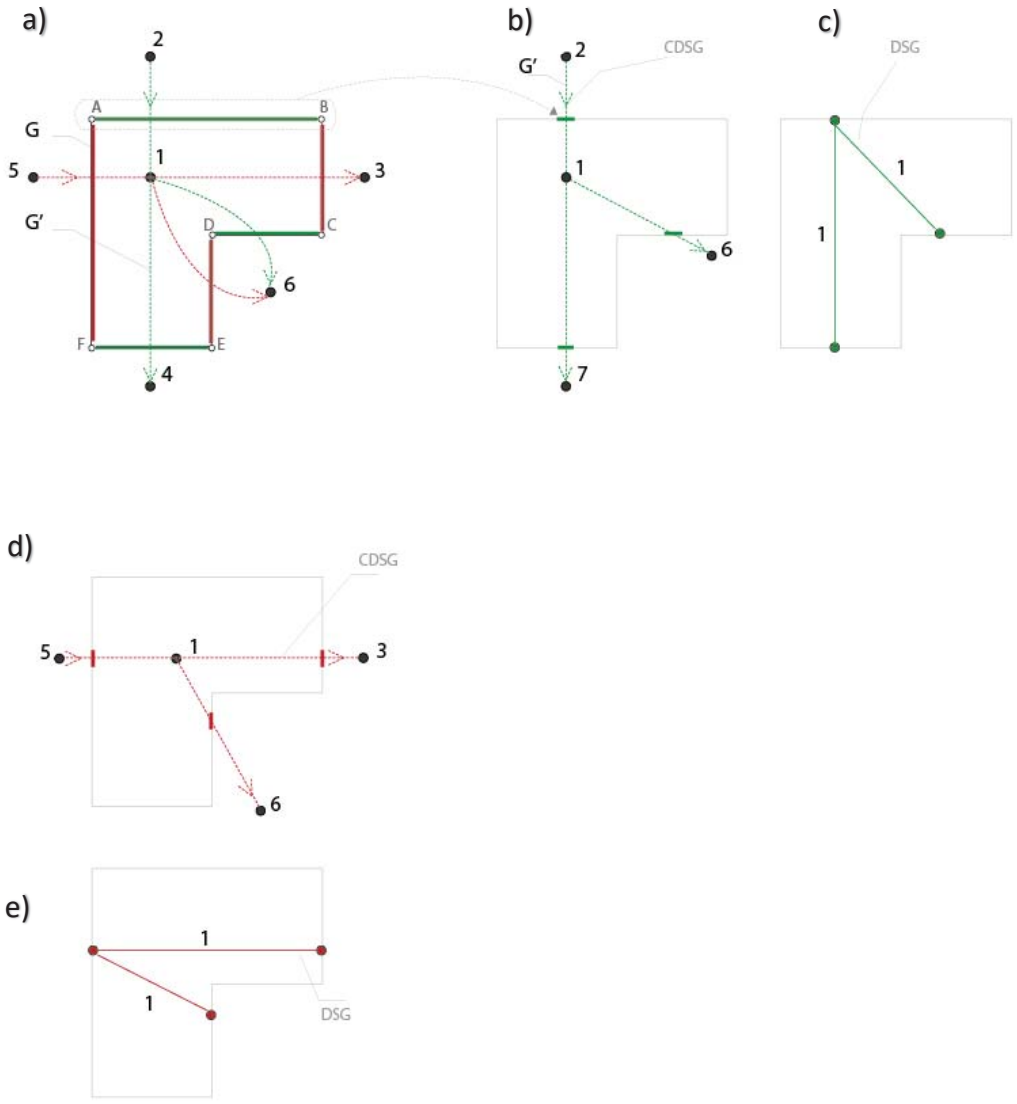


Figure 22 Turning non-convex orthogonal faces into graph based on the Dual Graph method. (a) The Plan Graph G and its Dual Graph G'. (b) CDSG with cut lines for y-direction which are generated according to Dual Graph method. (c) DSG for y-dimension. (d) CDSG with cut lines for x-direction which are generated according to Dual Graph method. (e) DSG for x-dimension.

5.4.2 Finding identical names representing different dimensions in Dimension Sub-Graph

The procedure of solving the issue can be done by trying to split the non-convex orthogonal face to a convex orthogonal face, as shown in Figure 23(a). Here we see that the edge (G, D) has been created. Because we don't need the wall represented by this edge, therefore this edge will not create a cut line (representing a wall). We will use this edge as an imaginary edge to split this non-convex orthogonal face. Thus, we can be sure that one vertex will not represent two different dimensions, since now we have two vertices represent two split faces.

According to the procedure above, there is not enough information available to create the system to name the edges for the DSGs.

In Figure 23(b), it shows further developments, which have been derived from the earlier step. The non-convex orthogonal face has now been split into three faces. We would call these faces split faces of face (1). The vertex is placed in each of the split faces. The cut lines are created according to the Dual Graph method, the edges (as imaginary edge marked in orange lines) between split faces will not generate cut lines, because these edges will not represent the wall in a Planar Graph G.

The edges which are created to split non-convex orthogonal faces will not change the original topology of a PG in terms of numbers of vertices and edges. They are needed for creating split faces to create a vertex representing a split face. These new vertices will not represent the face (1), they are needed for defining a system to identify the edges in DSG.

The result, as shown in Figure 23(b), are two sets of paths both in x and y direction. The red and blue dash lines are oriented horizontally as part of paths in the x-direction, the blue represents edges connecting the vertices that are representing the split faces. The blue dash edge will not generate a cut line. The first and second path in x-direction are the path (5, 1.0, 1.1, 3) and the path (5, 1.2, 6). In the y-direction are the path (2, 1.0 ,1.2, 4) and the path (2, 1.1, 6). There are now four paths representing four dimensions, two paths for the x-dimension and two paths for the y-dimension.

In a Dual Graph G' only one edge intersects one corresponding edge of the Planar Graph G, but in Figure 23, there are two edges, edge (2, 1.0) and edge (2, 1.1) of Dual Graph G' represents an adjacency between two faces. The face (1) is supposed to be adjacent to the infinite face (2) through only one edge of Dual Graph G'. The edge (2, 1.0) and edge (2, 1.1) which intersect with the same edge (A, B) of the Plan Graph G thus representing an adjacency between face (2) and face (1). The edges which separate split faces are not considered as part

of Plan Graph G and its Dual Graph G'. Considering vertex (1.0), (1.1) and (1.2) as sub-vertex they are representing vertex (1) which is corresponding to face (1).

To fulfil this requirement, we will merge edge (2, 1.0) and (2, 1.1) in Figure 23 (b) into one edge (2, 1.0), as shown in Figure 23(c), and merge edge (5, 1.0) and (5, 1.2) in Figure 23(b) into edge (5, 1.0), as show in Figure 23(c). The infinite face (2) adjacent to the face (1.0) represents the whole non-convex orthogonal face rather than the face (1.1) as a sub- face (split face of main face (1.0)), therefore the edge (2, 1.0) is used to merge edges.

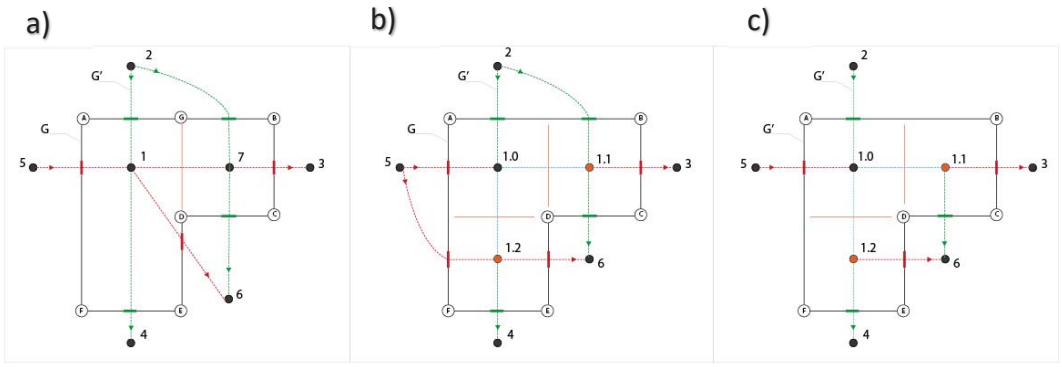


Figure 23 Steps finding identical names representing different dimensions in DSG from Dual Graph for a non-convex orthogonal face. (a) Split non-convex orthogonal face into two convex orthogonal faces. (b) Split non-convex orthogonal face into three convex orthogonal faces. (c) Split non-convex orthogonal face into three convex orthogonal faces and merging of adjacencies edge (2, 1.0) and edge (2, 1.1) and also adjacency edge (5, 1.0) and edge (5, 1.2) in Dual Graph G'.

Next, we identify the vertex in CDSG for naming the edges in DSG First, we consider the dimensions for the y-direction (see Figure 23 (b)), the first dimension corresponding to the path (2, 1.0, 1.2, 4). Only edge (2, 1.0) and (1.2, 4), which intersects the Plan Graph G, will generate cut lines. The edge (1.0, 1.2) is between split faces will not generate a cut line.

The two cut lines or edge (A, B) and edge (F, E) of the Plan Graph will be turned into vertices in the DSG and the vertices 1.0-1.2 are corresponding to the edge in the DSG., which is the identical name that represents a dimension of the Planar Graph G (see Figure 24(e)).

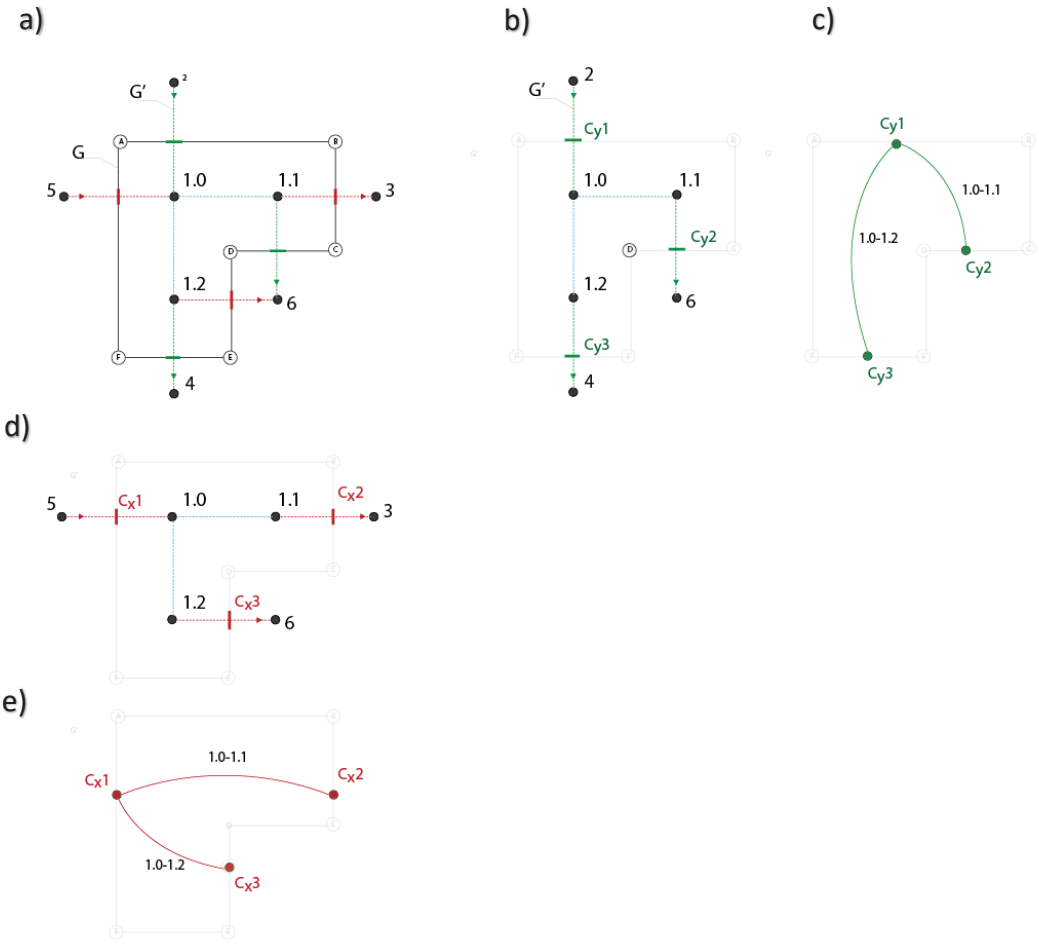


Figure 24 Assigning identical names representing different dimensions in DSG for non-convex orthogonal face. (a) Non-convex orthogonal face with Dual Graph G'. (b) Dual Sub-Graph (correspond to CDSG) in y-direction. (c) DSG for y-dimension. (d) Dual Sub-Graph (correspond to CDSG) in x-direction. (e) DSG for x-dimension.

In Figure 24(b), it is showing the CDSG for the y-direction which has two Dipaths. Dipaths or "*directed path is a sequence of edges which connects a sequence of vertices, but with the added restriction that the edges should all be directed in the same direction*" (Wikipedia, n.d.). This will be turned into a DSG, as shown in Figure 24(d). In Figure 24(b), the DSG has a vertex (C_y1) as a source and two vertices, vertex (C_y3) and vertex (C_y2), as a sink which connects to the source vertex by two edges representing the two y-dimensions of this face. This graph can be call Bipartite Graph, with two sets of vertices which join each other (Aldous, 2005). This type of graph will be used to represent each the x and y dimension of a non-convex orthogonal face, as show in Figure 24(c) and (e). More examples of Bipartite Graphs which are representing dimensions for non-convex orthogonal face are shown in Figure 25(a) to (d), where the dark blue vertex represents the first set of vertices and the green vertex represent the second set of vertices. This graph structure can only represent the L-shape or similar, but not the U-shape as show in Figure 25(e).

In Figure 25 it shows in the second column a way to split a non-convex face (as imaginary sub-face to name the edge in DSGs). Each pair of smaller edges will define the split faces marked in grey dash lines. The width and length of each face will be equal to their smaller edges. Figure 25 (c), in the second row are the edges oriented in the y-direction marked in orange and these edges will be separated in to two sets, one is the set of the longer edges and the second is the set of the smaller edges. These sets are corresponding to the vertices in DSG colored in dark blue and green (see Figure 25 in the last row). So, we have two subsets of vertex/vertices for this DSG and each edge of this graph joins the vertex in both sub-sets a so call Bipartite Graph, therefore we will use the Bipartite Graph to represent DSG for non-convex face.

The name of edges of DSG corresponds to the pair of intermediate vertices of each path in CDSG, as shown in Figure 25(c) at the second row.

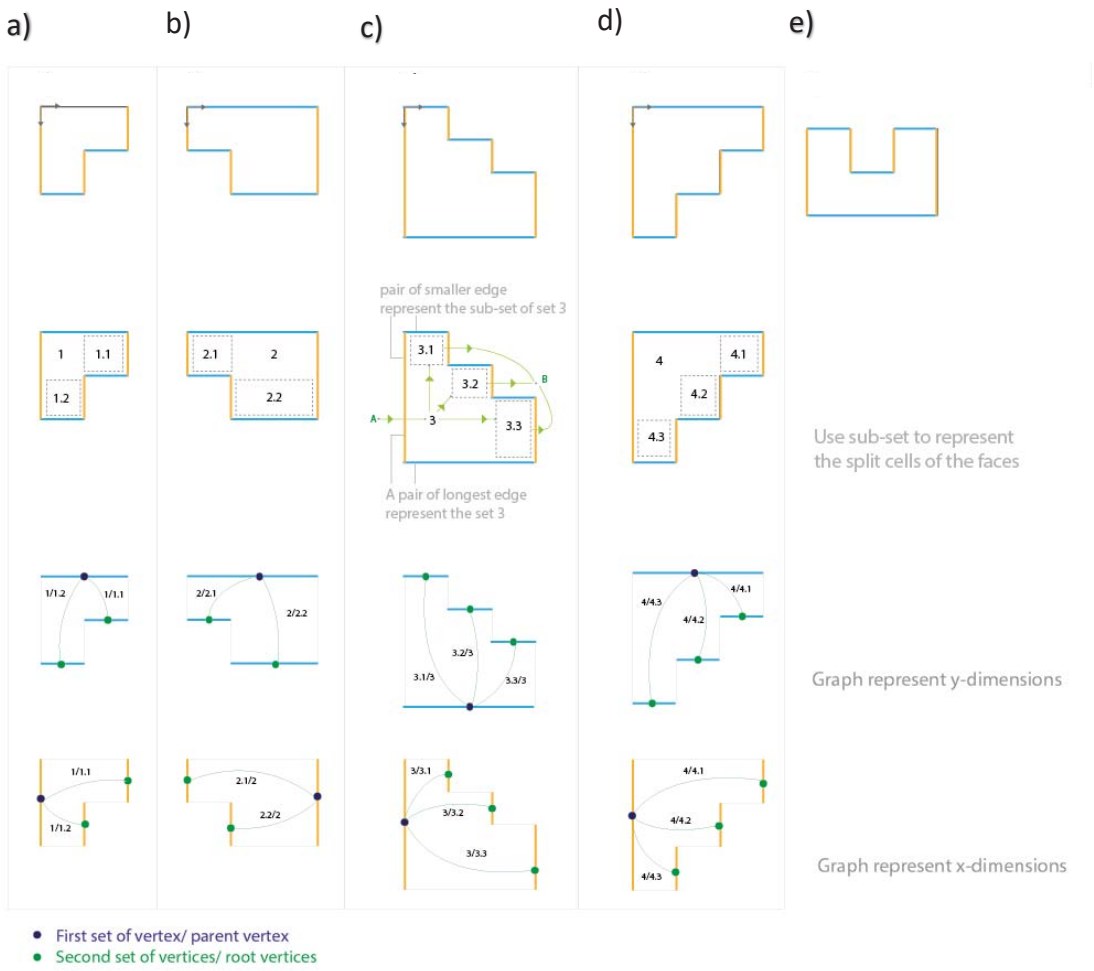


Figure 25 The non-convex orthogonal faces with different shapes and their Bipartite Graphs as DSG represent dimensions. (a) to (b) different shapes of non-convex orthogonal faces with split faces (as sub-set) and their corresponding DSG for both x and y-dimension. (e) A non-convex orthogonal face with U-shape.

5.4.3 Sets and graph data representing non-convex orthogonal face

Figure 26(b), each of this DSG have a character which can be defined as Bipartite Graph. This is a graph whose set of vertices can be split into two subsets (A) and (B) in such a way that each edge of the graph joins a vertex in A and a vertex in B (Aldous, 2005). According to the Bipartite Graph we would set (cy1) (see Figure 26 (b)) as a set of (A) and (cy2) and (cy3) as a set of (B), and the vertex in set (A) will connects to all vertices in the set (B). We can also define set (A) as the set of source-vertices and set B as sink vertices and each edge which connects between these two sets of vertices "source and sink" which will represent the dimensions of the face (Aldous, 2005).

In Figure 26 (b) Let's

A_y = {Cy1} Ay is set of source vertex in y-direction.

B_y = {Cy3, Cy2} By is set of sink vertices in y-direction.

The source refers to the starting point of the measurement and sink as the end point of the measurement of the face.

Figure 26(b) is showing two Bipartite Graphs. In the first graph for y direction, the vertex in set (Ay) is connected with vertices in the set (By) and the edges are connected with these two sets of vertices which represent the dimensions for face (1) in the y-direction.

So, each set of opposite walls (edge of plan graph mark in green) have dimensions which are represented by the edge of the DSG marked in green.

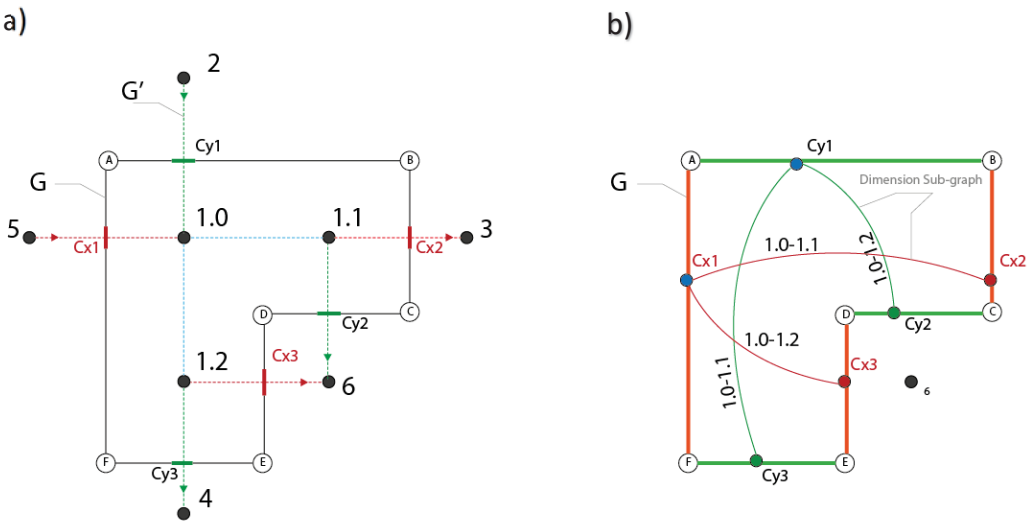


Figure 26 Graphs representing non-convex orthogonal face (a) Non-convex orthogonal face with Dual Graph G'. (b) DSG for non-convex orthogonal face in both x and y direction.

5.5 Turning the unit plan with non-convex orthogonal faces into graphs with a Dual Graph Method

In Figure 27 it is showing all the graphs which represent the floor plan with non-convex orthogonal faces. The blue dash line in Figure 27(a), (b) and (c) represents the edges connected between the interior vertices of a non-convex orthogonal face. This edge will not intersect any edge of plan graph, so it will not represent any wall of a Plan Graph.

In Figure 28 it is showing the comparison between two AGs, one at left for convex orthogonal face and right for both convex orthogonal and non-convex orthogonal face. The AG for both convex orthogonal and non-convex orthogonal faces will be used in this study for Layout editing operations.

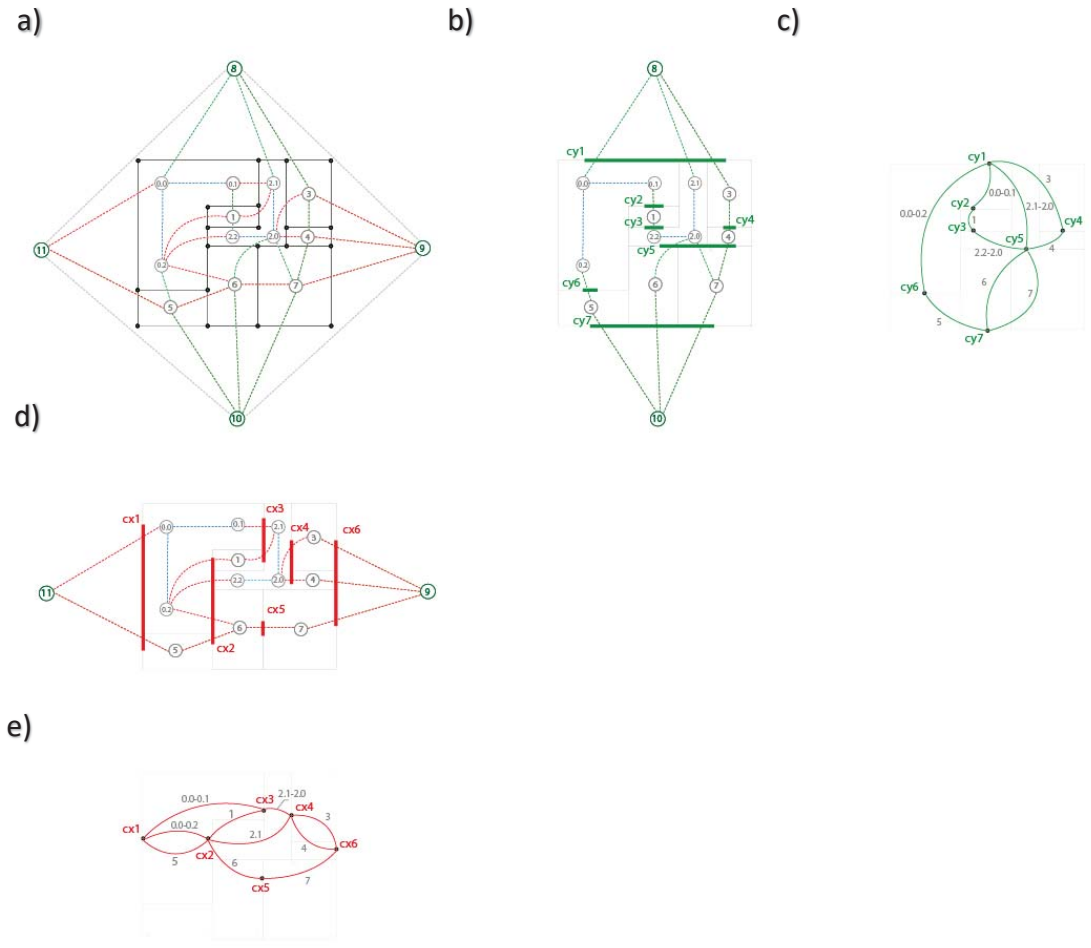


Figure 27 Turning the floor plan with non-convex orthogonal faces and convex orthogonal faces into graphs with a Dual Graph method. (a) PG and its Dual Graph for convex orthogonal face and non-convex orthogonal faces. (b) CDSG in y-direction. (c) DSG for y-dimension. (d) CDSG in x-direction. (e) DSG for x-dimension.

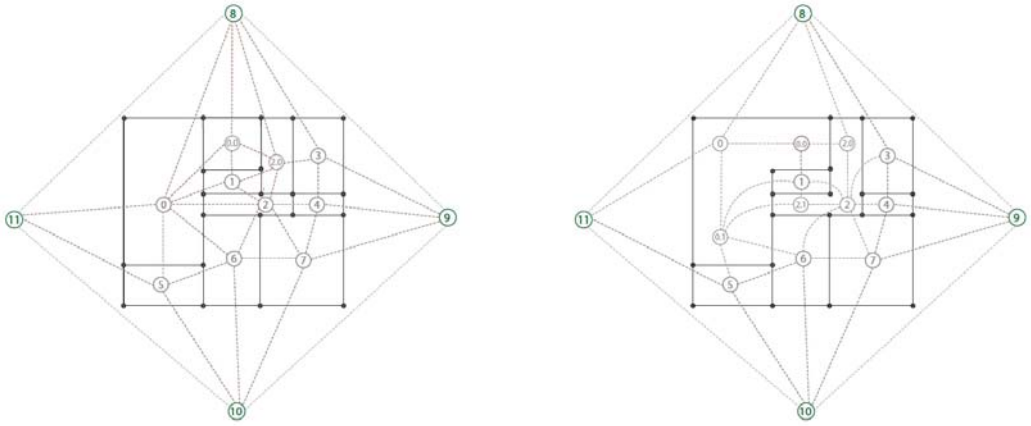


Figure 28 Comparison between two AGs, one at left for convex orthogonal face and right for both convex orthogonal and non-convex orthogonal face.

5.6 Sets of vertices representing the walls junction

In this chapter we look at the relationship between a PG and a DSG which is a fundamental part of graph-based floor plan representation. The vertices of the DSG corresponding to a specific set of vertices in the PG are representing a common wall. Sets of vertices will be used to assign the x and y coordinates relating to the dimensions of faces in the unit plan in "Layout editing operation".

In the graph theory, a graph consists of a set of points, called vertices, joined by lines, called edges; each edge joins exactly two vertices (Aldous, 2005) This can be written G = {V, E}, a G which denotes a graph where V denotes Vertices and E denotes Edges. In Figure 29(a) is shown a PG representing a source unit plan, which consist of a set of vertices and a set of edges as follows.

V = {A, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U}

E = {(A, C), (C, D), (D, E), (F,G), (H, I), (J, K), (L, M), (M, N), (N, O), (P, Q), (R, S), (S, T), (T,U), (A, P), (P, R), (F, H), (H, L), (L, Q), (Q, S), (C, G), (G, I), (M, T), (D, J), (J, N), (E, K), (K, O), (O, U)}

The sets of vertices representing each a wall junction of a unit plan will be created from this PG. The sets will be created based on the relationships between the PG and the DSGs as show in Figure 29. These sets will define the dimension constraint of the unit plan.

The sets of vertices will be created from all vertices in the PG. Each of the vertices in the DSG for the x-dimension (see Figure 29(c)) will have a set of corresponding vertices in the Plan Graph (see Figure 29(a)) representing the same y-coordinate. For example in Figure 29(b), the vertex (Cy1) in the DSG for the y-direction is the name of the set of the corresponding vertices

in the PG, as shown in Figure 29(a) so the set $V(C_y1) = \{A, C, D, E\}$ is a set representing a wall junction in the x-direction. They have the same value of the y-coordinate.

The vertices in DSGs will be used to name the sub-sets of vertices in PG which are representing the common wall. These sub-sets also represent the x or y-coordinate in the PG, where all the vertices in each sub-set have the same x or y-coordinate value. To summarize the vertices in DSGs for the y-dimension are representing the y-coordinate of common x-wall in the PG, and the y-coordinate will define the y-dimension of the face.

Below are sets of corresponding vertices in the PG representing the x value.

$V(C_x1) = \{A, P, R\}$ and $X_{Cx1} = \{x: \text{coordinate x of vertex and } x \in V(Cx1)\}$ 3.1

$V(C_x2) = \{F, H, L, Q, S\}$ and $X_{Cx2} = \{x: \text{coordinate x of vertex and } x \in V(Cx2)\}$ 3.2

$V(C_x3) = \{C, G, I\}$ and $X_{Cx3} = \{x: \text{coordinate x of vertex and } x \in V(Cx3)\}$ 3.3

$V(C_x4) = \{D, J, N\}$ and $X_{Cx4} = \{x: \text{coordinate x of vertex and } x \in V(Cx4)\}$ 3.4

$V(C_x5) = \{M, T\}$ and $X_{Cx5} = \{x: \text{coordinate x of vertex and } x \in V(Cx5)\}$ 3.5

$V(C_x6) = \{E, K, O, U\}$ and $X_{Cx6} = \{x: \text{coordinate x of vertex and } x \in V(Cx6)\}$ 3.6

Below are sets of corresponding vertices in the PG representing the y value.

$C_y1 = \{A, C, D, E\}$ and $Y_{Cy1} = \{y: \text{coordinate y of vertex and } y \in V(Cy1)\}$ 4.1

$C_y2 = \{F, G\}$ and $Y_{Cy1} = \{y: \text{coordinate y of vertex and } y \in V(Cy1)\}$ 4.2

$C_y3 = \{H, I\}$ and $Y_{Cy1} = \{y: \text{coordinate y of vertex and } y \in V(Cy1)\}$ 4.3

$C_y4 = \{J, K\}$ and $Y_{Cy1} = \{y: \text{coordinate y of vertex and } y \in V(Cy1)\}$ 4.4

$C_y5 = \{L, M, N, O\}$ and $Y_{Cy1} = \{y: \text{coordinate y of vertex and } y \in V(Cy1)\}$ 4.5

$C_y6 = \{P, Q\}$ and $Y_{Cy1} = \{y: \text{coordinate y of vertex and } y \in V(Cy1)\}$ 4.6

$C_y7 = \{R, S, T, U\}$ and $Y_{Cy1} = \{y: \text{coordinate y of vertex and } y \in V(Cy1)\}$ 4.7

The vertex (C_y6) (see Figure 29(b)) is a set which represents the wall junction (Figure 29(a)), in which are contained the vertices (P), (Q) and each of these vertices have the same y-coordinate value equal to 9.18. The edges of DSGs will also relate to the coordinate of vertices in DSG. The “Layout editing operation” relating to dimensions will be based on these coordinates.

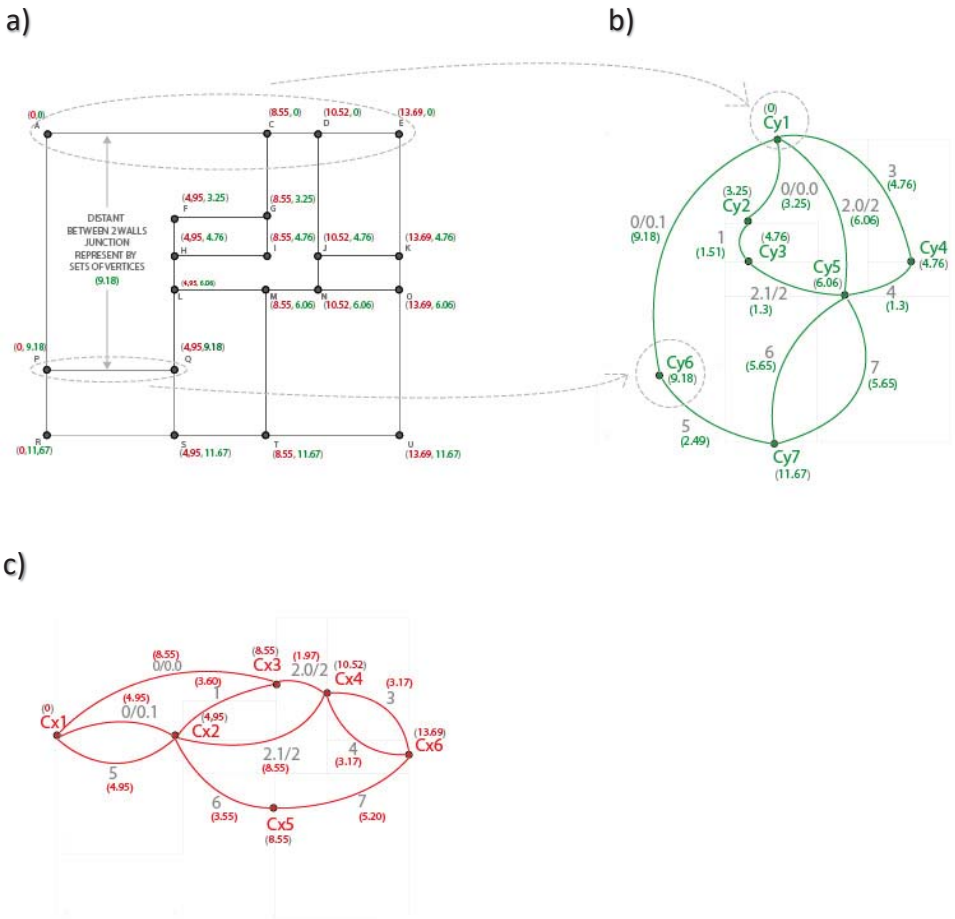


Figure 29 The coordinate of vertices in PG corresponding to coordinate in DSG. (a) PG with coordinate of each vertex. (b) DSG for y-dimension (c) DSG for x-dimension.

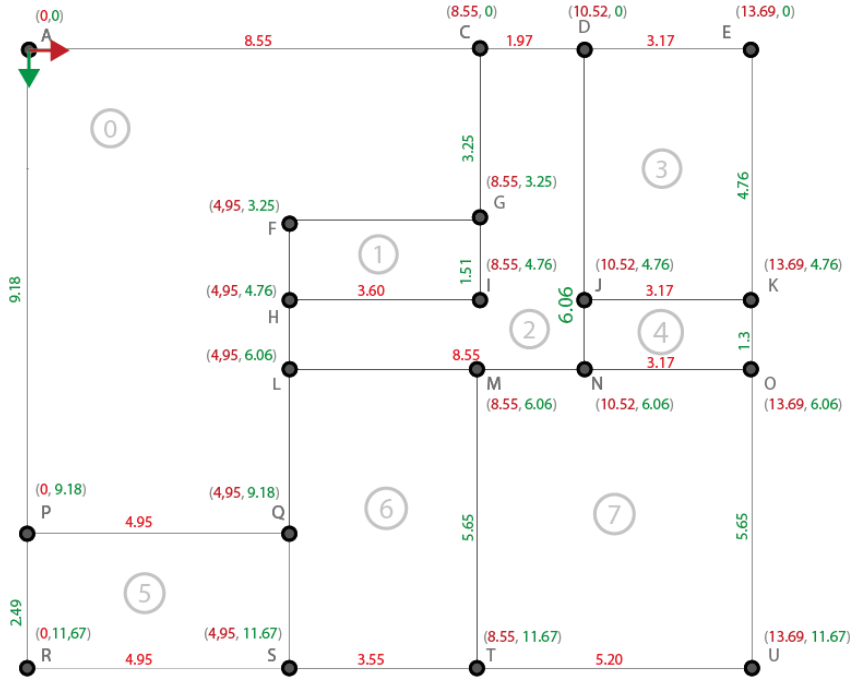


Figure 30 Planar graph with coordinate of vertices.

5.7 Turning unit plan with quadrilateral boundary into Dimension Sub-Graph

Figure 31(b) is showing the PGs with quadrilateral unit boundary meaning that the face (1) is attached to the orthogonal edge of this unit boundary and it has a quadrilateral shape. In the DSG (see Figure 31(d)) for a convex orthogonal face an edge represents a dimension of a face and this dimension is the distance between two parallel edges (B, E) and (C, F). There is only one value assigned to the edge which is different from the quadrilateral face (Figure 31(c)) and which need two dimensions to represent this face.

The coordinates of vertices are used to find the dimension of quadrilateral faces. The unit boundary with quadrilateral shape has two parallel edges (A, C) and (G, I), they have different length. The difference between these two lengths will be used to find the x-coordinate for the two end points of orthogonal edge (C, F) as show in Figure 31 (c) and (e). Computation of the coordinate is based on the ratio of boundary dimension and the face dimension as show in Figure 31 (c).

As a result (see Figure 31 (c)), there are two different dimensions representing two edges (B, C) and (E, F) being assigned to the edge of DSG for x-dimension.

In Figure 32(d) it shows the DSG representing a quadrilateral unit plan where edge (3_x) and edge (1_x) have different dimensions, because they start at the same vertex, therefore they have a different vertex as the sink vertex. As a vertex of DSG represents a common wall for

rectangular unit plans (see Figure 32(a) and (c)). The DSG for quadrilateral unit plans will have more than one vertex representing either sources or sinks, and each of the vertices represents an orthogonal wall of a face. If there are four quadrilateral faces, then there will be four vertices as sinks representing each an orthogonal wall of each face.

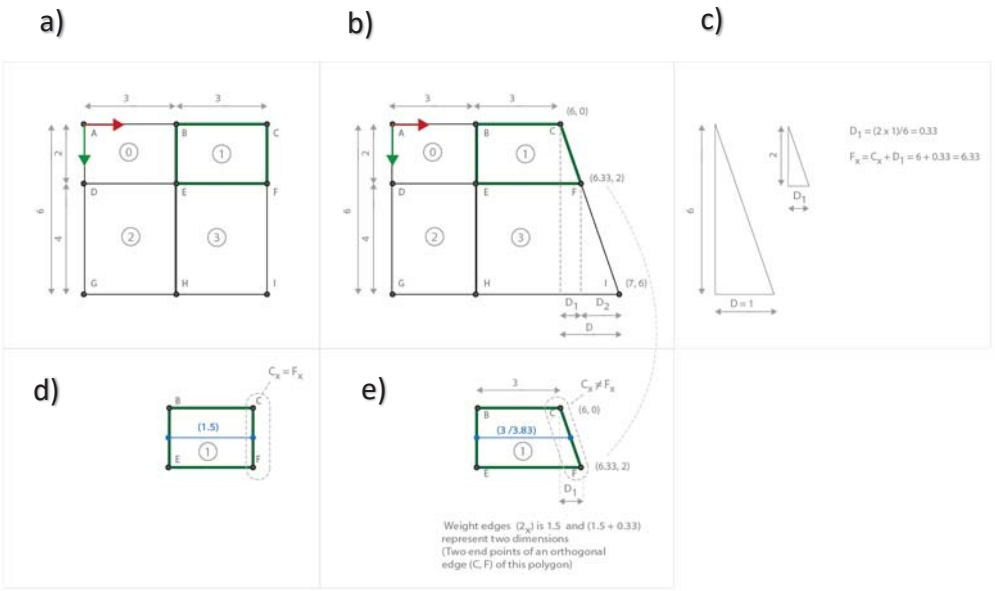


Figure 31 The steps turning a unit plan with quadrilateral unit boundary into DSG. (a) PG of a unit plan with rectangular unit boundary. (b) PG of a unit plan with quadrilateral unit boundary. (c) Computing of x-coordinate of end vertices of each an orthogonal edge representing each a wall segment of each face. (d) The face (1) with rectangular shape and its DSG with assigning one dimension for the edge. The face (1) with quadrilateral shape and its DSG with assigning two dimensions on the edge.

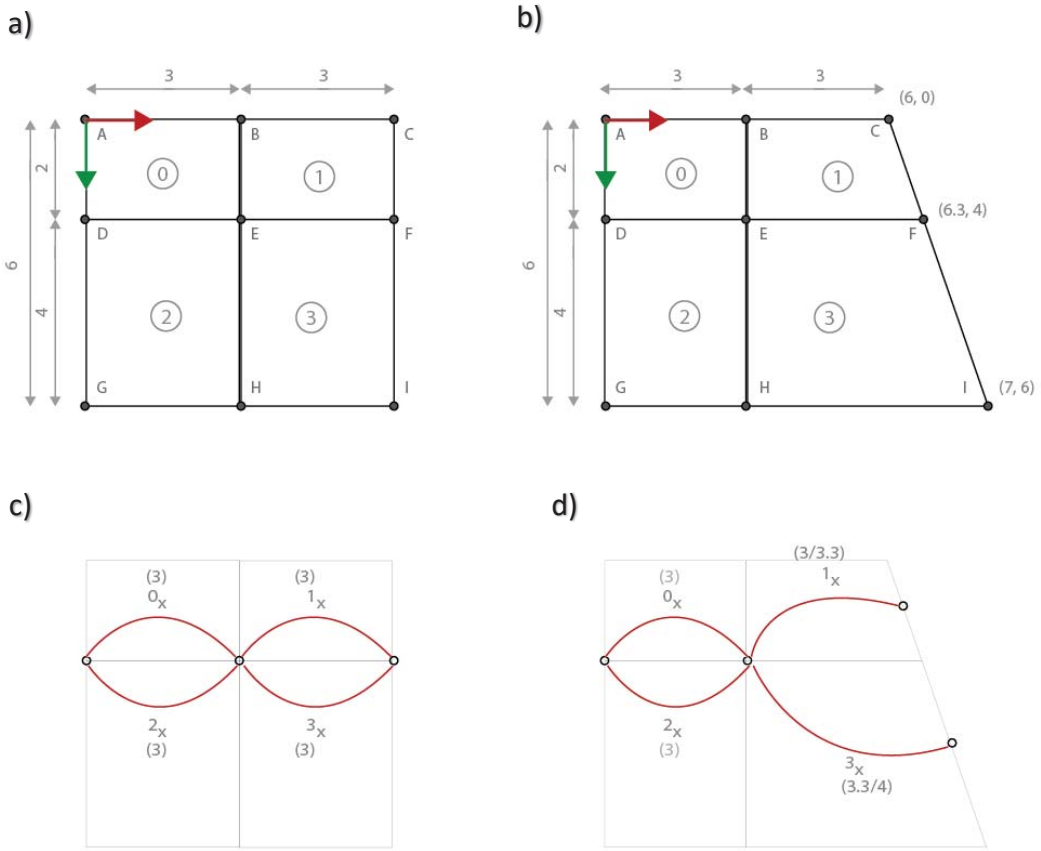


Figure 32 The PGs and DSGs of a unit plan with Rectangular boundary and a unit plan with quadrilateral boundary. (a) A PG of a unit plan with Rectangular boundary (b) A PG of a unit plan with quadrilateral boundary. (c) A DSG for x-dimension of a unit plan with Rectangular boundary. (d) A DSG for x-dimension of a unit plan with quadrilateral boundary.

5.8 Turning unit plan with non-convex orthogonal unit boundary into Dimension Sub-Graph

Figure 33(b) is showing the unit plan with a non-convex orthogonal unit boundary and its DSG, as shown in Figure 33(d). The DSG of the unit plan with non-convex orthogonal boundary is similar to the DSG of the unit plan with quadrilateral boundary, where they have more than one source or sink representing boundary walls. The source and sink of DSG for the unit plan of a non-convex orthogonal boundary represents a wall segment of each face which share a common wall with the unit plan boundary. There is only one value assigned to each an edge of a DSG which is different from the DSG for the unit plan with a quadrilateral boundary where two values are assigned to one single edge.

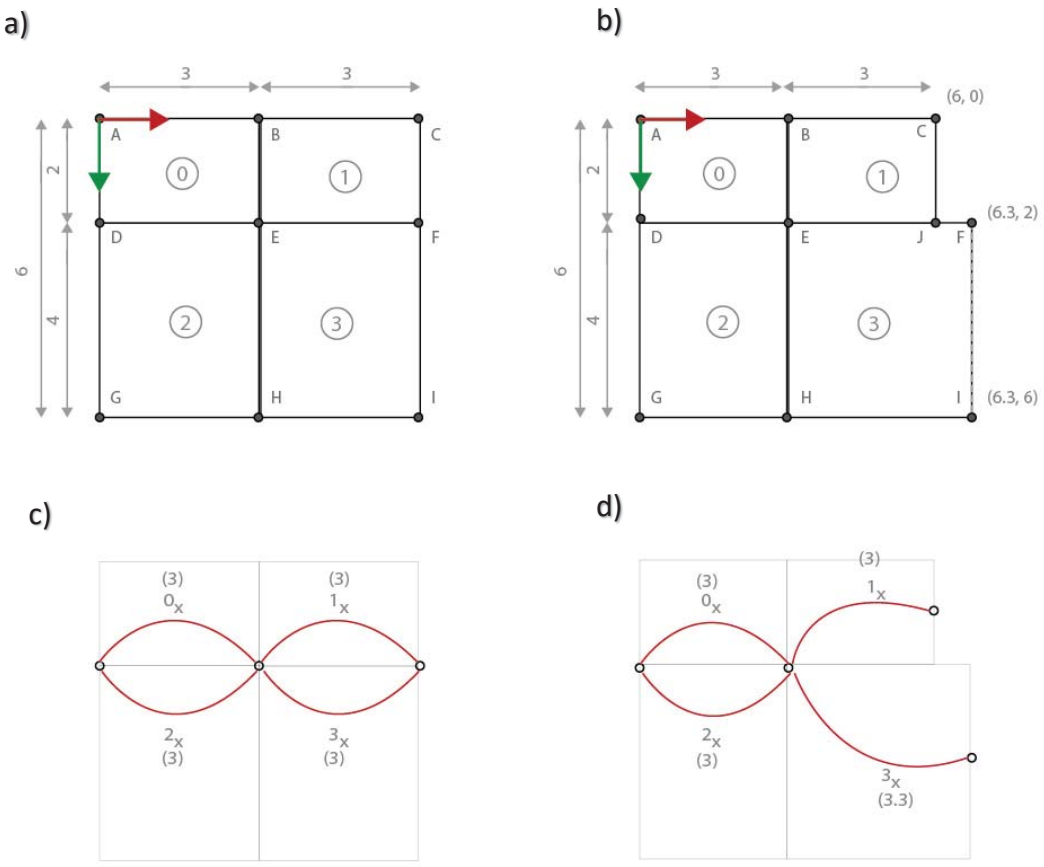


Figure 33 The PGs and DSGs of a unit plan with Rectangular boundary and a unit plan with non-convex orthogonal boundary. (a) A PG of a unit plan with Rectangular boundary (b) A PG of a unit plan with non-convex orthogonal boundary. (c) A DSG for x-dimension of a unit plan with Rectangular boundary. (d) A DSG for x-dimension of a unit plan with non-convex orthogonal boundary.

6 PROCESSING OF LAYOUT EDITING OPERATIONS

6.1 Overview

A graph-based floor plan representation is used as a fundamental tool in editing operations. The insertion of the room topology to a target unit boundary with different dimensions will cause changing of all room's dimensions, this may violate some room dimensions criterion, for example corridor, bath room or storage room. To adjust some specific rooms to meet the expected criterion without a graph-based floor plan representation, the designer might change the dimension of those specific face, and as result the topology of the target unit plan in terms of spatial structure and the room adjacency are changed. This may not be a desired outcome. The graph-based floor plan representation is introduced to address this issue, it is used to defined sets of constraints for the layout editing operation.

1. Inserting input for the Layout editing operation
2. The scale factors are computed. These are needed to scale the source unit plan to fit into the target unit boundary. This step is called "Scale-based of source unit plan in target boundary" is the essential basics for all the editing operations.
3. The specific type of editing operation is defined by designer.
4. Plan graph generation.
5. Computing the Dual Graph and CDG represents the adjacency between rooms.
6. CDSGs are generated.
7. DSGs are generated.
8. "Scale-based of source unit plan in target boundary" where the scaling of a source unit plan to fit into the target unit boundary. This is the first step for all types of insertion of room topology in a target unit boundary.
9. Generation the sets of corresponding vertices from the PG and the DSGs.
10. The specific type of "Insertion of rooms in a target boundary" is computed. Moving a wall editing operation is done by designer. This editing operation is used for manual assigning of dimension constraint of the room and monitoring all other possible change occurs to the other rooms which shared the common wall to this room. If there are any further requirements in terms of small configuration in either the source unit plan or in the target unit plan, then the "Moving a wall" editing operation can be made optional by a designer. If not, this step will be skipped.

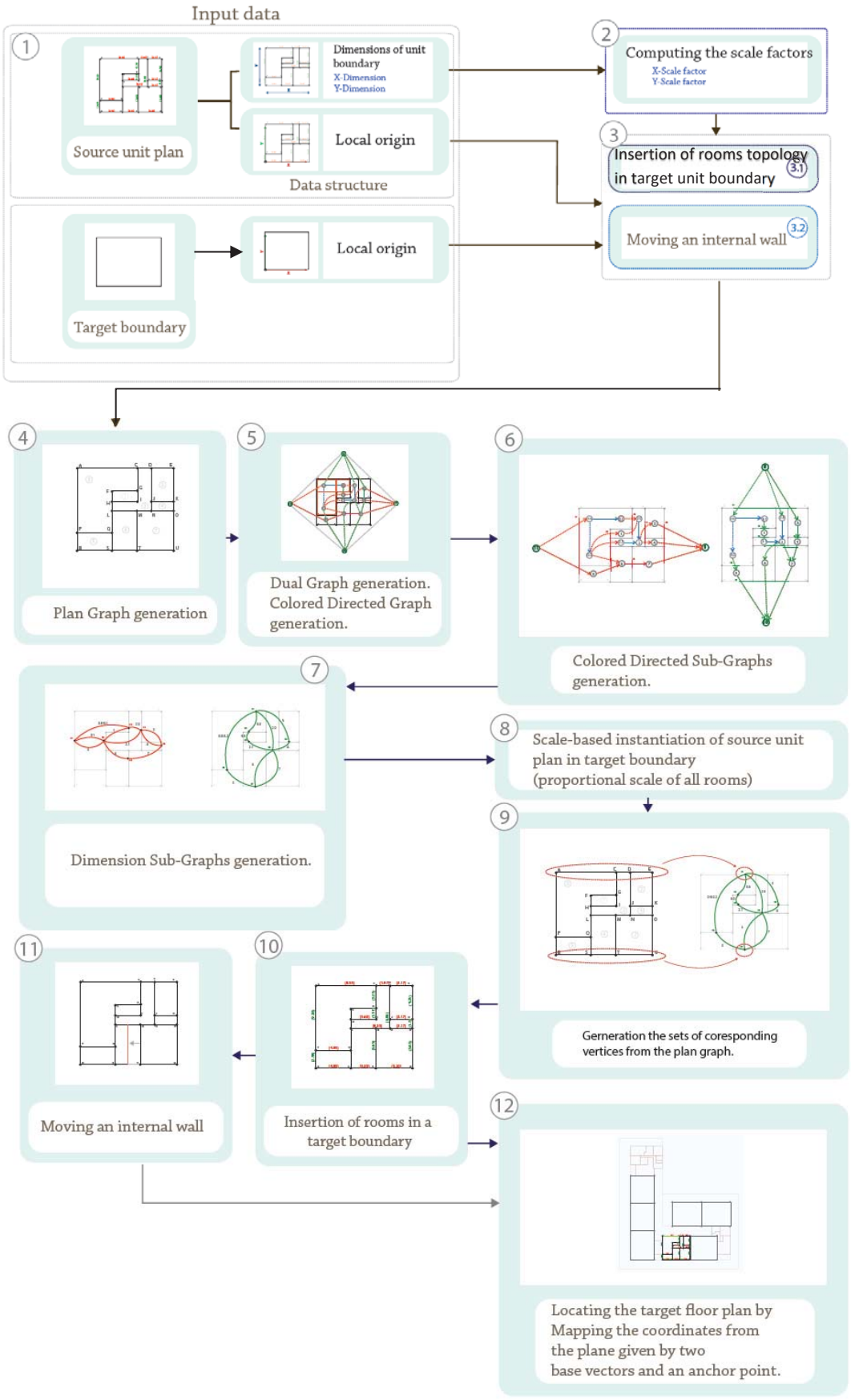


Figure 34 Flowchart shows steps required for the editing operations.

The step 10 and 11, as illustrated in Figure 34 shows the Layout editing operations, the first one is “Insertion of rooms in a target ” and the second one is “Moving a wall”. Each will be described in the following chapters.

6.2 Insertion of rooms in a target unit boundary

There are required steps of operations to place the source unit plan into the target unit boundary, one of the essential steps for this is the “Scale-based of source unit plan in target boundary” as a part of the steps of the editing operation, as show in step 8 in Figure 34 which is to scale the source unit plan so to adjust with the target unit boundary without yet determining the dimension constraint of the rooms. This step together with the determining of the dimension constraint of the rooms applies to all types of unit boundaries which follow subsequently.

6.2.1 Rectangular unit boundary

In this Layout editing operation some of the rooms will maintain their original dimension, which is defined by the designer as a constraint (in the source unit plan) and the dimensions of other rooms will be changed after scaling. The following are the steps explaining how this method works and how to handle the constraints.

A preference corner on the source unit plan is used as the original coordination point of the plane. This point defines which room will maintain its original dimension.

This editing operation might be useful or recommended for some apartment building projects which have similar situation as follow. As giving example, to scale unit plan template, for those rooms which maintain their size are WC, bath room, vertical duct. A unit plan we see that two ducts of two apartment rooms are joined so the bath room and WC need to be oriented to this duct. The bath rooms of both apartments have two orientations one oriented to the other apartment and the other oriented to the corridor, which will be used for the orientation of the rooms which will maintain their size. This method will be also appropriate for any project especially apartment buildings, where some of apartment units might share the same vertical duct, vertical circulation and services.

Figure 35 is the example of revert scaling of the room no. 6 (marked in red) for each target floor plan. Only room 6 in all floor plans has changed after rescaling, and all of them have the same size as in the prototype.

There are some issues about topology and geometry of some of the source unit plans, for example the source unit plan 3-3 at the right bottom corner has the right wall of room (1) and room (2) which are not aligned to each other in the vertical axis like in the source unit plan.

This also causes the change of adjacency between rooms, for example room (1) now has not only the wall aligned to the vertical axis adjacent to room (7) but also has the wall aligned to the horizontal axis adjacent to the same room, which is different from the prototype floor plan.

The solution to this problem is using the dimension subgraphs to control the constraint of the floor plan representation, which will be explained in a further chapter. In Figure 36 it is showing the result of the rescaling of the room (6) based on the graph constraint of the floor plan representation. As the result, the adjacency between rooms and topology, a common wall, is maintained as in the prototype but some rooms have changed in size. For example, room (1) for every target floor plan (marked in red) has the same width as in the room (6), because room (1) and room (6) share a common wall, so this wall has the length of 1,5 and the length of this wall is corresponding to the width of room (1) and room (6).

Because the width of room (1) has been adjusted to the width of room (6) and it causes the room (6) to push or pull all the rooms next to it and the rooms in the last section (marked in red) will be changed, either they shrink or stretch. These rooms need to be changed in order to fit into the fixed target unit boundary.

The rooms are arranged in sequence from left to right, the effect of one change might have an impact to the others in this sequence, either less or more. In the designer perspective we would like to control this change by setting up the constraint to minimize the effect which is caused by a change. The DOF will also be one part of the constraints for the dimension subgraph of the floor plan representation to tackle this issue and this constraint is mentioned in the chapter “Layout editing operations”.

In Figure 37, it is showing another example of rescaling of rooms in the target unit boundary. In this example all rooms are scaled. The dash grey rectangle is marked to show the original source unit boundary.

The scaling method might have some issue with the dimension of rooms, for which designers would like to maintain the dimensions as in the source unit plan, for example the size of the corridor or storage room. In the example (see in Figure 40) it is shown how the scale factors have an impact on the shape of the rooms in the target unit plan, the room (6) is marked in red in every target unit plan and is changed. This might not be the expected outcome for the designer.

Designers might want to keep the size of these rooms small or as the same as the source unit plan, even if all the other rooms in the target unit plan are changed in size after scaling.

The designer's preference referring to the dimension of these rooms in the target units would be taken in consideration in detail in this study in chapter "Layout editing operation and the relating constraints", which refers to different types of operations and how to set up the constraints.

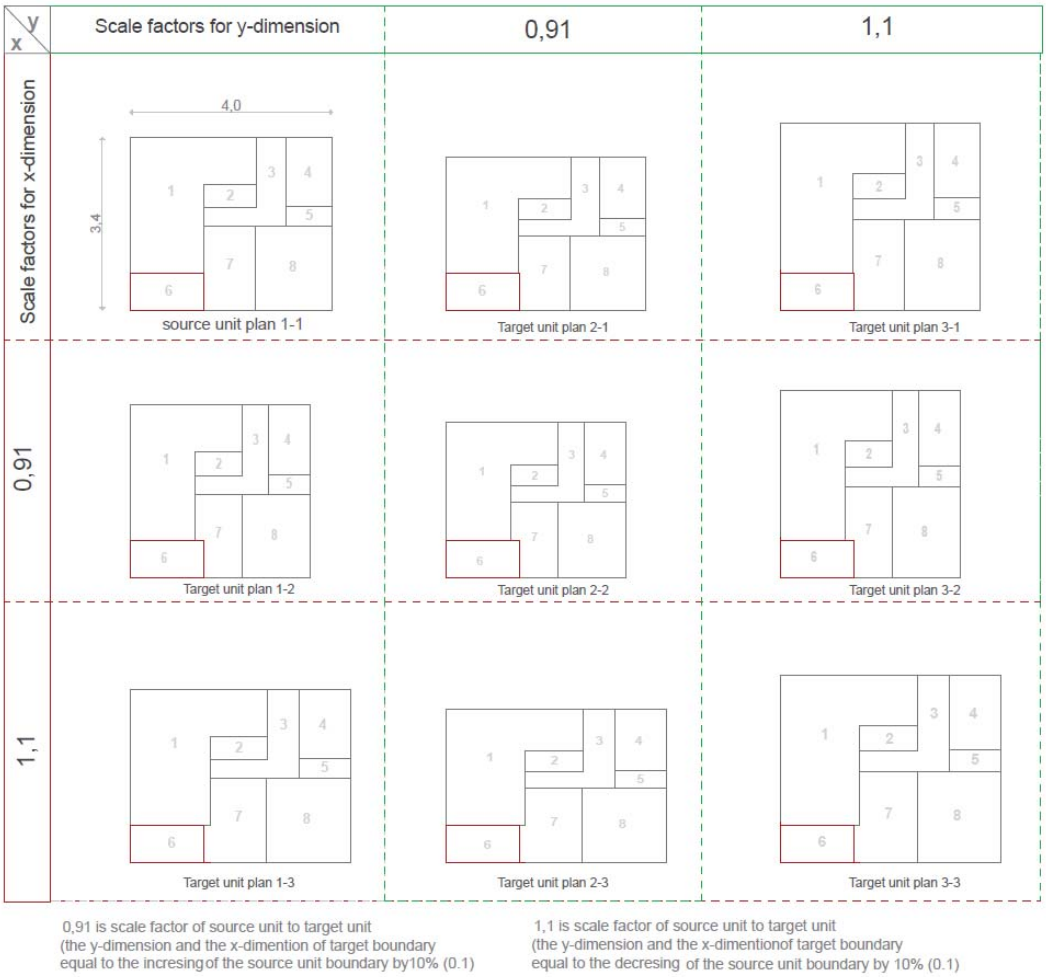


Figure 35 Results of the rescaling of room 6 (to maintain its original size as in prototype floor plan) in each target floor plan without the adjusing of all other rooms, becaause of the rescaling is not based on graph based floor plan representation.

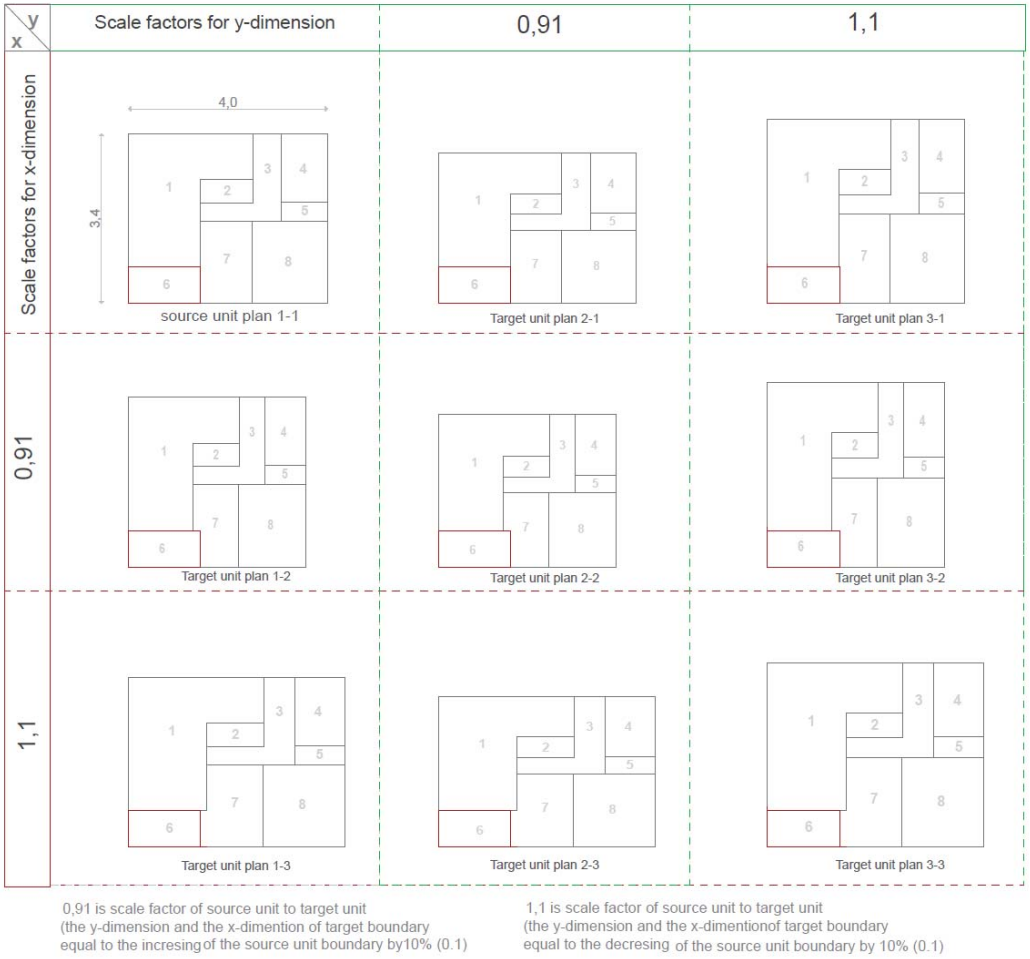


Figure 36 Results of the rescaling of room (6) in each target floor plan, and the adaptation of other rooms to maintaint their original topology, based on the graph based floor plan representation method.

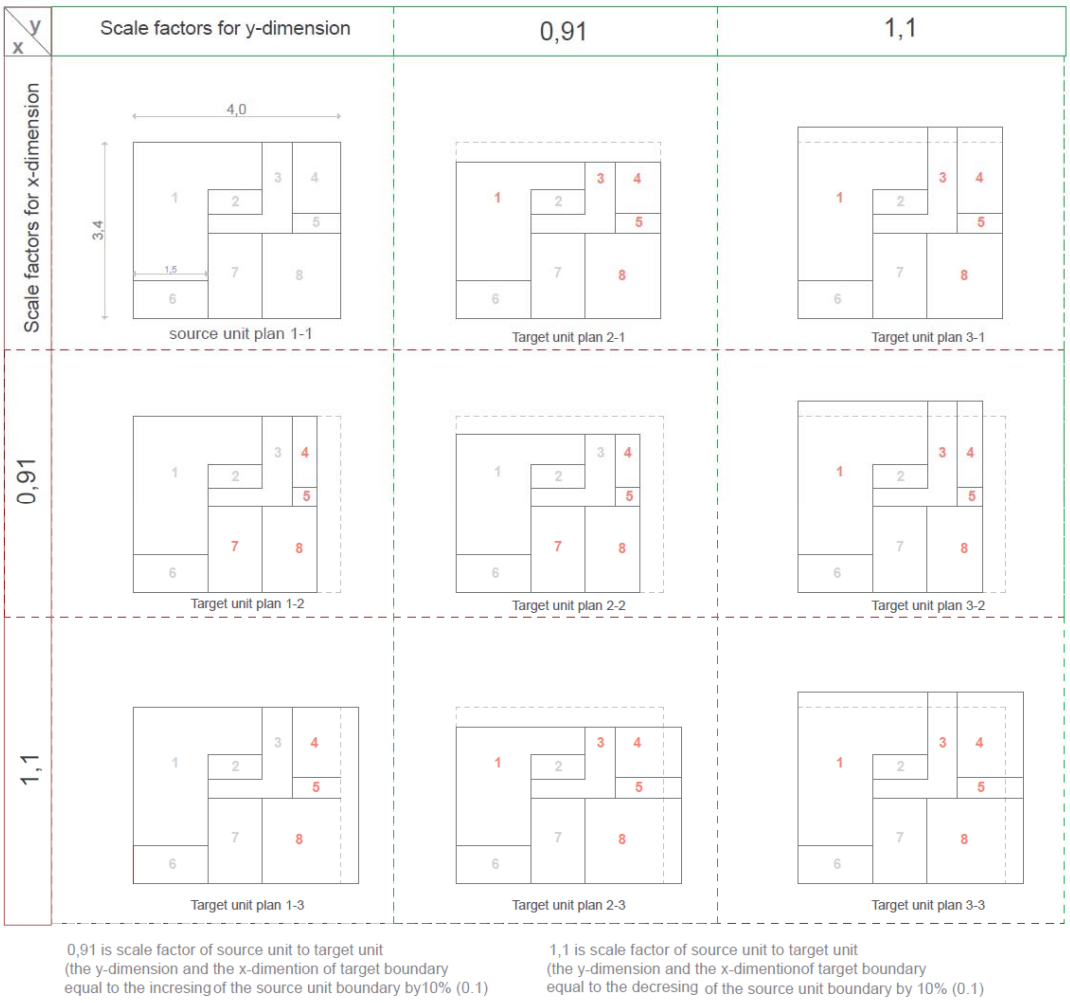


Figure 37 Results of rescaling of all the rooms (to maintain its original size as in prototype floor plan) in each target floor plan.

The steps for this operation will begin with “Scale-based of source unit plan in target boundary”. This operation is the fundamental operation for most variations of “insertion of rooms topology in a target boundary” editing operation. The possible outcomes related to the scaling factor.

We will use “A floor plan of an apartment unit from the Housing Complex by Hermann and Johannes Kaufmann architects” as case study to show different results of scaling which were influenced by scale factors.

Figure 40 is showing the floor plans where the wall thickness and doors and windows were ignored, and the walls are represented by lines. The source unit plan 1-1 (on top left) is the original unit plan of the Housing project, all others are the different versions of the target unit plans. They are scaled proportionally. The Target unit plans (2-1), (2-2), (2-3) have a y-dimension of the boundary equal to 3,1 meters and this dimension is equal to the y-dimension

of source unit boundary and decreased by 10%. The same operation will also occur in the other direction meaning the x-direction (row wise).

Each floor plan is unique, each of them multiply by different scale factors for both x-dimension (row wise) and y dimension (column wise).

As result we could sperate the target unit plans into two groups. The first group are the floor plans, which are not distorted by the different values of the scale factors for the x and y dimension.

If the scale factor for the x-dimension is equal to the y-dimension, then the target unit plan is not distorted.

In Figure 40 the target unit plan 2-2 and the target unit plan 3-3 are not distorted. Each of them was scaled with the same value of scale factor for both the x-dimension and the y-dimension.

To be able to understand this better see in Figure 39 which shows the illustrations of two different results of scaling. Figure 39 (a) is showing a non-distorted scaling where the scaling factors for both dimension x and y have the same value. The rectangular (B) is a copy-scale of the rectangular (A) and this shape has not been distorted. In Figure 39 (b), there is a distorted shape because of scaling factors where the scale factors for the dimensions x and y are different and as a result the rectangular B is distorted.

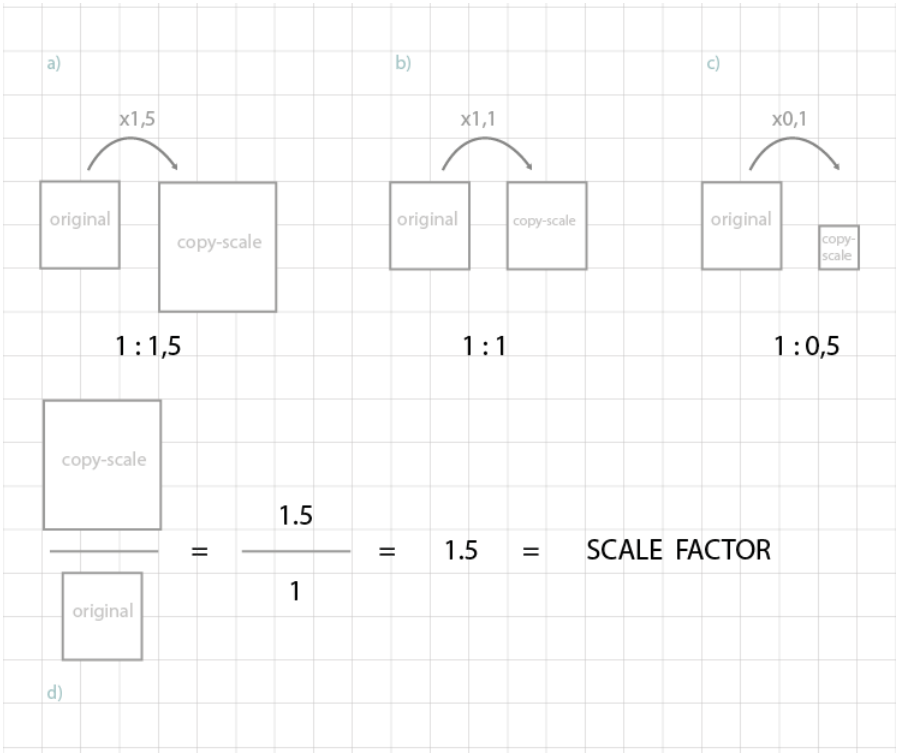


Figure 38 Scale factor and results of scaling based on the scale factors

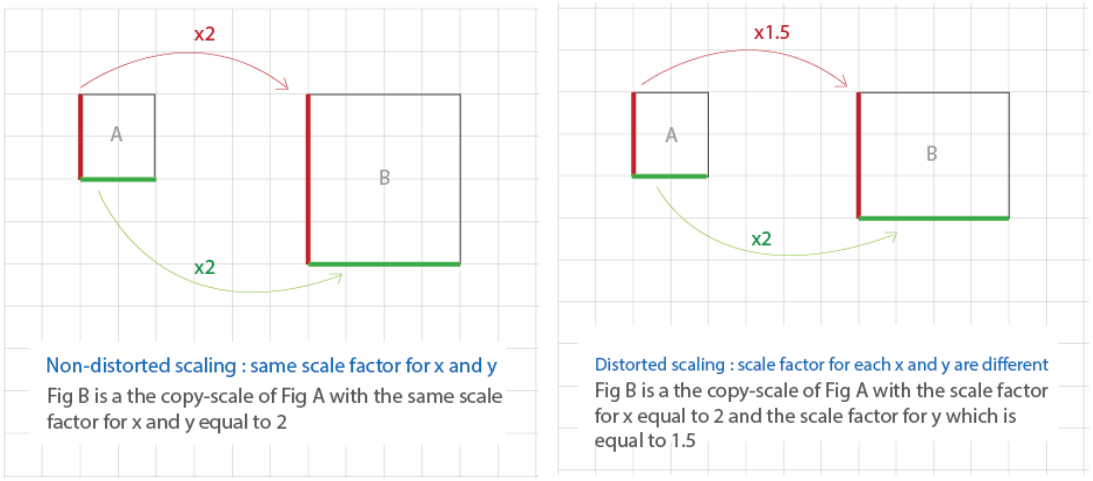
To map the prototype floor plan on a target boundary, we will apply this copy-scale operation to map the final floor plans. To scale we have to multiple the geometry with scale factors. The scale factors will be defined by the geometry, if it is going to be bigger or smaller than the original shape. Figure 38 is the illustration of the three main possible outcomes of scaling. The scale factor is the result of the ratio of the copy-scale shape to the original shape.

In Figure 38 (a) the copy-scale shape is 1,5 times bigger than original shape. The original shape multiplied by 1,5 (scale factor for x and y dimension) is equal to the copy-scale shape.

If the scale factor > 1: the copy-scale geometry is bigger than the original one. (Figure 38(a))

If the scale factor = 1: The copy scale geometry is equal to original one (Figure 38(b))

If the scale factor < 1: the copy scale geometry is smaller than original one. (Figure 38(c))



a b

Figure 39 Scale factors relate to dimensions x and y influences the distortion of the shape , Figure 39 (a) Non-distorted scaling, Figure 39 (b) Distorted scaling

Below are the formulas for calculating the scale-factors in both x-dimension and y-dimension. These will be one of the basic parameters relating to "Layout editing operations".

$$Fx = \frac{Xt}{Xp} \qquad (1)$$

$$Fy = \frac{Yt}{Yp} \qquad (2)$$

Table 1 formular-index of scale factors

Index	*Description*
Xp	$the\ X-dimension\ of\ prototype\ unit.$
Xt	$the\ X-dimension\ of\ target\ unit.$
Yp	$the\ Y-dimension\ of\ prototype\ unit.$
Yt	$the\ Y-dimension\ \ of\ target\ unit.$
Fx	$the\ scale\ factor\ for\ x\ dimention.$
Fy	$the\ scale\ factor\ for\ y\ dimention.$

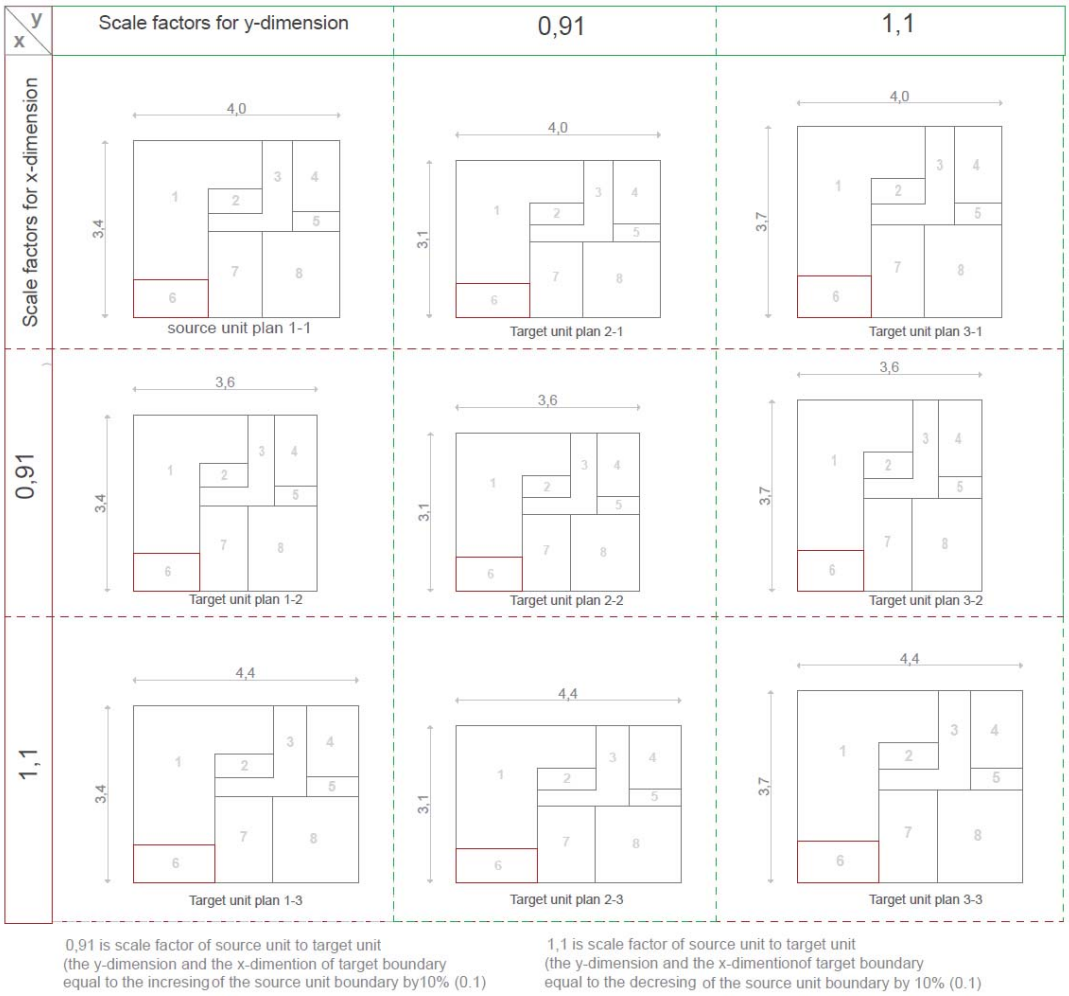


Figure 40 Different results of copy-scale of a prototype floor plan influenced by different scale factors for each x-dimension (rowwise) and y-dimension (columnwise)

This editing operation is equivalent to the operation in chapter the “Scale-based of source unit plan in target boundary” and is an operation which is an essential step for all other “categories” of all “layout editing operations”.

Basically, the dimensions of the source unit plan are represented by D.SG and will be multiplied with the scale factors. As a result, the new dimensions which are generated for DSG are representing the target unit plan.

The next step is to determine the dimension constraint of the set of rooms, which will remain their original size where the scale factor (reverts scale factors as shown Formula 3 and 4) multiplies only corresponding edges in the DSGs representing the room that will maintain its original size.

Below there is the example and an explanation in detail how this operation can be done.

In Figure 41(b) and (c) are showing all Dipaths which are representing the dimensions of the floor plan; for example in Figure 41(b) the pink dash line has been marked along the Dipath (C1), (C4), (C5), (C7), corresponding to the given dimension mark in pink in the PG (Figure 41(a))

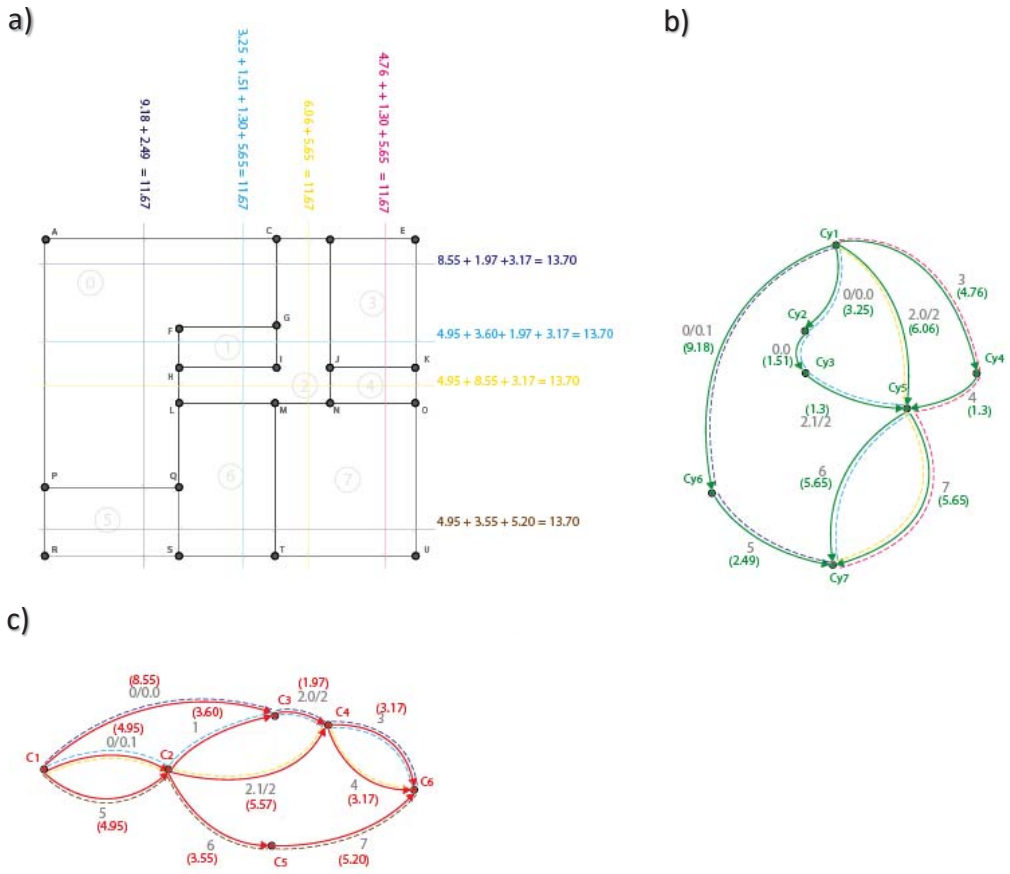


Figure 41 The corresponding dimension between PG and Dipath in DSG. (a) The Plan Graph with the sets of dimensions corresponding to Dipath in DSG. (b) DSG for y-dimension with dash lines in different colour define each Dipath and each of them corresponding to the sequential dimensions of the Plan Graph. (c) The D.S.G. for x-dimension with dash lines in different colour define each Dipath.

We will use one of the four corner vertices, which belong to the PG representing the unit boundary. In Figure 42 (a) the vertices (A), (E), (U), (R) are four vertices and only one of these four vertices will be used to define the sequential dimension constraint for the target unit plan. This is an important step because if there is a change of any room's dimension in the first part of the sequence then the following rooms position will adjust to this change.

Proportional scaling or "Scale-based of source unit plan in target boundary" operations are the first step for this editing operation; the corresponding scale factor (for the x-dimension), which has been computed, will multiply with all corresponding edges in DSG (for the x-dimension) of the source unit plan and repeat the same procedure to the y-dimensions. This

will create the new dimension for the target unit plan. The coordinate plane is an element that defined the constraint of a room or more rooms which will maintain its/their dimensions.

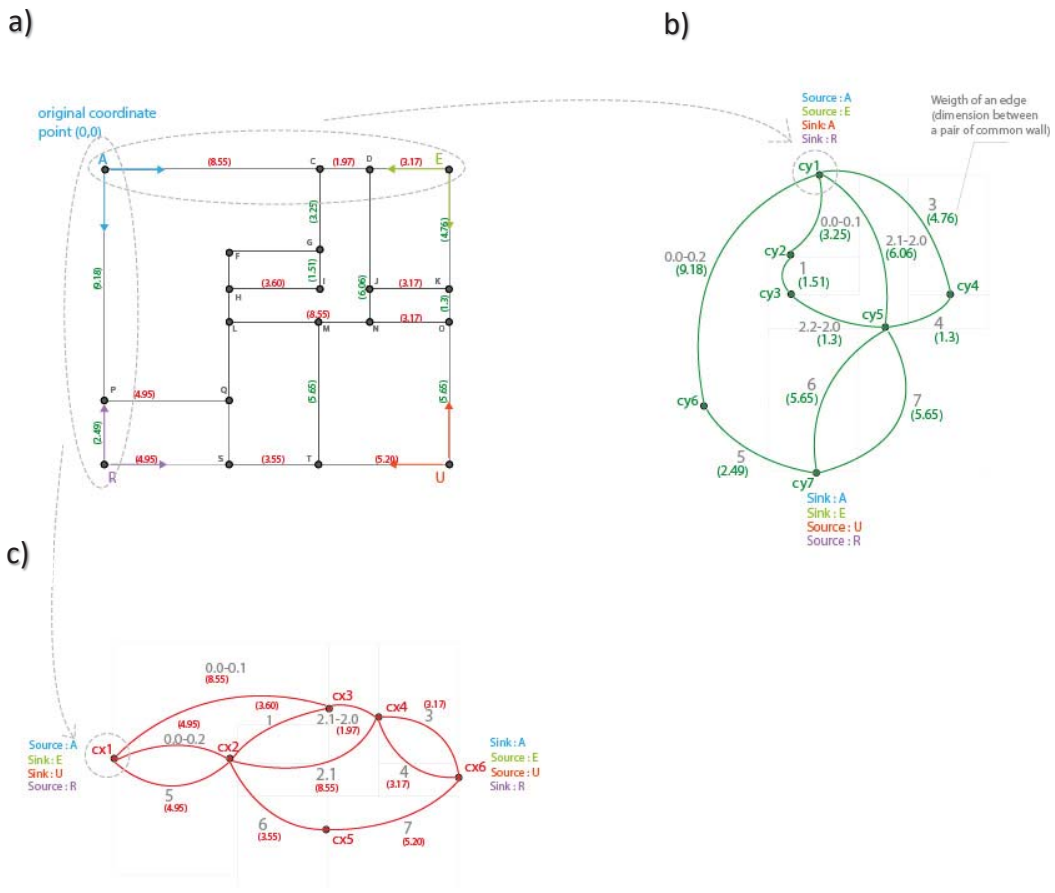


Figure 42 A coordinate plane in a PG defining a source vertex and a sink vertex in DSG, a PG with the sets of vertices representing the common walls in DSG. (a) A PG with vertex (A) as coordinate plane. (b) The DSG for y-dimension. (c) The DSG for x-dimension.

Figure 43 (a) to (b) are showing the different target unit plans which have been scaled according to this operation, each has a different vertex (the corner vertex) representing the coordinate plane. Each of them has a room which maintains its original size. A coordinate plane is one of the elements for the dimension constraint of a unit plan. The room which will maintain its original size is oriented to this coordinate plane.

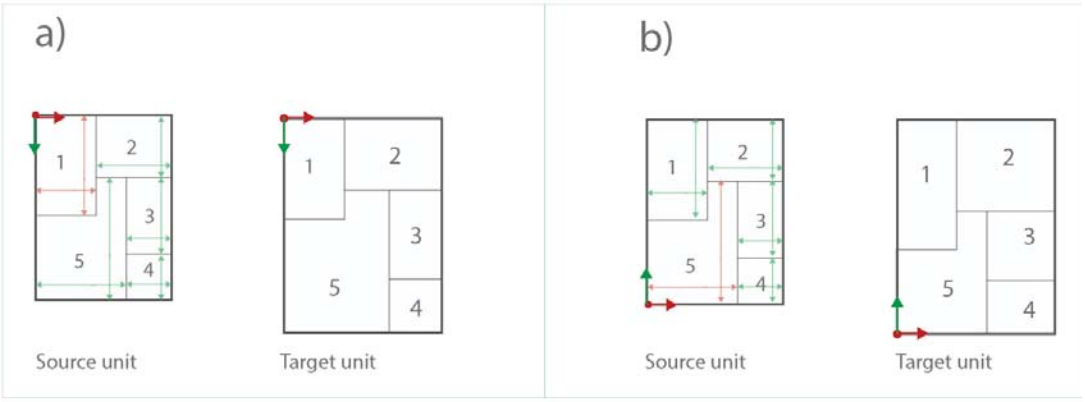


Figure 43 The two different versions of target unit plans from "Insertion of room topolog in a target boundary" for rectangular unit boundary. (a) The source unit plan left with coordinate plane or left corner define dimension constraint and right is a target unit plan as outcome. (b) The source unit plan left with coordinate plane define dimension constraint at left bottom and the target unit plan as outcome at the right.

The computation of rescale factors for this editing operation results in obtaining our preferred dimensions of the selected rooms which then will be the same as in the original (the source unit plan).

$$Fxr = \frac{Xp}{Xt} \qquad (3)$$

$$Fyr = \frac{Yp}{Yt} \qquad (4)$$

Table 1 formula-index of rescale factors for the room

Index	***Description***
Xp	$the\ X-dimension\ of\ prototype\ unit.$
Xt	$the\ X-dimension\ of\ target\ unit.$
Yp	$the\ Y-dimension\ of\ prototype\ unit.$
Yt	$the\ Y-dimension\ \ of\ target\ unit.$
Fxr	$the\ reverse\ scale\ factor\ for\ x\ dimention.$
Fyr	$the\ reverse\ scale\ factor\ for\ y\ dimention$

Proposal towards rescaling one of a middle room and related issues.

In this chapter we will show the steps and results of rescaling the non-convex orthogonal face in the middle of the unit plan after" Scale-based source unit plan in target boundary" (proportional scaling of the whole floor plan). In Figure 44(a) the face (2) will be rescaled to maintain its original size (as in the source unit plan), the face (2) are marked in blue and they are represented by the edges marked in blue in the DSG in Figure 44(b) and (c) these edges will be multiplying by the corresponding scale factor (reverts scale factor from formula (3) and (4))

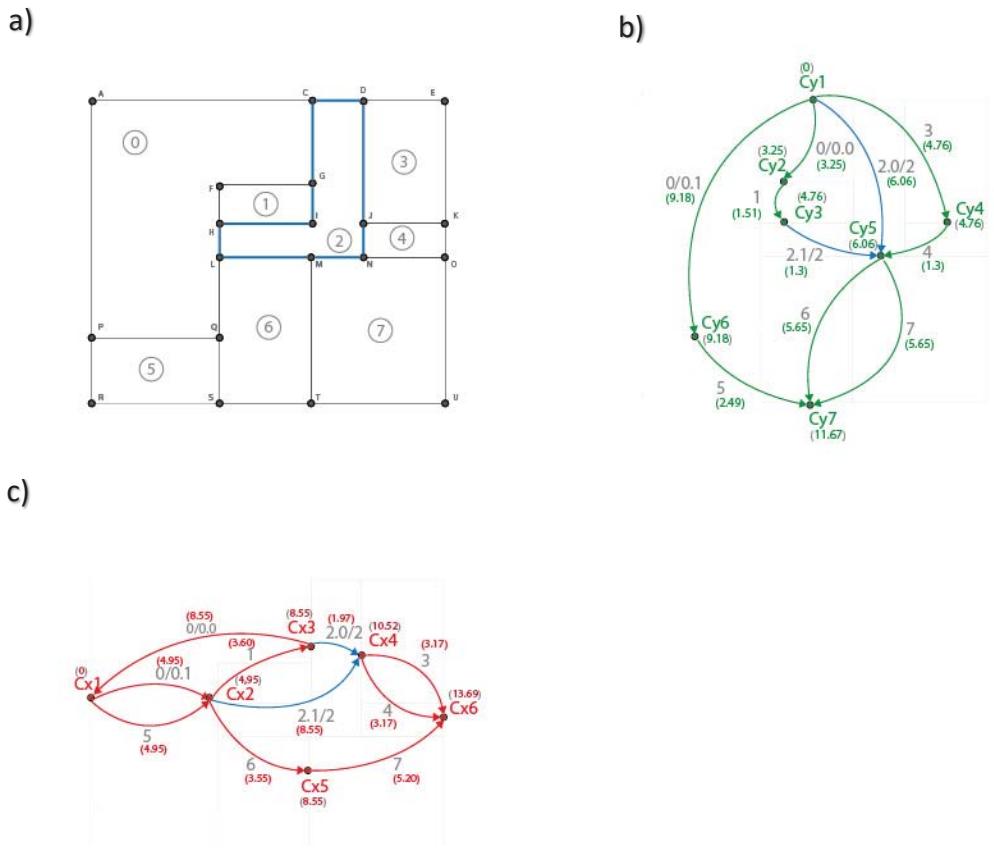


Figure 44 Blue marking of a face (2) in the middle which then will be rescaled. (a) PG with face (2) marked in blue as it will be rescaled to match the original size. (b) The DSG for y-dimension with the edges marked in blue represent the dimension of a face (2). (c) The DSG for x-dimension with the edges marked in blue represents the dimension of a face (2).

The result is shown in Figure 45, the dimension of face (2) now has been changed. The position of vertex (c5) (see Figure 45 (b)) for the y-dimension and the vertex (c4) for the x-dimension (Figure 45 (c)) have also changed. These vertices are the parent vertices of dependant vertices. If the parent vertices change its position, then their dependant vertices (following vertices) will change also. In Figure 45(b), the edge weight of the parallel edges (6) and (7) have been changed from 5.80 to 5.96. Only the immediate vertices will change their position

and not the source and sink vertices (represent exterior wall), because we only modify the room dimensions and not the exterior wall boundary and its position.

See Figure 45(b), the graph represents the y-dimension of floor plan. The position of vertex (c5) which represents the common wall has been changed. The result is not only that the dimension of the face (2) has changed, but also the other rooms' dimensions have also changed. We will use a DSGs as dimension constraint in order to detect which of the rooms will get the impact from the change. In Figure 45(b) the edge (2.0) has same edge weight equal to the sum of all edge weight in Sub-Dipath (c1 c2 c3 c5) and Sub-Dipath (c1 c4 c5), because they share the start vertex (c1) and end vertex (c5) representing common walls. We can show these relationships as following the formula.

Edge (2.0) = Edge (0.0) + Edge (1) = Edge (3) + Edge (4) or

Edge (2.0) = Dipath (c1 c2 c3 c5) = Dipath (c1 c4 c5)

The edge weights in two mentioned Sub-Dipaths must be adjusted to this rescaling of face (2). There are several edges form each Sub-Dipath, because of this designer might have to control the constraint by lock or unlock the edge which will change or not change their dimension after rescaling of the room.

We could say each of these Sub-Dipaths will have only one Degree of Freedom, which will be defined by designer. The Degree of Freedom (DOF) is the number of the independent parameters which are defined by its configuration in the system (Wikipedia, n.d.)

For example, in Figure 45(b) it shows the updated version of the Sub-Dipaths marked in a rectangle after the changing of dimensions of face (2). The edges (0.0) and the edge (3) are independent parameters.

In Figure 45(c) it shows the DSG and its composition. Each of the composition has at least one hinge connected to one another. They are interrelated. Changing one composition will affect the others and their members inside the composition.

The DOF would be set first at the level of the compositions and then to its edge members which are contained in it. Figure 45(c) shows the composition (4) which is the only one which is independent. The next level is the level which is inside the compositions. For example, in composition (2), the edge (1) is the only parameter which is independent.

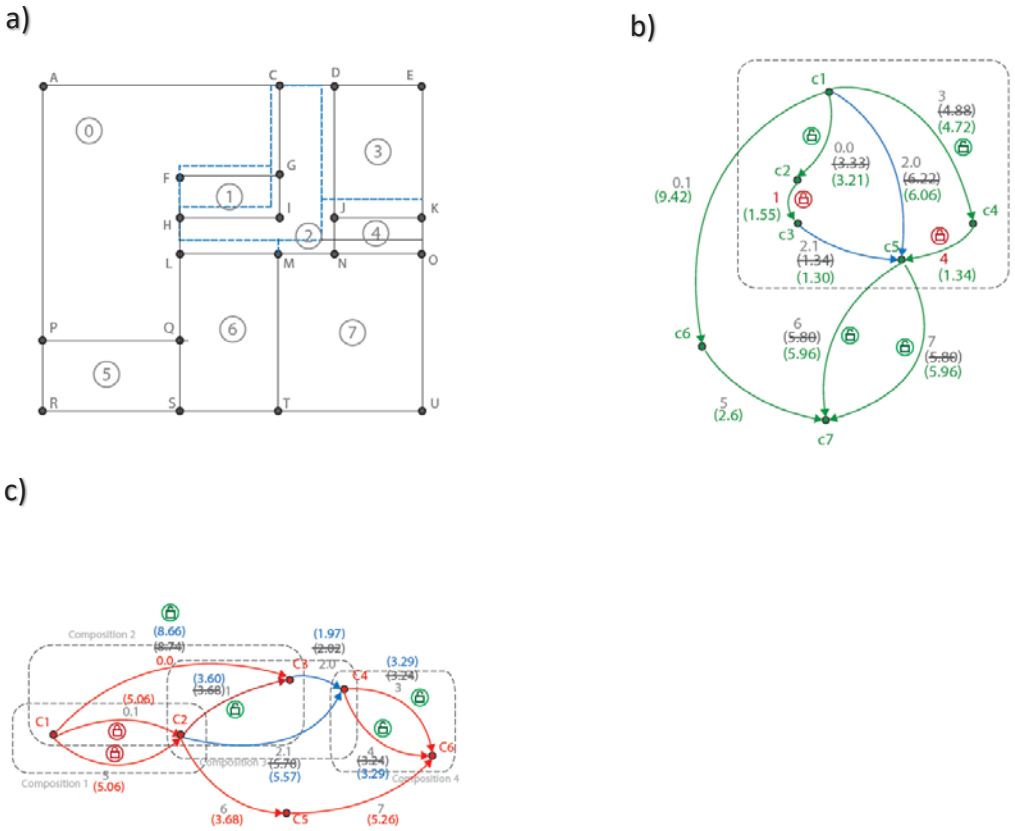


Figure 45 The update change of rescaling of face (2) and the change which also occurred to another face. (a) The PG with dash blue lines show the change of rescaling of face (2). (b) The DSG for y-dimension with the update of dimension and the dimension constraint apply to the edges. (c) The D.SG for x-dimension with the update of dimensions and the dimension constraint apply to the edges, together with decomposition of the graph.

The designer will have a freedom to control the constraint system representing the unit plan based on the DOF and the level of decomposition of the graphs.

In Figure 46 there are two different results of the floor plan according to the designer's preferences. The face (2) which maintains its original size is a non-convex orthogonal face. There are two common walls of this room (represent by vertex (C2) and vertex (C3) in Figure 45(c)) which define the original coordinate point of the plane for the scaling of face (2).

The designer would have to consider all these different types of changes based on graph constraints which may lead to different results. This could be over constraint and difficulty for a designer to monitor and keep trace on updating the dimensions after rescaling.

Figure 46 shows the results of the rescaling a middle face where only this face maintains its original size. As a result, most of the rooms do not maintain their consistency. It might be difficult for the designer to detect all changes and maintaining control. For better control of floor plan configuration, "Moving a wall" editing operation is introduced to address this issue. This editing operation will control the changing/rescaling of the dimensions of the face in the

middle by a manual change of a common wall to better monitor the application of this change to other faces.

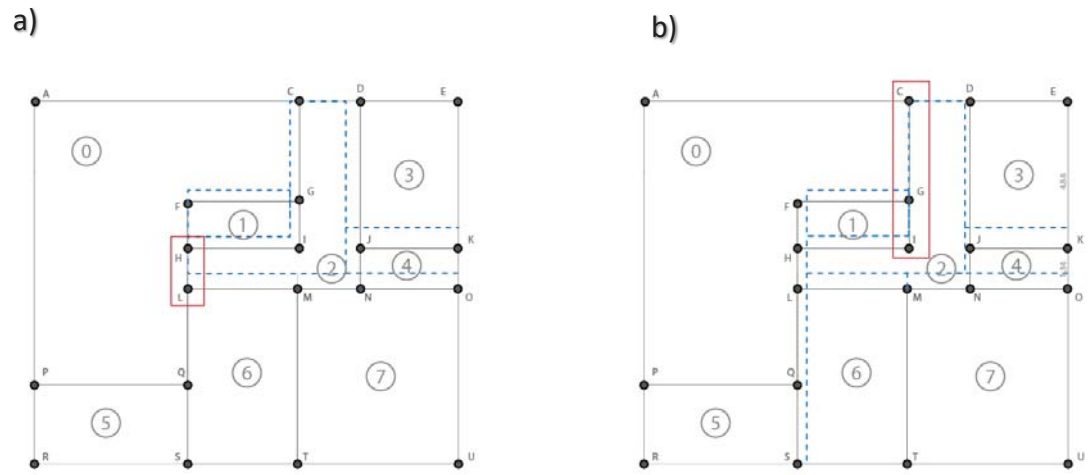


Figure 46 Two different types of changes relating two rescaling of face (2). (a) The PG and its updated change (result 1) done by rescaling of face (2) marked in blue dash lines. (b) The PG and its updated change (result 2) done by rescaling of face (2) marked in blue dash lines.

6.2.2 Quadrilateral unit boundary

The target unit plan with quadrilateral boundary as expected output. This category has the same procedure as the previous chapter, except the DSG for the x-dimension which has more vertices representing the sink. The non-orthogonal common wall of the source unit boundary is represented by the path (E, K, O, U) (see Figure 47(a)). This has not only one vertex represented in DSG but 3 vertices (see Figure 47(c)), because the dimension of each face in the last section have been changed by this boundary (the path (E, K, O, U)). This external common wall is represented by vertices in the DSG is different from the one for “Rectangular unit boundary”. See also the “Non-convex orthogonal unit boundary”, it has the same graph topology relating to the source and sink of DSG.

The dimension constraint for the room which share a common wall to the non-orthogonal boundary wall will not maintain their original size.

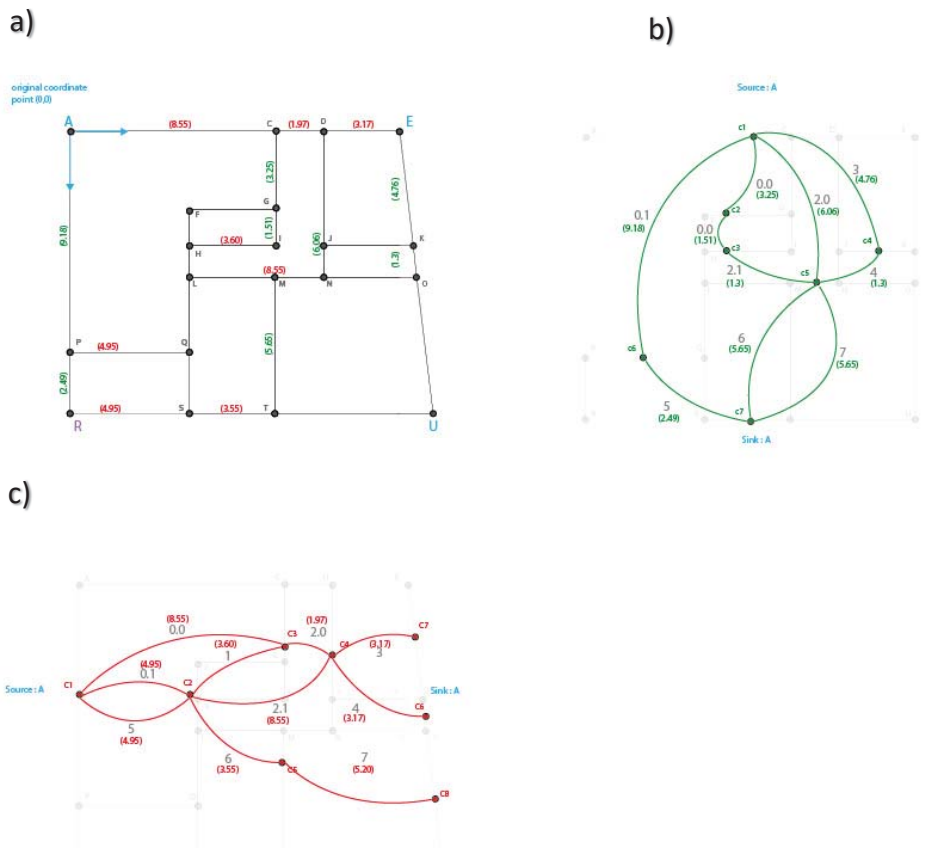


Figure 47 Graphs representation of unit plan with quadrilateral unit boundary. (a) The PG representing the target unit plan with Quadrilateral unit boundary. (b) The DSG for y-dimension. (c) the DSG for x-dimension.

6.2.3 Non-convex orthogonal unit boundary

The target unit plan with non-convex orthogonal unit boundary as expected output. In this editing operation, the same principle of graph representation of a rectangular unit boundary editing applies to this non-convex orthogonal target unit boundary and the only difference is that the number and position of source and sink of the unit boundary in DSG It is as shown in Figure 48. Each of the wall sections which oriented to the same external orientation do not share the same common wall. This would mean there are more than one source and sink vertices representing exterior boundary walls.

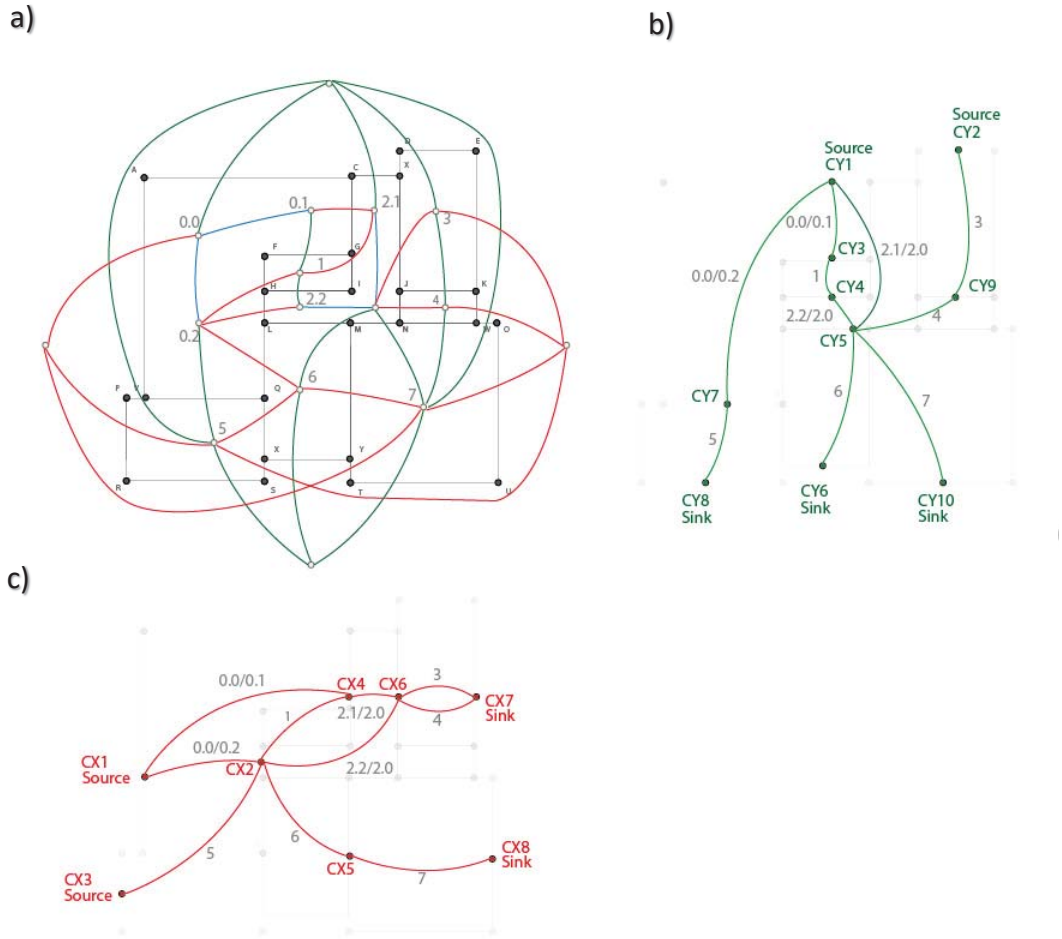


Figure 48 Graphs representation of unit plan with non-convex orthogonal unit boundary. (a) The PG representing the target unit plan with non-convex orthogonal boundary. (b) The DSG for y-dimension. (c) the DSG for x-dimension.

This editing operation is more complex than the editing operation for the “rectangular unit boundary” The geometry of the target unit plan boundary needs to be met according to certain requirements. In Figure 49(b), the left is the source unit plan and the right one is a target unit plan. We insert the source unit plan into a given target unit boundary. This might not be possible for this target unit boundary. Figure 49(c), is showing the DSGs in the x-direction for both source and target unit plans. The graphs are used to identify the constraint of the unit plan relating to the adjacency between faces and topology of the unit plan.

As it shows in Figure 49(b) the topology of the target unit plan in the right side is different from the source unit plan, the adjacency of the face (2) is no longer adjacent to the room (5) through the wall in y-direction, but through the wall in x-direction. It might not be the desired result for the designer. It might not be possible to insert the source unit plan on this target unit boundary by using the graphs representation.

To have a valid target unit boundary, the exterior wall segments topology (positioning) needs to be the same as in the source unit plan, as show in Figure 49(a). The wall segments marked in blue at the bottom (in the target unit plan on the right) need to be aligned to the same axis (blue dash lines) to the wall segments on the top, so that the topology of the target unit boundary will be the same as in the source unit plan boundary.

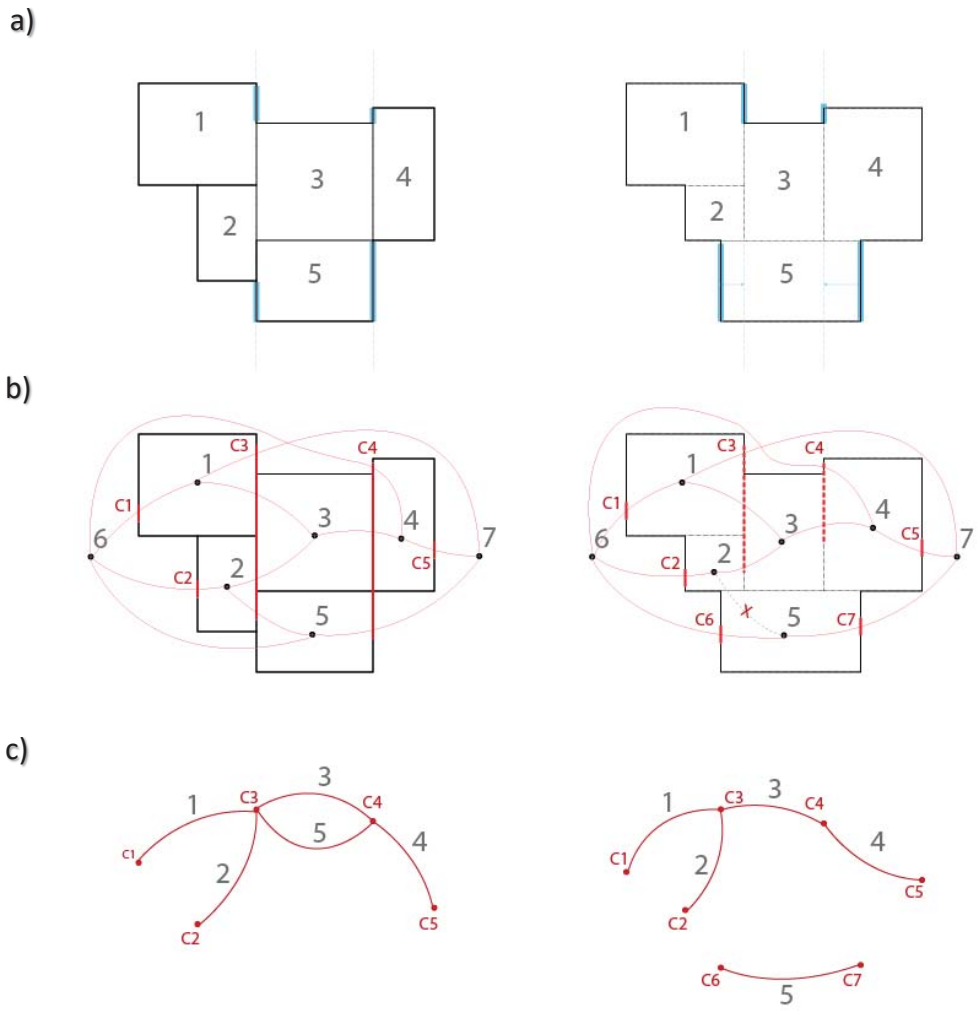


Figure 49 The difference of unit boundary topology of source unit plan (left) and target unit plan (right) will give the different AG and DSG. (a) The topology of the source unit boundary (left) and the target unit boundary (right). (b) PG with the AG represent both source unit plan (left) and target unit plan (right). (c) The DSG represent both source unit plan (left) and target unit plan (right).

After the target unit boundary is set according to the requirements, the mapping or the editing operation can proceed. The procedure of this operation (mapping a non-convex orthogonal boundary) is same as the "rectangular unit boundary" editing operation.

The DSG of the source unit plan (see Figure 50 (b), left side) will be used for the target unit plan with different assignments of the value to the edge. This assignment value we got from the input of the target unit boundary, as shown in Figure 50(a), is at the right side. The

distance between the blue cut line (C1) and (C3) will be the value to assign on the edge (1) in Figure 50(b), on the right side. The DSG for both unit plans have the same structure but have different values assigned to the edges.

There might be some case where the dimension of the intermediate faces of the target unit plan cannot be received from the target unit boundary. In that case the dimension will be generated proportional to the source unit plan as show in Figure 51.

The assigning of the dimension constraint to rooms by the designer may not be possible, because the room dimensions have been defined by their unit boundary.

If the assigning dimension constraint to rooms by the designer can be done without any change of their unit boundary then some of the rooms, which are attached to the boundary, may change their shapes, for example a convex shape turns into a non-convex shape. This is a proposal for further study.

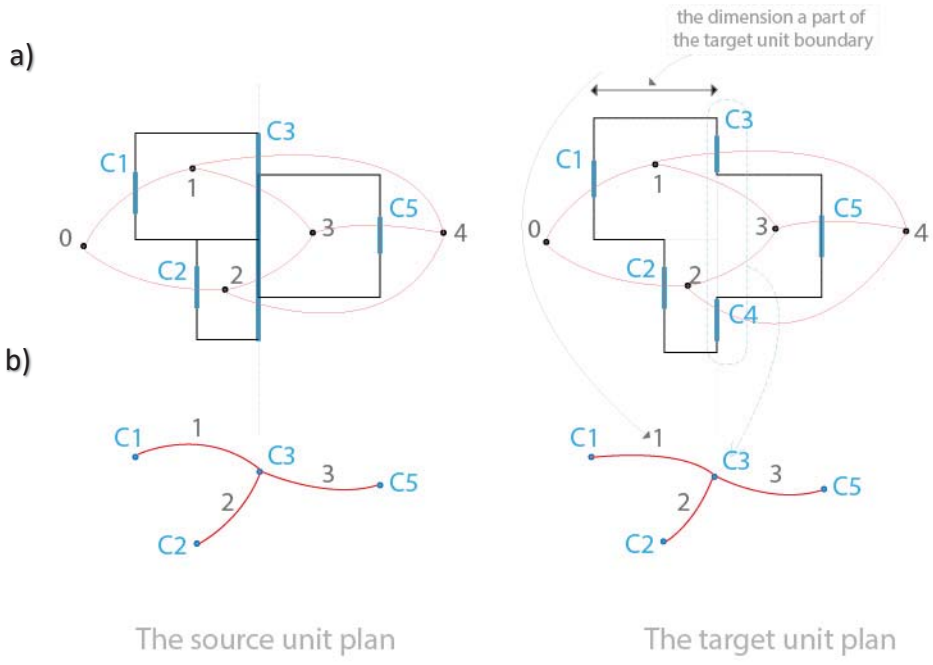


Figure 50 Assigning dimension received from the target unit boundary on the DSG (a) The PG with AG represent both source unit (left) and target unit (right). (b) the DSGs represent both source unit (left) and target unit (right).

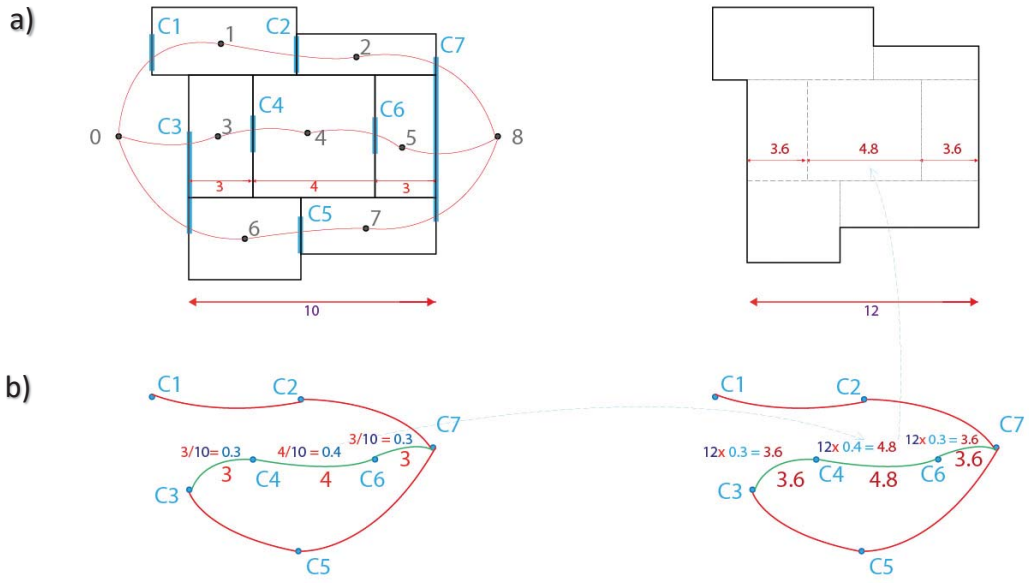


Figure 51 The computing of dimensions for intermidate faces for the target unit by using the ratio of the source unit plan to target the unit plan. (a) The PG with AG represent both source unit (left) and target unit (right). (b) the DSGs represent both source unit (left) and target unit (right).

6.2.4 Non-convex and non-orthogonal unit boundary

This editing operation is the combined version of “non-convex orthogonal unit boundary” editing operation and “Quadrilateral unit boundary” editing operation.

The defining dimensions constraint done by the designer is considered difficult for this type of unit boundary, as mentioned in previous chapter. This could be the proposal for the further study. In this category the dimension constraint for all the rooms, which attached to the non-orthogonal common wall of the unit boundary, will be set free. It would mean all the dimensions in the target unit plan will be changed according to its unit boundary.

The dimension constraint for the room which share a common wall to the non-orthogonal boundary wall will not maintain their original size.

6.3 Moving a wall

This chapter is addressing the assigning of a dimensions constraint in the intermediate room (as mentioned in the last section of chapter 6.2.1 “Rectangular unit boundary”) As a result there are many changes in the dimensions of others rooms which are adjacent to this intermediate room. This is considered difficult to monitor.

This editing operation is also optional for manual changing of the room-dimension/s. The preferable dimension, or dimension constraint, will be assigned by the designer to the edge of the DSG corresponding to the room in the PG. The dimension constraint will also be assigned to the subsequent room/s which share the common wall to this room which has been configurated. One type of the dimension constraint is that the following room will not preserve its original dimension. The second type of the dimension constraint is that the following room preserves its original dimension. This assigning of the dimension constraint is only possible if the following room is not adjacent to the external side. For example, if we move the wall (NO) upward (see Figure 54(a)) we can assign the constraint to the following room (4) to maintain its original size or not. At the contrary, if we would move the wall (JK) upward we cannot assign the dimension constraint to the following room (3) to maintain its original size, because this room is adjacent to the exterior space. Note the wall (NO) is a part of the common wall (LMNO) (see chapter “Sets of vertices representing the walls junction”), so to move the wall (NO) it is equivalent of moving the common wall (LMNO).

The underlying idea of this editing operation is to respond to a limitation in “Rectangular unit boundary

The limitation is the dimension constraint. It is difficult to control the change or to keep track of the changing of the rooms, which are not conform with the user preferences. Figure 52 is showing the applied changes (dash lines); there are many rooms affected by the change done by rescaling of the room in the middle (room (2)).

Dimension constraints of rooms have the purpose to solve this issue. This will help the user to better control the change of any preference room in any position of the unit plan and to keep tracking or monitoring the changes.

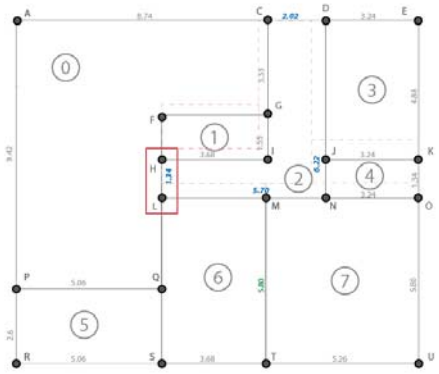


Figure 52 Applied changes (dash lines) of the floor plan done by rescaling room (2).

6.3.1 Weighted Graphs and Dimension subgraph

This is the operation of moving walls corresponding to the reassigning of the weight of the edge in a D.SG which is representing the room dimensions. In this chapter we will introduce the concept of Weighted Graphs and their relationship to the Dipaths representing all dimensions of the unit plan as background knowledge relating to the moving wall operation.

The numerical value or weight assigned to each edge of the graph is called "Weighted Graphs". In this paper the graphs representing the floor plan and the weight assigned to each edge will represent the distance between two wall junctions where the wall junctions are represented by vertices.

In Figure 41(c), the weight assigned to the edge (0.0), which connects vertex (C1) and (C3) is 8.55, and this represents the distance between two walls (in y direction) of the room (0). The weight assigned to the edge (2.0) is 1.97 which represents a dimension of the room (2). Finally, the weight assigned to the edge (3.0) is 3.17 and represents the dimension of room (3). The sum of all mentioned room's dimensions will be 8.55 + 1.97 + 3.17 = 13.70 (see Figure 41(a)). This sum value represents the distance between two external boundary walls (in y-direction). Respectively to the graph theory, the source vertex (C1) and sink vertices (C6) (Figure 41 (c)) represents the external walls in the y-direction. The sum of all the edge weights in each Dipath, which starts from the source vertex (C1) and ends at the sink vertex (C6), represents a distance between two external walls. In Figure 41 it is showing all Dipath marked in coloured dash lines and in table 2.1 it shows the sum of all the edge-weights for each Dipath.

All the Dipaths have the same sum value. In the table. 2 and Figure 41 is showing all the Dipaths representing the sum of all corresponding room dimensions and the set of blue dimensions in the x-direction of the plan graph (Figure 41 (a)) which are corresponding to the Dipath which are marked in blue dashed lines in the DSG for the x-direction (Figure 41 (c)).

In Table 1 it shows all Dipaths (in DSG for x direction) with the assigned dimensions. The floor plan based on graphs representation will be the constructed based on the coordinate plane. The coordinate x can be read from each vertex. This is depending on the coordinate plane and the original coordinate point, which has been set by the designer and this will define the rooms' dimensions, in case of changing of the first rooms 'dimension it will affect sequentially the following rooms. In Figure 43 it is shown the different results for each different coordinate plane which is defined by the corner of the unit plan boundary. In the Table 1 it is showing the relationship between the coordinate of each vertex and the weight edge representing the room dimension. For example, in Dipath (1) the vertex (C3) has the coordinate x equal to x-coordinate of vertex (C1) (the previous vertex in the sequence) plus the weight of the edge, which connects between them, which shall be represented by the equation:

$$0 + 8.55 = 8.55$$

Previous vertex + Following vertex = X-Coordinate of Following vertex

All the source and sink vertex in each Dipath will have the same x-coordinate. There is an interrelationship of the values of the edges in different Dipaths. For example, the weight of the edge (0.0) in Dipath (1) is equal to the weight of the edge (0.1) plus the weight of the edge (1) in Dipath (2). Following shows some of the equations which represents the interrelation between the Dipaths.

Edge name: 0.0 = 0.1 + 1

In Dipath name: Dipath 1 = Dipath 2

Edge name: 2.1 = 1 + 2

In Dipath name: Dipath 3 = Dipath 2

These relationships will be the constraint for the designer's preferences relating to the room scaling method or modification of the floor plan.

Dipath 1	Vertex	C1		C3		C4	C6
	x-coordinate	0		8.55		10.52	13.69
	Weight edge		+ 8.55		+ 1.97		+ 3.17
	Edge name		0.0		2		3
Dipath 2	Vertex	C1	C2	C3		C4	C6
	x-coordinate	0	4.95	8.55		10.52	13.69
	Weight edge		+ 4.95	+ 3.6	+ 1.97		+ 3.17
	Edge name		0.1	1	2		3
Dipath 3	Vertex	C1	C2			C4	C6
	x-coordinate	0	4.95			10.52	13.69
	Weight edge		+ 4.95		+ 5.57		+ 3.17
	Edge name		0.1		2.1		4
Dipath 4	Vertex	C1	C2		C5		C6
	x-coordinate	0	4.95		8.5		13.7
	Weight edge		+ 4.95	+ 3.55		+ 5.2	
	Edge name		5	6		7	

Table 1 Set of all Dipath represent the sum of all the corresponding rooms' dimension.

6.3.2 Maintaining the adjacencies between faces

Figure 54(a) shows the plan as an input for the operation of moving a wall. The study of Roth, as mentioned earlier, relating to the as DSGs shows that there is no concrete solution for assigning the dimension in a way that the rooms will maintain their adjacencies requirements. This issue concerns the study of the floor plan layout specially in the moving a wall option. If we move any wall, the AG will maintain its original topology.

For example, if we would move the wall (MT) from its original position (see Figure 54(a)) to the right side, as illustrated in Figure 54(b), the adjacency of room (6) in y-direction will be changed. In Figure 54(a) it is showing the adjacency of room (6) in y-direction before the moving of the wall (MT) and the room (6) adjacent to the room (2) in y-direction only. After moving the (MT) wall, the original adjacency topology of this Plan Graph is changed, as shown in Figure 54(b) and the room (6) is not only adjacent to the room (2) in y-direction but also adjacent to the room (4). The change in the AG will change also the DSG

The solution to any change of a position of any wall then will be controlled by the dimension subgraphs to maintain the adjacency between rooms and the dimension constraint will be

given to the DSG as will described in the following about the dimension constraint relating to the layout editing which need to be of concern to the designer.

6.3.3 Dimension constraint assigning to Dimension Sub-Graph for moving a wall operation

In Figure 53(c), the vertex (C5) corresponding to Edge (MT) in the Plan Graph (Figure 53 (a)) represents a wall junction. This wall could be moved to the left and the right by the designer, but only for a certain distance and this wall cannot change the adjacency between rooms on this floor plan.

It is necessary to create a set of constraint-based relationships between the D.S.G.s and the Plan Graph.

In the Plan Graph there are a set of polygons and each polygon has a set of control points (corner points), for example in Figure 53(a) the polygon 2 has vertex (C), (D), (N), (L), (H), (I) (marked in red), as control points, but in between these control points there is vertex (J), (M) and (G) (marked in green) which are on the edge of this polygon. These vertices will be used to control the distance constraint.

The vertex (M) lies on the horizontal edge of the polygon (2) which is between the vertex (L) and (N). This vertex (M) is the end vertex of an edge (M, T) which is perpendicular to the mentioned horizontal edge (L, N). We would set the constraint to that edge (M, T) and should not move over the range - 1 meter of the edge (L, N) to the left and right. Number 1 as the reference number for the size of door.

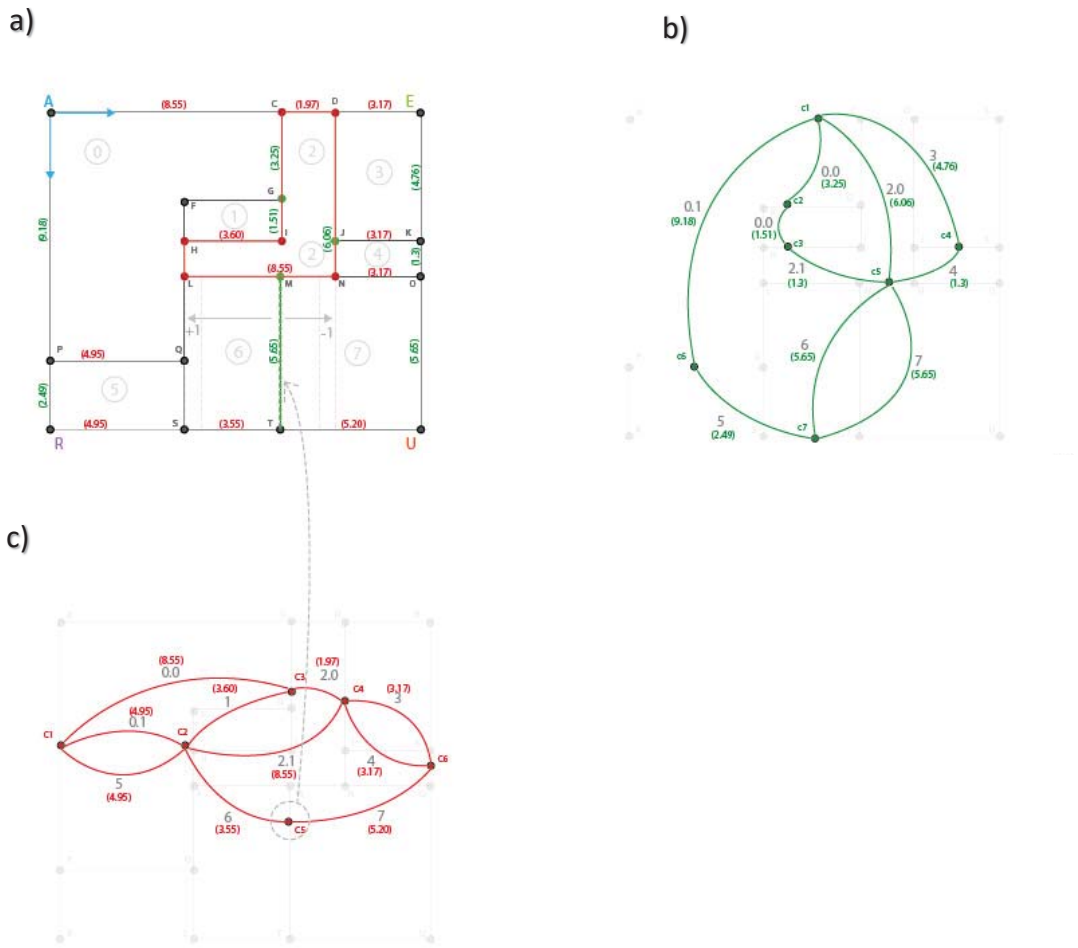


Figure 53 The manual change done by shifting the wall section. (a) the Plan Graphs. (b) the Dimension Subgraph for y-dimension. (c) the D.S.G. for x-dimension.

The designer preferences for the room rescaling (or changing dimension) specially for the intermediate room will be made under a certain dimension constraint. The designer will define which room to modify, then assign the new value to the edges which are representing the dimensions of selected rooms in DSG.

The result would be that some rooms would be affected by this change. The rooms which are affected by this change can be seen from the DSGs (see Figure 53(b) and (c)).

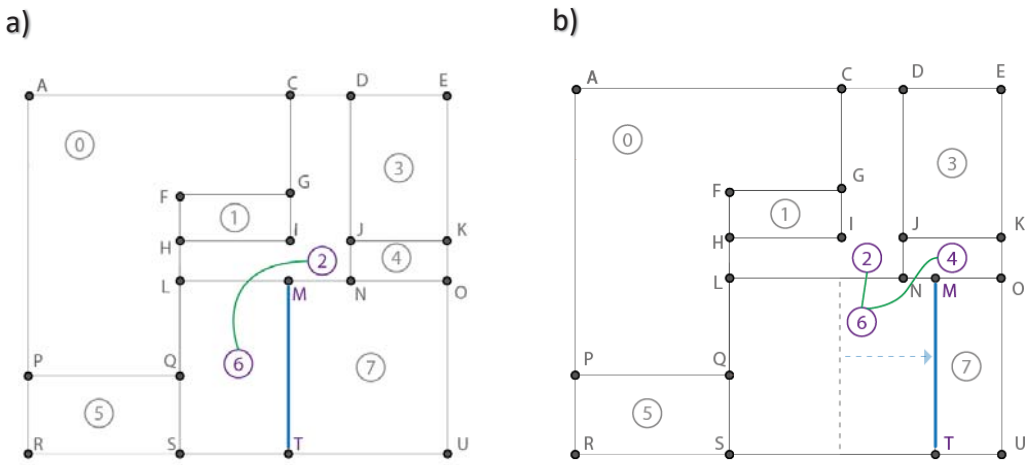


Figure 54 The changing of rooms adjacency by moving of the wall (MT). (a) The floor plan with room (6) adjacent to room (2). (b) The floor plan with room (6) adjacent not only to room (2) but also room (4) after the (MT) have been moved.

7 IMPLEMENTATION

In this study not all the steps for “Deriving a graph-based floor plan from a rectangular floor plan” (see Figure 55) and “Layout editing operations” (see Figure 56) are partially implemented. The outline marked in red are steps that have not been implemented, in blue are steps which have been implemented and violet are steps which are partially implemented.

The Rhino-Grasshopper and plugin application such as space syntax (Jeong & Ban, 2011), spiderweb, kangaroo have been used for the implementation.

The step of “Deriving a graph-based floor plan from a rectangular floor plan” is a part of constructing the source unit from existing templates which do not have valid dimensions and it has not been implemented.

In this paper the unit plan from Housing Complex by Hermann and Johannes Kaufmann architects is used for implementation and it has valid dimensions. Because some of these steps are the same as the steps in “Layout editing operation” we will consider that these steps are also implemented and they are marked in blue.

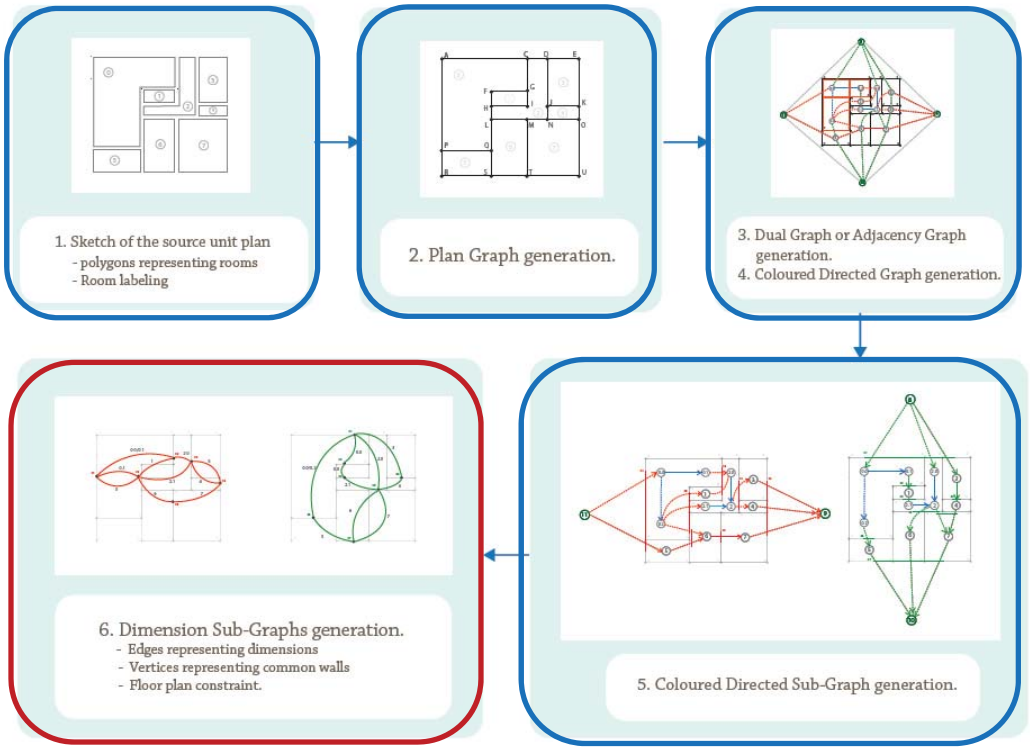


Figure 55 Overview of the steps to derive a graph-based floor plan from a rectangular floor plan, which have been implemented. The outline marked in red are steps that have not been implemented and blue are steps which have been implemented.

Following steps have been implemented:

- The sketching polygons, “Deriving a graph-based floor plan from a rectangular floor plan” which is representing rooms and which is a part of “Layout editing operations” (see Figure 56).
- The step “Computing the scale factors”.
- The editing of “rectangular unit boundary” one of four variations of “Insertion of rooms topology in a target boundary”.
- Implementation of a further step of the above, the “Plan Graph generation”.
- The step “Dual Graph generation”.
- “Scale-based of source unit plan in target boundary” and the step “Generation of sets of corresponding vertices from the plan graph”.
- “Insertion of rooms topology in a target boundary” and the step “Moving a wall”
- The moving of a wall of the source unit plan and applying the change to all target units.
- The step “Locating the target unit plan in the building floor plan” has been implemented.

Not implemented are:

- The assigning of constraints to the moving of a wall and their adjacency room will maintain its dimension has not been implemented due to the time limitation.
- It is not feasible to implement the step “Directed Sub-Graphs generation” and the step “Dimension Sub-Graphs generation” due to the limitation of graph application capacity in grasshopper. There is a possible way to implement this with a greater number of components and a further module is needed in grasshopper, this might not be suitable for any designer budget.

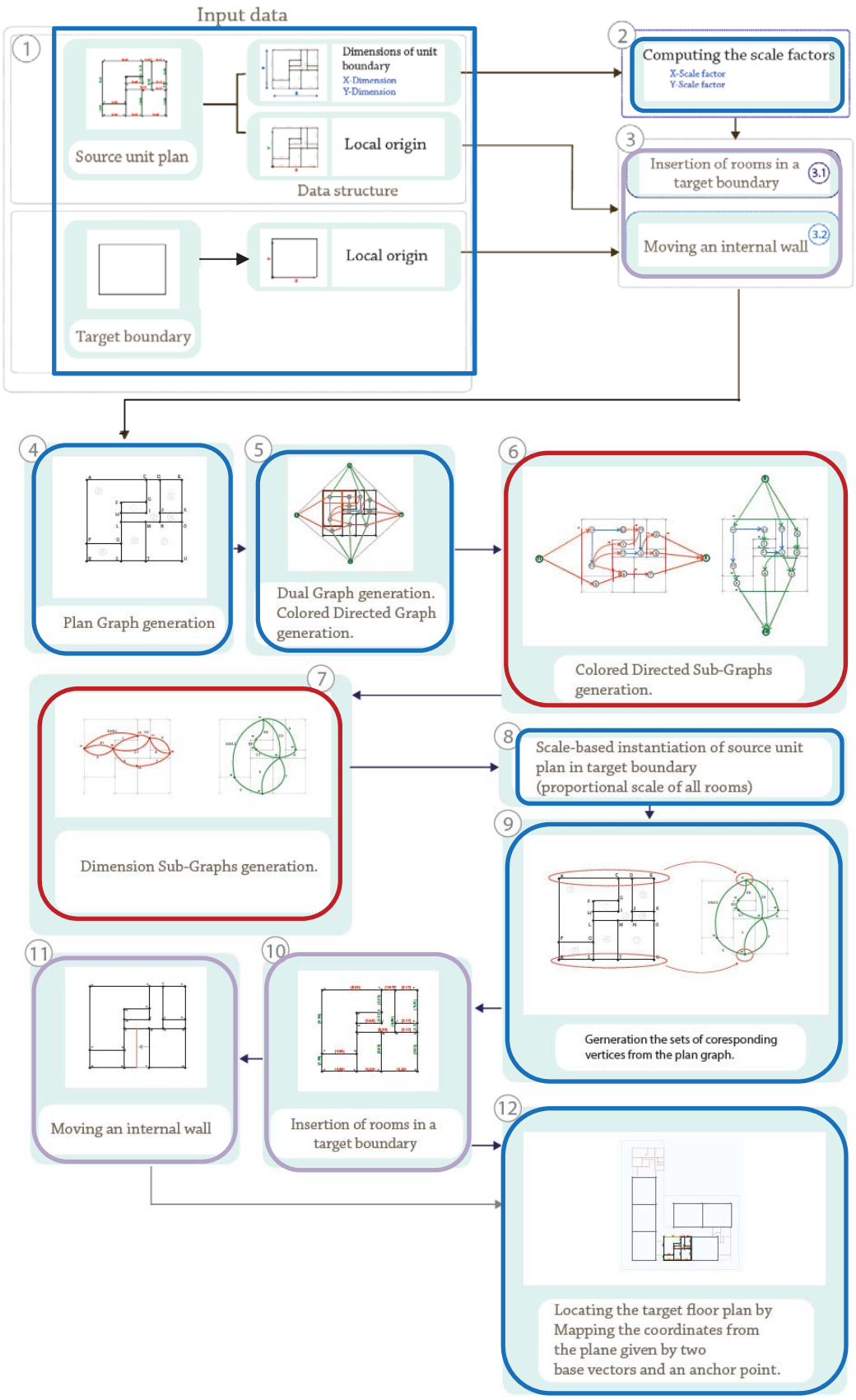


Figure 56 Flowchart shows steps required for the editing operations.

8 VALIDATION

8.1 Housing project selection

Example: case study “Housing complex” first floor plan by Johannes Kaufmann Architect, located in Vienna. This Project has been chosen as example because it has a similar unit plan with the same unit plan topology but has slightly differences in terms of unit area or unit boundary and one of the unit plans, unit plan (4) as shown in Figure 58(a), has some room dimensions which are different from the other unit plans, therefore this unit plan can be used as example for the “Moving a wall” editing operation.

Figure 57 The Housing Complex by Hermann and Johannes Kaufmann architects.

a) b)

Figure 58 The first-floor plan of the building project “Housing complex” by Hermann and Johannes Kaufmann architects. (a) First Floor plan. (b) The unit layout of Housing complex.

The project Housing complex by Hermann and Johannes Kaufmann architects is used to demonstrate the implementation of the floor plan layout editing operations.

In Figure 58 it is showing the first-floor plan of “Housing complex” which consists of 9 units. We will use unit (1) to (7) which have the same floor plan topology but are different in dimension of unit boundary. The unit (2) marked in red will be used as the source unit plan and the unit (1) and (3) to (7) will be used as the target unit boundary.

In Grasshopper there is one set of steps for the source unit plan and there are six sets of steps for the target unit plans. The reason to have six copies represents each a target unit plan, because each target unit boundary has different attributes, this in terms of boundary dimensions, dimension constraints of rooms and their orientation in the building.

8.2 Insertion of rooms in a target unit boundary

The step of this editing operation as follow.

1. The user assigns polygons representing rooms and 4 sides representing orientations (see Figure 59(a) to (b)).

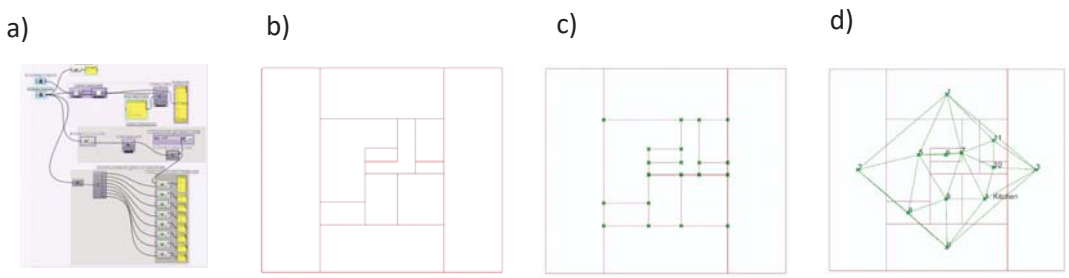


Figure 59 Insertion of rooms in the the source unit plan in Rhino-Grasshopper. (a) Grasshopper component. (b) The polygons representing rooms where 4 sides represent orientation as input. (c) The set of vertices for the plan graph generation in Rhino-grasshopper. (d) A Dual Graph representing room adjacency.

2. Computing a set of vertices for the plan graph, this will be later be used for layout editing operations (see Figure 59(c)).
3. Computing a Dual Graph representing room adjacency (see Figure 59(d)).
4. Computing the sets of vertices representing each room.
5. Inserting 4 points representing 4 corners of the target boundary in the floor plan (see Figure 61(0)). This will compute the x-dimension and y-dimension of the target unit boundary. The dimensions of this target unit boundary together with the dimensions of the source unit boundary will be used to compute the scale factors (see Figure 60 and Figure 61(1)).

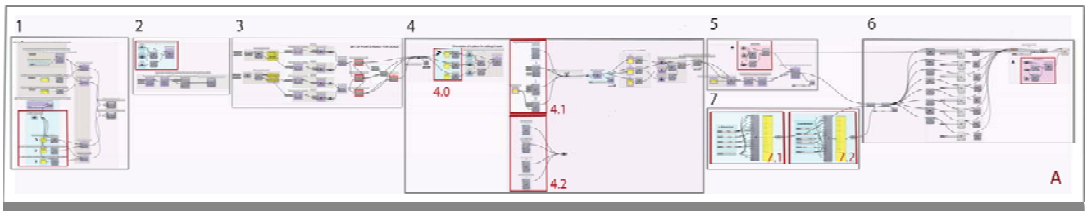


Figure 60 The layout editing operations in Rhino-grasshopper.

6. Copying vertices of the source unit plan in to a new local plane, which is defined by the designer (see Figure 61(2)).

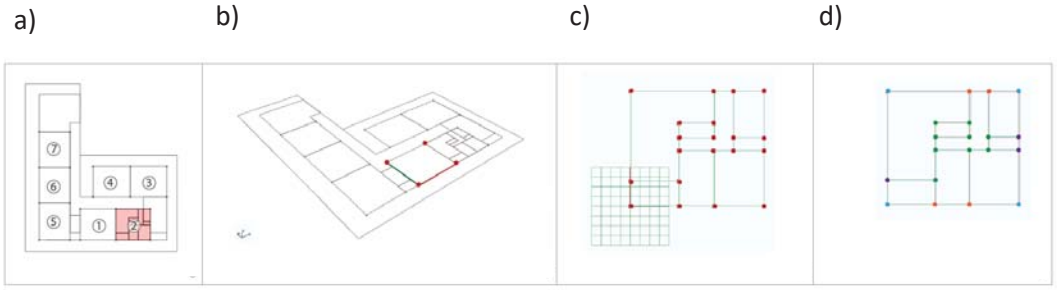


Figure 61(a) The floor plan layout with the source unit (2) plan marked in red and other target unit boundaries are marked with number (1) and (3) to (7). (b) 4 points representing 4 corners of the target boundary in the floor plan and the dimensions between them. (c) The copying of the vertices of the source unit plan. (d) Creating sets of vertices for scaling.

7. Generating sets of vertices by changing of the coordinates following an editing operation (Figure 61(3)).

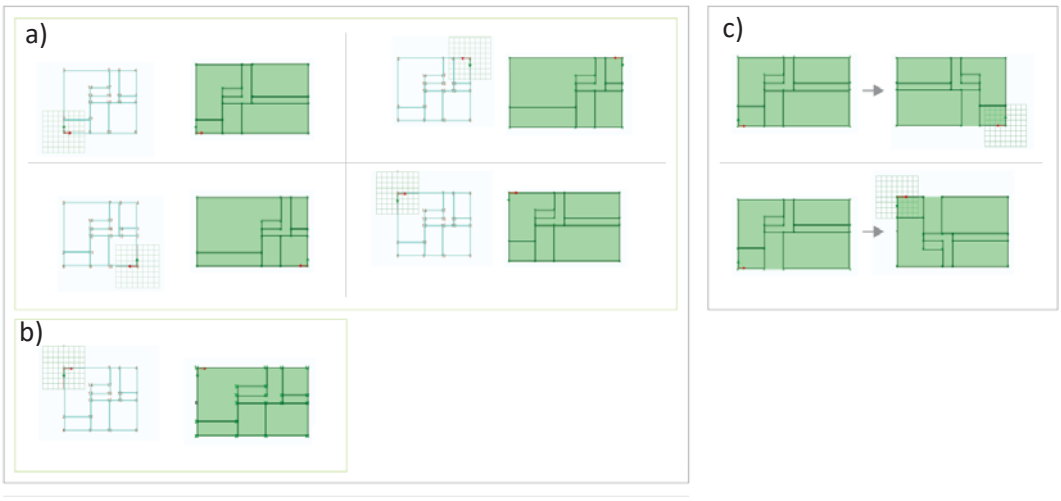


Figure 62(a) Different variations "Scale-based of source unit plan in target boundary" editing operation defined by a plane. (b) The target floor plan unit done by "Scale-based instantiation of source unit plan in target boundary". (c) The mirror operation defining the orientation of the target unit plan in the building floor plan.

8. Designer defined plane for either the “Scale-based of source unit plan in target boundary” Rectangular unit boundary or “Constraint-based of source unit plan in a target boundary” (see Figure 61(b)).

   For the “Scale-based of source unit plan in target boundary” the plane will define the orientation of rooms which will maintain their original dimensions, as shown in Figure 62(a).

9. Designer defined plane for mirror operation, this is to define the orientation of the target unit plan in the building floor plan (Figure 62(c)).

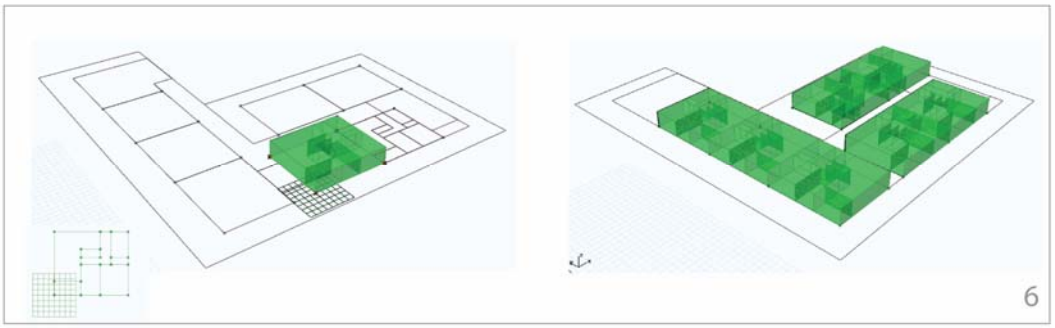


Figure 63 The rooms of the target unit are copied to the building floor plan by difining the plane.

10. Computing a 2D polygon and 3D polygon representing rooms. Then copy this outcome into the building floor plan by difining the plane made by the designer (see Figure 63)

8.3 Moving a wall

The Designer made configuration “Moving a walls” of individual target unit plan (see Figure 64 and (58)). In Figure 66 it is showing the configuration done on the source unit plan and the change occures to all other target unit plans.

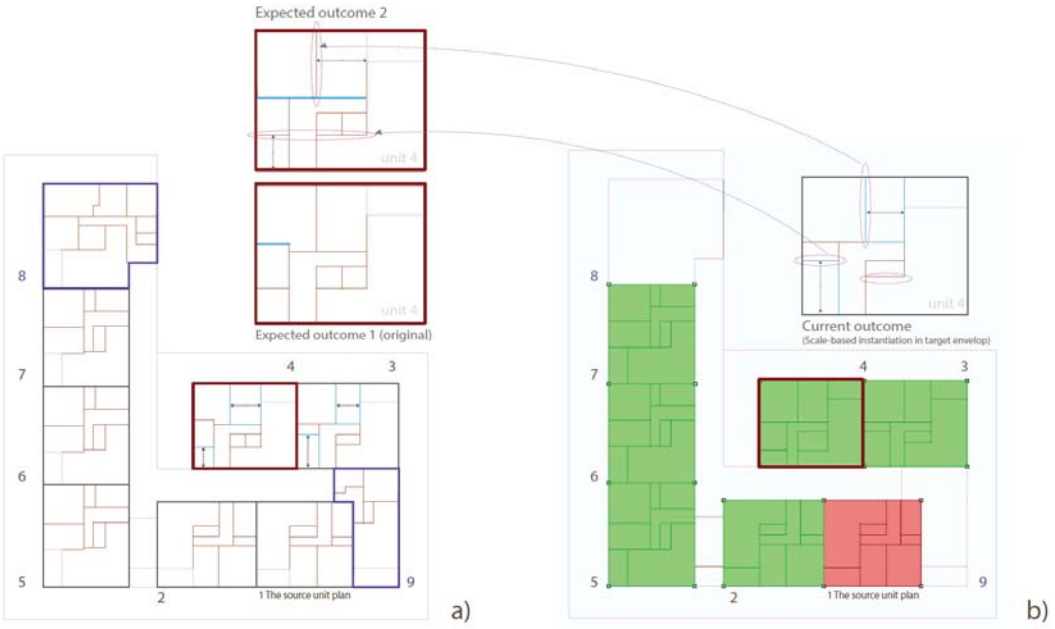


Figure 64(a) The expected outcome of “movings wall” editing operation. (b) current input for moving wall operation.

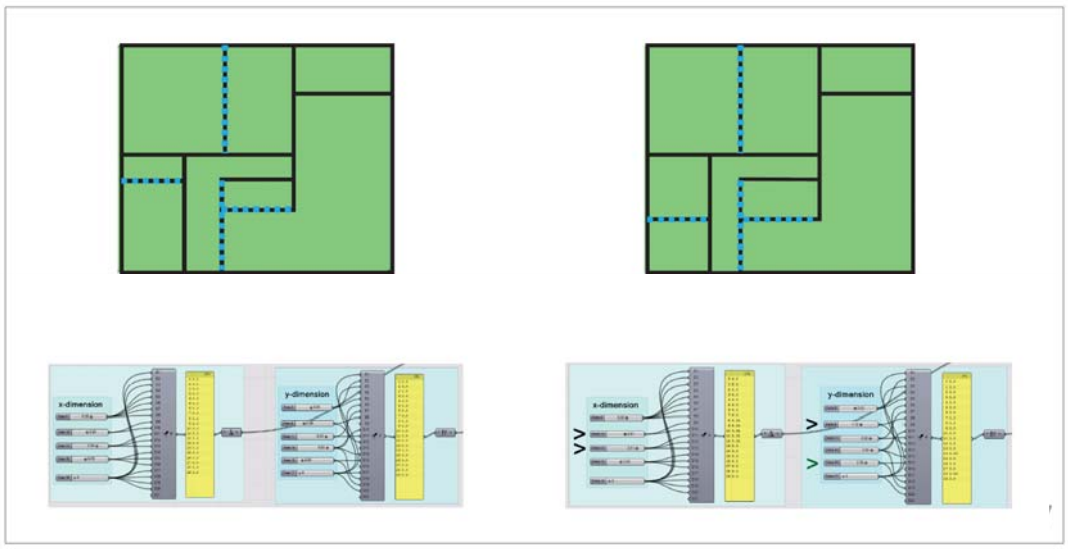


Figure 65 The expected outcome (left) and the current input (right) in Rhino-grasshopper.

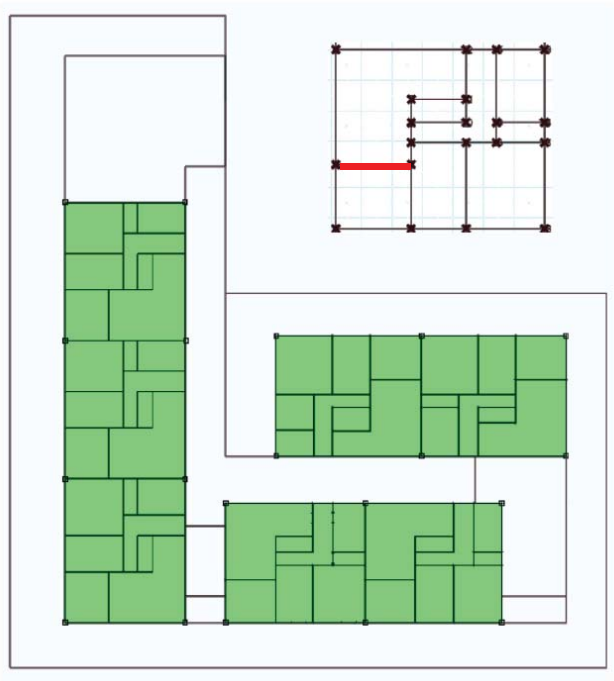


Figure 66 Moving a wall of a source unit plan and apply changes to all other target unit plans.

9 CONCLUSION

This study is about graph-based floor plan representation. The primary scientific literature which has been influencing this work is “Turning a graph into a rectangular floor plan” by Roth study (Roth et al., 1982). The principle of the Roth study has been adapted for the layout editing operation in our study. There are some issues and limitations of the Roth method, which is that the graph only represents convex faces but not non-convex orthogonal faces. Therefor in this study the graph base floor plan representation for non-convex orthogonal faces has been developed/extended from the Roth study. The limitation of this study, in the context of a master thesis, is the limited experience and the limited knowledge of computer science related “Layout editing operation” and “Deriving a graph-based floor plan from a rectangular floor plan”. This could only be partially implemented with the software Rhino and Grasshopper. The application “space syntax”, which generates the AG, generally works but this application has some limitation which is that it could not generate the DSG. The solution to this limitation is to generate this graph manually through additional components in Grasshopper but again this requires a fair amount of time and knowledge. To achieve the goal of constructing a floor plan in this study, sets of vertices have been created to represent the DSG. The editing operation “Constraint-base insertion of source unit plan in target boundary” for a rectangular unit boundary and “Moving a wall” implemented. Because of the limited of suitable software and applications for parametric modelling of floor plans based on graph representation, the exercise to apply all editing operations might be for a further study. The “Moving a wall” of a unit plan with fix non-convex orthogonal unit boundary has not been mentioned in this study, this is also expected for a further study.

The process of adapting the system for each editing operation in Grasshopper requires a lot of effort and knowledge on computer science, which is surely useful only for larger and complex architectural projects.

LITERATURE

Aldous, J. M. (2005). *Graphs and applications an introductory approach Joan M. Aldous and Robin J. Wilson* (5th ed.). London: Springer.

Eastman, Charles. “Automated space planning.” Artificial Intelligence (Elsevier) 4, no. 1 (1973): 41-64.

Jeong, S. K., & Ban, Y. U. (2011). Computational algorithms to evaluate design solutions using Space Syntax. *Computer-Aided Design*, *43*(6), 664–676.

Liggett, R. S. (2000). Automated facilities layout: past, present and future. *Automation in Construction*, *9*(2), 197–215.

Nourian, P. (2016). *Configraphics : Graph Theoretical Methods for Design and Analysis of Spatial Configurations*. Technische Universiteit Delft.

Roth, J., Hashimshony, R., & Wachman, A. (1982). Turning a graph into a rectangular floor plan. *Building and Environment*, *17*(3), 163–173.

Roth, J., Hashimshony, R., & Wachman, A. (1985). Generating layouts with non-convex envelopes. *Building and Environment*, *20*(4), 211–219.

Steadman, J. P. (1983). *Architectural morphology : an introduction to the geometry of building plans*. (Pion, Ed.) (First). London: Pion.

Suter, G., Petrushevski, F., & Šipetić, M. (2014). Operations on network-based space layouts for modeling multiple space views of buildings. *Advanced Engineering Informatics*, *28*(4), 395–411.

Thompson D’A W, (1961) On Growth and Form (Cambridge University Press, Cambridge abridge by JT Bonner from the 1917 edition.

Woodbury, R. (2010). *Elements of parametric design* (1st ed.). London: London [u.a.] : Routledge.

Wikipedia contributors. (2018, October 21). Quadrilateral. In Wikipedia, The Free Encyclopedia. Retrieved 13:20, October 23, 2018, from https://en.wikipedia.org/w/index.php?title=Quadrilateral&oldid=865031144

Wikipedia contributors. (2018, August 13). Path (graph theory). In Wikipedia, The Free Encyclopedia. Retrieved 13:21, October 23, 2018, from https://en.wikipedia.org/w/index.php?title=Path_(graph_theory)&oldid=854763314

Wikipedia contributors. (2018, September 4). Degrees of freedom (statistics). In Wikipedia, The Free Encyclopedia. Retrieved 13:40, October 23, 2018, from https://en.wikipedia.org/w/index.php?title=Degrees_of_freedom_(statistics)&oldid=857983167

GLOSSARY

Adjacency Graph (AG)	The graph representing the adjacency between rooms. Vertices representing rooms and edges represent adjacency between them. (Roth et al., 1982). See also dual graph.
Bipartite Graph	A graph whose set of vertices can be put into two subsets A and B in such a way that each edge of the graph joins a vertex in A and a vertex in B (Aldous, 2005).
Cell	Referring to room or small unit of a room, which is part of unit plan (Roth et al., 1982).
Coloured Directed Graph (CDG)	Coloured Directed Graph is a Adjacency Graph with assigned arrows (directed edges) in two directions are representing adjacencies between rooms through walls in both in x and y direction, vertices represent faces and edges represent adjacency (Roth et al., 1982).
Coloured Directed Sub-Graph (CDSG)	Subgraph of the Coloured Directed Graph, representing adjacency between rooms through walls only in one direction either x or y. Vertices represent faces. Edges represent adjacency between them. (Roth et al., 1982).
Common cut line	Common cut line or cut line referring to the Roth method of generating a cut line in a Coloured Directed Subgraph (CDSG). The rules of generating cut lines are that any edge coming out or entering the common vertex, generates a cut line. If any edge has more than one cut line, then these cut lines will be merged into a single cut line (Roth et al., 1982).
Common wall	The wall that divides two rooms which are joined together and belongs to both or the wall that belongs to both of the rooms or more.
Continuator square	Continuator square is a square in a rectilinear polygon such that the intersection of its own boundary and the boundary of rectilinear polygon is continuous (Wikipedia, n.d.).
Convex polygon	A polygon with all its interior angles less than 180°.
Convex orthogonal polygon	A polygon with all its interior angles equal to 90°.
Degree of Freedom (DOF)	The degree of freedom is the number of the independent parameters which are defined by its configuration in the system (Wikipedia, n.d.).
Dimension constraint	Dimensional constraints will control the proportions of unit plan and room dimensions done by designer. They can be constraints of distances between objects, or between points of objects (Autodesk, 2017).

Dimension Sub-Graph (DSG)	Graph which is converted from Coloured Directed Sub-Graph (CDSG) by cutting procedure (describe in J. Roth method). This graph also is directed by arrows from one direction to another as flow network. Vertices represent common walls and edge represent distance between them (Roth et al., 1982).
Dipath or directed path	"A directed path, (sometimes called dipath) is a sequence of edges which connects a sequence of vertices, but with the added restriction that the edges should all be directed in the same direction" (Wikipedia, n.d.).
Directed Graph or Digraph	The graph which consist of directed edges is called the directed graph or the digraph (Aldous, 2005).
Dual graph	Dual Graph is a graph that has a vertex for each face of Planar Graph G. The dual graph has an edge whenever two faces of graph G are separated from each other by an edge (Aldous, 2005).
Face	Face is a close loop of planar graph in graph theory, the term will be used to represent a room. See also cell.
Graph	In the graph theory, a graph consists of a set of points, called vertices, joined by lines, called edges; each edge joins exactly two vertices (Aldous, 2005).
Interior wall	The wall in an apartment unit
Non-convex polygon	A polygon which have at least one of its interior angles equal to the range between 180 degrees and 360 degrees (Wikipedia, n.d.).
Non-convex orthogonal shape	A polygon or shape which will always have at least one of its interior angles equal to 270 degrees.
Path	"In graph theory, a path is a finite or infinite sequence of edges which connects a sequence of vertices which, by most definitions, are all distinct from one another" (Wikipedia, n.d.).
Plan Graph	Graph representing unit plan.
Planar Graph	The graph, where the edges are not in intersection with each other (Aldous, 2005).
Quadrilateral	Quadrilateral is a polygon with four edges and four vertices and sum of all interior angle equal to 360° (Wikipedia, n.d.).
Rectilinear polygon	Rectilinear polygon or orthogonal polygon is a polygon with all their edge intersections are at right angles, and edges parallel to the axes of Cartesian coordinates (Wikipedia, n.d.).

Sink	Sink refers to a dimension subgraph as a flow network, there is a single sink from which no edge comes out (Roth et al., 1982).
Source	Source refers to the dimension subgraph as a flow network, there is a single source, into which no edge enters (Roth et al., 1982).
Source unit plan	An apartment unit with specific topology as a template.
Square orthogonal polygon	A polygon or shape which are rectangle and have four edges (see also convex rectilinear polygon).
Target unit plan	A unit plan as output of an insertion of rooms topology in a target boundary editing operation.
Unit boundary	The boundary of an apartment unit which may be defined by boundary walls.
Weight graph	Weighted graph is a graph in with a number (the weight) which is assigned to each edge. Such weights might represent for example costs, lengths or capacities, depending on the problem at hand (Aldous, 2005).
Weighted directed graphs	It is mixed graph between directed graph and weight graph.
x-wall	The wall which is placed parallel to x-axis (Roth et al., 1982).
y-wall	The wall which is placed parallel to y-axis (Roth et al., 1982).