Artificial Intelligence for Engineering Design, Analysis and Manufacturing

cambridge.org/aie

Graph-based approach for enumerating floorplans based on users specifications

Krishnendra Shekhawat, Rahil N. Jain, Sumit Bisht, Aishwarya Kondaveeti and Dipam Goswami

Department of Mathematics, BITS Pilani, Pilani Campus 333031, India

Research Article

Cite this article: Shekhawat K, Jain RN, Bisht S, Kondaveeti A, Goswami D (2021). Graph-based approach for enumerating floorplans based on users specifications. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* **35**, 438–459. https://doi.org/10.1017/S0890060421000275

Received: 2 September 2021
Accepted: 8 September 2021

Key words:
Algorithm; circulation; floorplans; graph theory; layouts; symmetry

Author for correspondence:
Krishnendra Shekhawat,
E-mail: krishnendra.iitd@gmail.com

Abstract

This paper aims at automatically generating dimensioned floorplans while considering constraints given by the users in the form of adjacency and connectivity graph. The obtained floorplans also satisfy boundary constraints where users will be asked to choose their preferred location based on cardinal and inter-cardinal directions. Further, spanning circulations are inserted within the generated floorplans. The larger aim of this research is to provide alternative architecturally feasible layouts to users which can be further refined by architects.

Introduction

The floorplanning automation has been well studied in the literature where different approaches, such as artificial intelligence, graph theory, shape grammar, and optimization, have been widely used. In this paper, we present a graph-theoretic approach for the automated generation of multiple floorplans.

A floorplan is a polygon, which is the plan boundary, divided by straight lines into component polygons called rooms. Corresponding to a floorplan, there associates a graph called dual graph, which is obtained by replacing each room by a vertex and any two vertices are adjacent, that is, joined by an edge, if the corresponding rooms share a wall or a section of a wall. For example, in Figure 1, the dual graph (Fig. 1b) of a floorplan (Fig. 1a) is shown. Constructing the dual graph of a floorplan is easy but deriving a floorplan from the dual graph (in this case, we call the dual graph as an adjacency graph) is not obvious. For example, Figure 1c represents an adjacency graph for which constructing a floorplan manually is a difficult task. Also, it is interesting to observe that the adjacency graph represents the geometrical and topological constraints of a floorplan. For example, the shape of the room is not provided explicitly by the user but from the graph in Figure 1b; it can be observed that at least one of the rooms needs to be non-rectangular (room 1 is non-rectangular in Fig. 1a) because of the presence of separating triangle $\triangle 135$ (refer to Definition 4). Further, the position of a vertex (in the given embedding of a graph) as internal or external vertex guides the position of the corresponding room. In Figure 1a, room 5 is internal while room 2 is external as suggested by the given graph.

Two rooms are adjacent if they share a wall among themselves where sharing a wall may not imply that the corresponding rooms are connected, mainly via a door. The connectivity of rooms is given by the connectivity graph which must be a sub-graph of the adjacency graph. For example, Figure 1e,f represents adjacency and connectivity graphs, respectively, and the corresponding floorplan is shown in Figure 1g. In this paper, we present graph-theoretic algorithms for generating a set of floorplans corresponding to a given adjacency graph and connectivity graph while satisfying dimensional constraints given in the form of width, height, and aspect ratio of each room. Further, we give an algorithm for inserting circulation in the obtained floorplan.

Literature review

The generation of floorplans using graph-theoretic tools has been well studied in the literature. It started in the 1960s (Levin, 1964) and then in the subsequent years, a lot of research has been done which is focused on enumeration and construction of floorplans corresponding to a given graph (Steadman, 1973; Mitchell *et al.*, 1976). Koźmiński and Kinnen (1985) gave a necessary and sufficient condition for the existence of rectangular floorplans (RFPs) corresponding to 1-connected and 2-connected planar triangulated graphs (PTGs). Bhasker and Sahni (1988) gave a linear time algorithm for the construction of a RFP corresponding to 2-connected PTGs. At the same time, Rinsma (1987) discussed the existence of RFPs for given adjacencies and dimensions.

Yeap and Sarrafzadeh (1993) realized that there are graphs for which RFPs do not exist. Liao *et al.* (2003) gave a linear time algorithm for constructing orthogonal floorplans

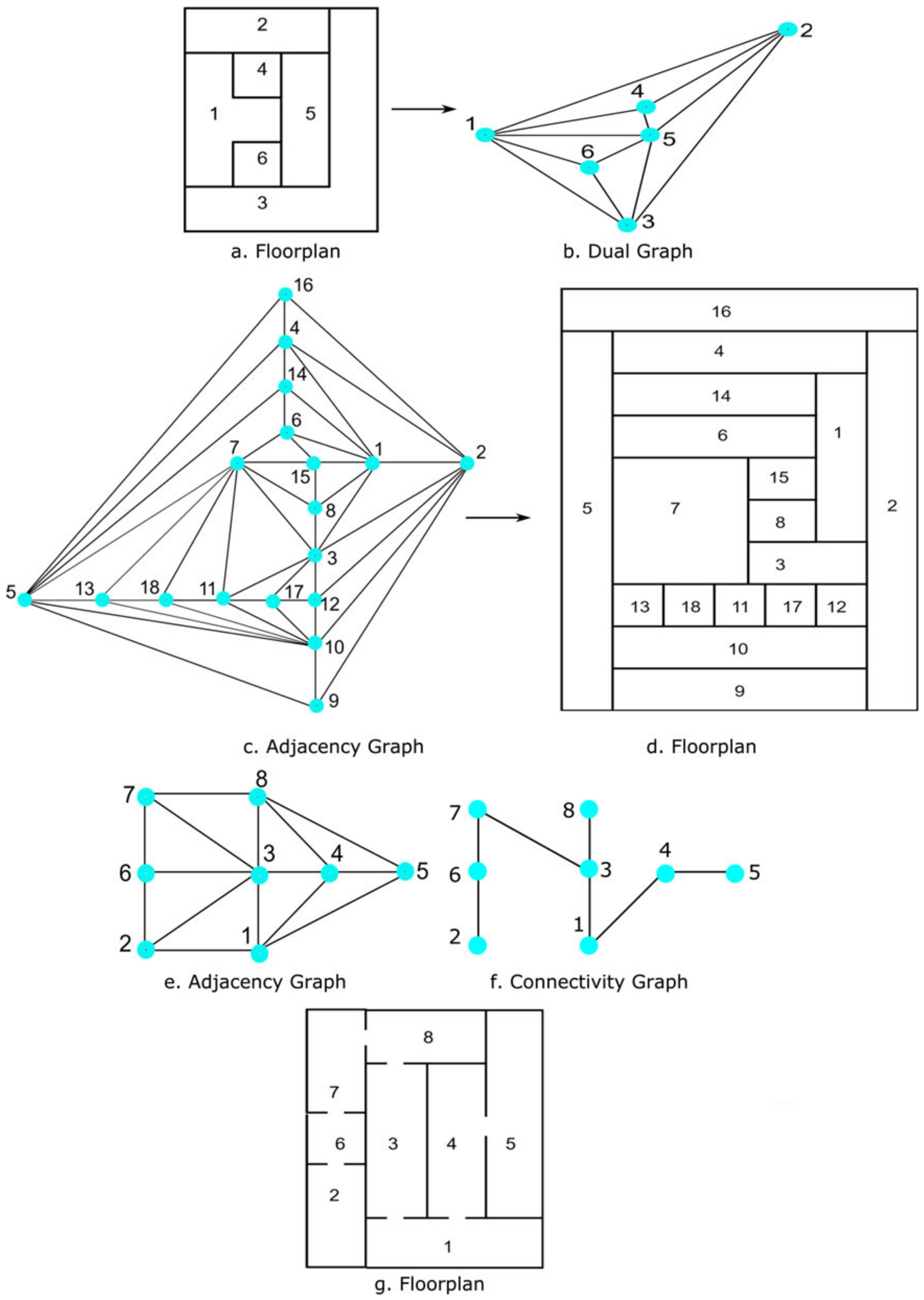


Fig. 1. Adjacency and connectivity graph and corresponding floorplans.

(OFPs) for maximal planar graphs. Marson and Musse (2010) generated sliceable floorplans having aspect ratios close to one, without considering the adjacency constraints. Eppstein *et al.* (2012) gave a necessary and sufficient condition for an RFP to be area universal.[1] They also proposed the construction of area-

[1]An RFP is area universal if any assignment of areas to rectangles can be realized by a rectangular room.

universal RFPs. Alam *et al.* (2013) gave the construction of area-universal OFPs for maximal planar graphs. Wang *et al.* (2018) introduced prototype GADG (graph approach to floor plan generation) for regenerating an existing RFP by considering the underlying graph of the existing floorplan. In the same year, Shekhawat and Duarte (2018) introduced generic RFPs and Shekhawat (2018) proposed an algorithm for their enumeration. Upasani *et al.* (2020) developed a prototype which transforms a drawn rectangular arrangement into a dimensioned RFP while satisfying adjacency and symmetric requirements. In the same year, Wang and Zhang (2020) extended GADG (Wang *et al.*, 2018) for generating dimensioned floorplans corresponding to user-specified design requirements. Shi *et al.* (2020) used reinforcement learning-based heuristic search technique (Monte Carlo Tree Search) to produce a closest RFP corresponding to any given adjacency graph.

There also exist other methodologies for generating floorplans. Shape grammar is an efficient approach for automation where the idea is to generate designs through the execution of shape rules (Mitchel, 1990; Duarte, 2005; Müller *et al.*, 2007; Wu *et al.*, 2014). Evolutionary approaches are also well studied for this purpose (Rodrigues *et al.*, 2013). Wu *et al.* (2018) presented a hierarchical framework and used mixed-integer quadratic programing for the dimensioning of layouts with fixed exterior boundaries. In recent times, many new approaches have evolved for supervised floorplan automation (Merrell *et al.*, 2010; Wu *et al.*, 2019). Hu *et al.* (2020) presented the generation of floorplans using a graph neural network while considering room adjacencies in the form of layout graphs. Laignel *et al.* (2021) used a genetic optimization approach for the automated generation of apartment layouts where the area of rooms and boundary layout have been considered as input.

In this paper, we present a prototype based on graph-theoretic algorithms and optimization tools for generating multiple floorplans while satisfying given adjacency, boundary, and size constraints.

Gaps in the existing literature and proposed work

The gaps in the existing literature can be listed as follows:

(i) Floorplans for 1-connected PTGs
It can be easily seen in the literature that graph-theoretic algorithms for generating floorplans are restricted to 2-connected PTGs (Bhasker and Sahni, 1988; Kant and He, 1993; Yeap and Sarrafzadeh, 1993; Liao *et al.*, 2003; Alam *et al.*, 2013; Shekhawat *et al.*, 2021). In this paper, we propose a graph-theoretic algorithm for the construction of a floorplan corresponding to any given graph.

(ii) Dimensioned floorplans
In the 1980s, authors proposed the dimensioning of RFPs. Roth *et al.* (1982) suggested the use of PERT technique which often resulted in non-rectangular rooms. More recently, Wang and Zhang (2020) generate dimensioned floorplans with fixed aspect ratio constraints; however, a generalized optimization technique for producing feasible floorplans for any given room dimensions is still not available. In this paper, we consider minimum and maximum width, as well as area for each room as dimensional constraints, and using graph-theoretic and optimization tools, we construct a floorplan satisfying the given constraints.

(iii) Boundary constraints
Recently, Wang and Zhang (2020) considered boundary constraints for the generation of floorplans. In this paper, we propose a graph-based algorithm for generating multiple floorplans satisfying boundary constraints where users can choose the preferred location for each room based on cardinal and inter-cardinal directions.

(iv) Circulations
Baybars (1982) presented a graph-theoretical approach for inserting circulations within a floorplan. Recently, Egor *et al.* (2020) gave an evolutionary algorithm for inserting circulations. In this paper, we present an algorithm for generating floorplans along with spanning circulations.

(v) Time complexity
Many existing works can produce residential building layouts for a small number of rooms (Merrell *et al.*, 2010; Wu *et al.*, 2018; Nisztuk and Myszkowski, 2019), but for the complex building structures with a large number of rooms, we need efficient algorithms. The proposed prototype takes $O(N^2)$ space for generating floorplans for any given number of rooms N.

Preliminaries

In this section, we present a few important terminologies which are used frequently in literature and also throughout this paper.

Definition 1 **(Floorplans)** Floorplans can be categorized on the basis of the shape of rooms and the boundary of a layout. If the boundary and all the rooms of a plan are rectangular, we call it rectangular floorplan, denoted as RFP. If the boundary is rectangular and at least one of the rooms is orthogonal, we call it orthogonal floorplan, denoted as OFP. Floorplans other than RFPs and OFPs are termed as non-rectangular floorplans (NRFPs). For example, the floorplan in Figure 1a is an OFP, in Figure 1d is an RFP, and in Figure 10e is an NRFP. Unless specified, a floorplan refers to a dimensionless floorplan. In the literature, floorplans are generally without empty spaces but we show that for non-triangulated graphs, floorplans have empty spaces (Fig. 11b).

Two rooms in a floor plan are adjacent if they share a wall or a section of it, where a wall of a room refers to the edges forming its perimeter.

Definition 2 **(Graphs)** A graph G is a finite nonempty set $V(G)$ of vertices and a (possibly empty) set $E(G)$ of 2-element subsets of $V(G)$ called edges. G is said to be planar if it can be embedded in the plane without crossing of edges; otherwise, it is a non-planar graph (the graph in Fig. 2a is non-planar while the graph in Fig. 2b is planar whose planar embedding is shown in Fig. 2c). A plane graph is a planar graph with an embedding dividing the plane into connected components called faces/regions (the graph in Fig. 2c has five internal faces and one exterior face).

A floorplan itself is a planar-connected graph where each room is the face of the floorplan. Therefore, we are considering planar-connected graphs only and in the paper, a graph shall mean a connected planar graph. In architectural terms, when the given graph provides a specific neighborhood of each room, we call it an adjacency graph, denoted as G_A. A connectivity graph, denoted as G_C, is a sub-graph of the adjacency graph representing the connectivity of rooms (via a door). Figure 1e,f represents adjacency and connectivity graphs, respectively.

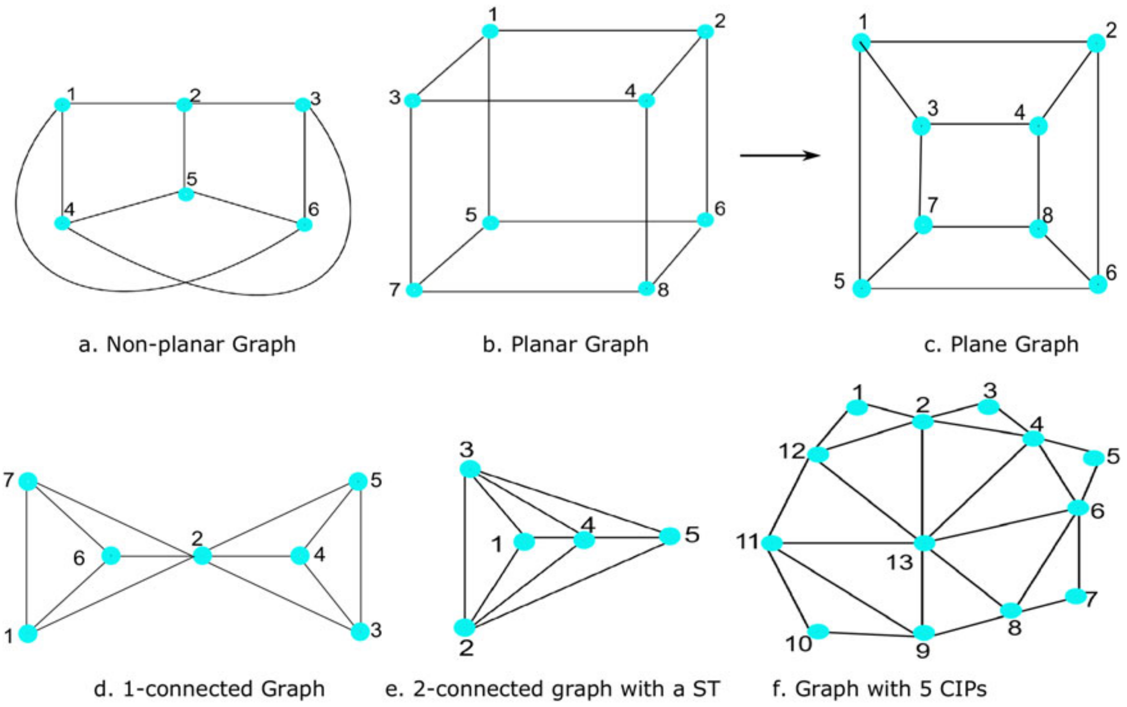


Fig. 2. Illustration for different terminologies.

Definition 3 **(Graph connectivity)** A graph G is said to be connected if there is a path from each vertex to any other vertex. It is obvious that the adjacency graph is always planar and connected. A connected graph G is said to be 1-connected if it has a cut-vertex. A vertex v is said to be a cut-vertex if its removal disconnects the graph. G is said to be bi-connected or 2-connected if it has no cut-vertices. The graph in Figure 2d is 1-connected having vertex 2 as a cut-vertex while the graph in Figure 2e is 2-connected with no cut-vertices.

Definition 4 **(Triangulation)** A connected planar graph is triangulated if all of its faces (except the exterior) are triangular; the exterior face may or may not be a triangle. This graph is known as inner PTG (the graphs in Figure 2d–f are PTGs).

A separating triangle $\triangle abc$ is a cycle of length three in a graph G such that $G - \triangle abc$ is a disconnected graph. It is denoted as ST. $\triangle 234$ in Figure 2e is a separating triangle.

A PTG with no separating triangle and with the exterior face of the length at least 4 is called properly triangular planar graph (PTPG) (Bhasker and Sahni, 1987). The graph shown in Figure 2f is a PTPG, whereas the graphs shown in Figure 2d,e are not PTPGs because of the presence of STs.

Definition 5 **(Corner implying path (CIP) (Bhasker and Sahni, 1987))** A shortcut in a planar bi-connected graph G is an edge that is incident to two vertices on the outer boundary of G but is not a part of the outer boundary. The edge (2, 4) in Figure 2f is a shortcut. A CIP in a planar bi-connected graph G is a path $u_1, u_2, \ldots, u_n$ on the outer boundary of G with the property that (u_1, u_n) is a shortcut and $u_2, u_3, \ldots, u_{n-1}$ are not the endpoints of any shortcut. 2–3–4 in Figure 2f is a CIP.

Definition 6 **(Regular edge labeling (REL) (Kant and He, 1993))** An REL of a bi-connected PTPG G is a partition of the interior edges of G into two subsets T_1, T_2 of directed edges such that for each interior vertex u, the edges incident to u appear in a counterclockwise order around u as follows: a set of edges in T_1 leaving u, a set of edges in T_2 entering u, a set of edges in T_1 entering u, and a set of edges in T_2 leaving u.

Let N, E, S, W be the four exterior vertices in clockwise order. All interior edges incident to N are in T_1 and entering N. All interior edges incident to E are in T_2 and entering E. All interior edges incident to S are in T_1 and leaving S. All interior edges incident to W are in T_2 and leaving W. An REL for the PTPG in Figure 4a is shown in Figure 4c.

Methodology

In this section, we present the construction of floorplans with circulations for any given adjacency and connectivity graphs while satisfying boundary and dimensional constraints.

Floorplans for 1-connected and non-triangulated graphs

In the literature, the construction of floorplans is restricted to 2-connected PTGs only. Koźmiński and Kinnen (1985) proved that there exists an RFP for a 2-connected PTPG if and only if it has at most four CIPs. Recently, Shekhawat *et al.* (2021) presented a prototype GPLAN which is capable of generating RFPs for any 2-connected PTPG with at most four CIPs and it generates OFPs for 2-connected PTGs having separating triangles or more than four CIPs (refer to Fig. 3). The steps of the algorithm for constructing a floorplan corresponding to a given PTPG are illustrated in Figure 4, which are briefly explained as follows:

(i) 4-completion: This step identifies rooms, which form the boundary of a floorplan based on cardinal directions

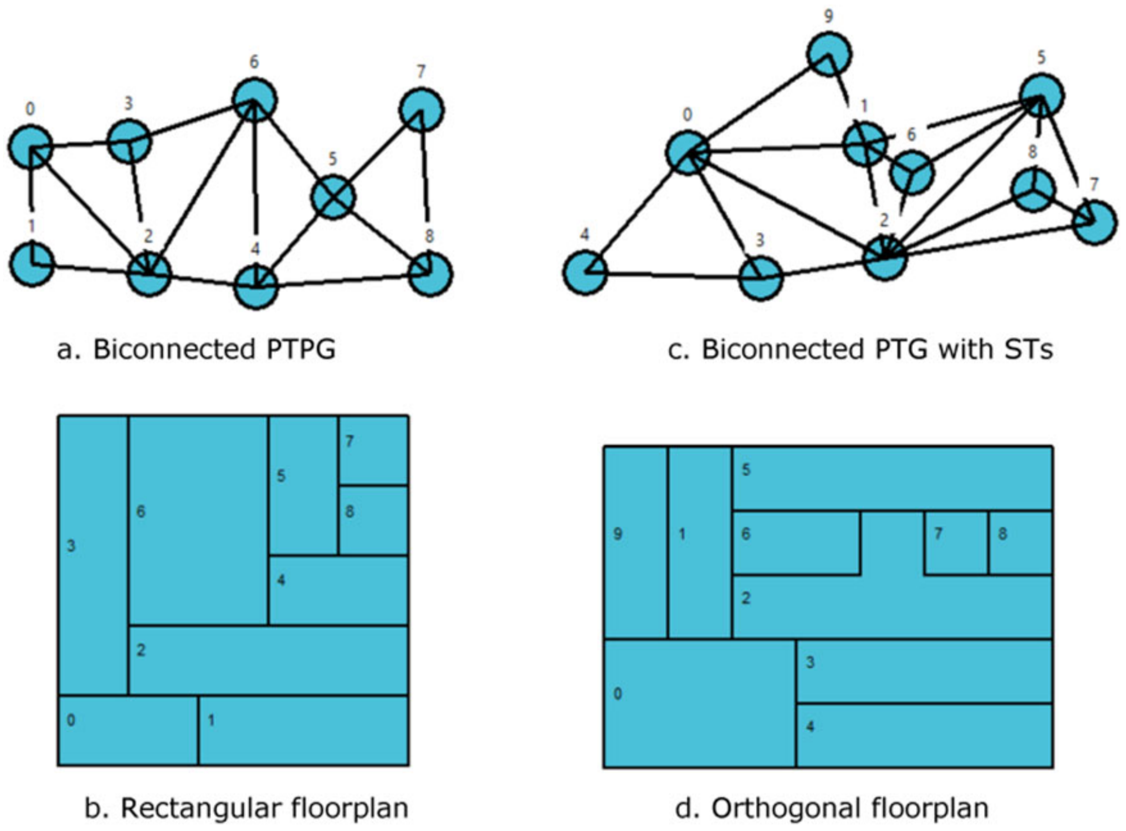


Fig. 3. Floorplans corresponding to bi-connected PTGs.

north, west, south, and east. Kant and He (1993) proposes the addition of four new vertices n, w, s, e representing north, west, south, and east, respectively, to a given bi-connected PTPG. This process requires 4 corner vertices on the outer boundary which are obtained by using the concept of CIPs (see Definition 5).

For the graph in Figure 4a, 4-completion is illustrated in Figure 4b where rooms 6 and 7 form north boundary and other exterior rooms form remaining boundaries.

(ii) REL: Constructing an REL of a given graph helps us in identifying horizontal and vertical adjacencies within a floorplan. Kant and He (1993) proposed an algorithm for transforming a bi-connected PTPG obtained by 4-completion into a REL. This algorithm first removes all the vertices in a described order and then adds the label T_1 and T_2 to the edges while adding the vertices. For a better understanding, refer to Figures 5–7 where a good vertex in a PTPG is an inner vertex with the following properties:
  (a) has degree 4 and (has at most 1 heavy neighbor or has 2 non-adjacent heavy neighbors)
  (b) has degree 5 and has at most 1 heavy neighbor

The contractible neighbor (of a good vertex) is defined as follows:

Let u be a neighboring vertex of a good vertex v, and y and z be their two common neighbors. If x ($\neq y, z$) is any other neighbor of v, then only mutual neighbor(s) of u and x must be v (or either of y or z) for u to be a contractible neighbor of v.

Based on the REL in Figure 4c, the horizontal and vertical adjacencies are shown in Figure 4d,e, respectively.

(iii) Floorplan: Using the horizontal and vertical adjacencies, next is to construct two separate floorplans satisfying these adjacencies as shown in Figure 4f,g, respectively, and then merging these floorplans gives us the required floorplan as given in Figure 4h. The merging of horizontal and vertical floorplans are illustrated in Figure 8.

In case of a non-triangulated (bi-connected) graph G, we first triangulate G by adding new edges to have a triangulated bi-connected graph G' (refer to Fig. 9a,b). For triangulation, the algorithm given by Berry *et al.* (2004) has been used. Each of the added edges is subdivided by introducing new vertices as shown in Figure 9c. The obtained graph is further triangulated to have a bi-connected triangulated graph (Fig. 9d) for which a floorplan is constructed (Fig. 9e). In the obtained floorplan, rooms corresponding to added vertices are removed to have a floorplan corresponding to the input graph as shown in Figure 9f. It is interesting to note here that any cycle C of length greater than 3 represents an empty space (can be seen as an atrium) in the corresponding floorplan surrounded by the rooms corresponding to the cycle C. For example, refer to the graph in Figure 9a having two cycles of length greater than 3 and corresponding floorplan is shown in Figure 9f having two atria.

In case of a 1-connected PTG G, we first biconnect it by adding new edges (refer to Fig. 10a,b). Then new vertices are introduced for sub-dividing the new added edges, which are followed by triangulation (see Fig. 10c). For the obtained graph, a floorplan is constructed from which rooms corresponding to added vertices are removed to have a required floorplan as shown in Figure 10d,e, respectively. 1-connected graphs are very useful in representing different blocks of a building connected through cut-vertices. For example, in Figure 11a, two blocks made up of rooms $\{R_1, R_2, R_3, R_4\}$ and $\{R_5, R_6, R_7, R_8\}$ are connected through room R_0.

When the adjacency graph is 1-connected and non-triangulated, we first biconnect it and then triangulate it by adding new edges. Using the proposed methodology, GPLAN is capable of generating floorplans for any given adjacency graphs (refer to Fig. 11b where using GPLAN, we obtained a floorplan corresponding to a 1-connected non-triangulated adjacency graph shown in Fig. 11a).

Connectivity graph

The information given by an adjacency graph explains a floorplan in terms of how rooms share a wall (full or partial) with each

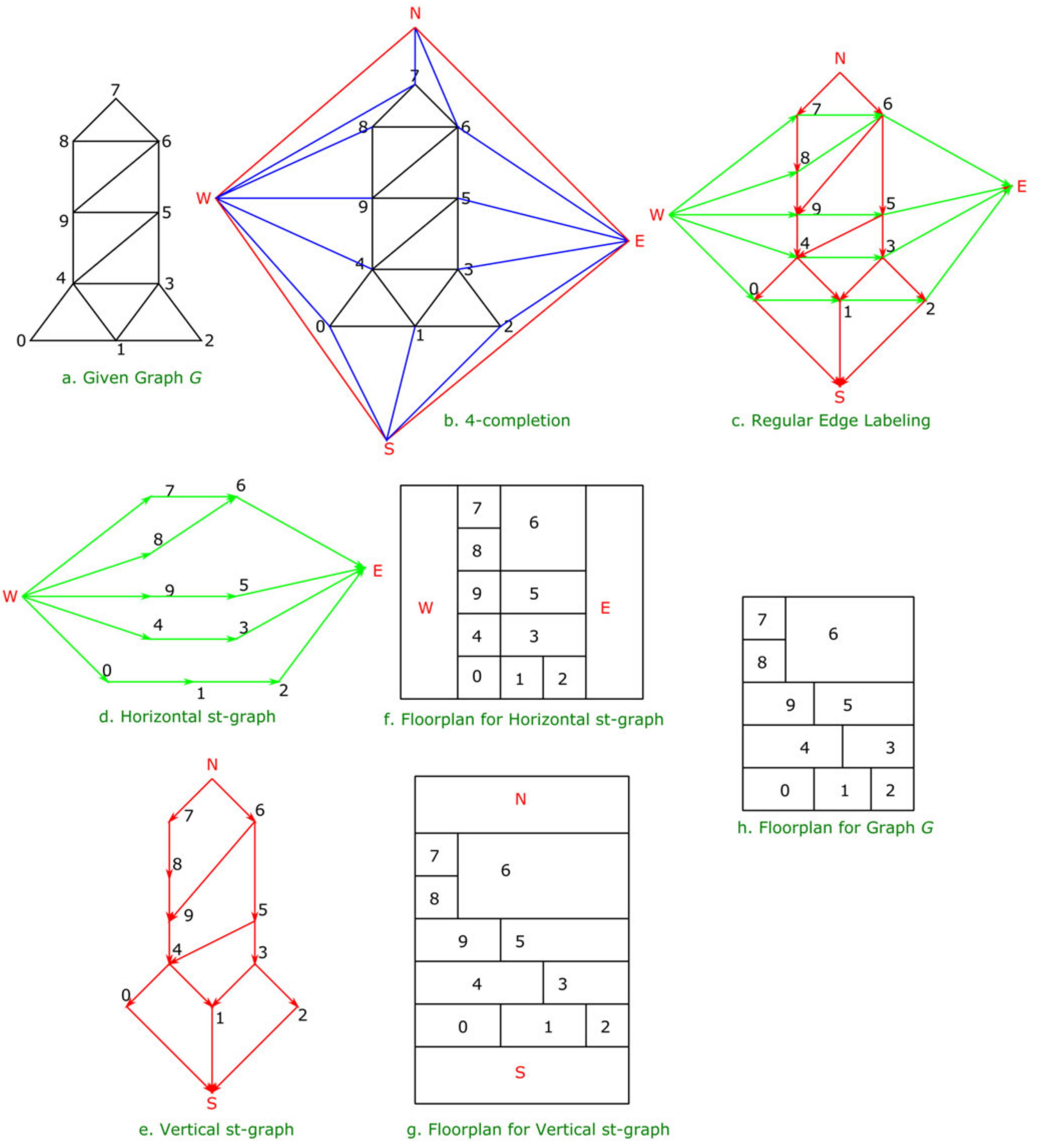


Fig. 4. Constructing a floorplan corresponding to a given bi-connected PTPG.

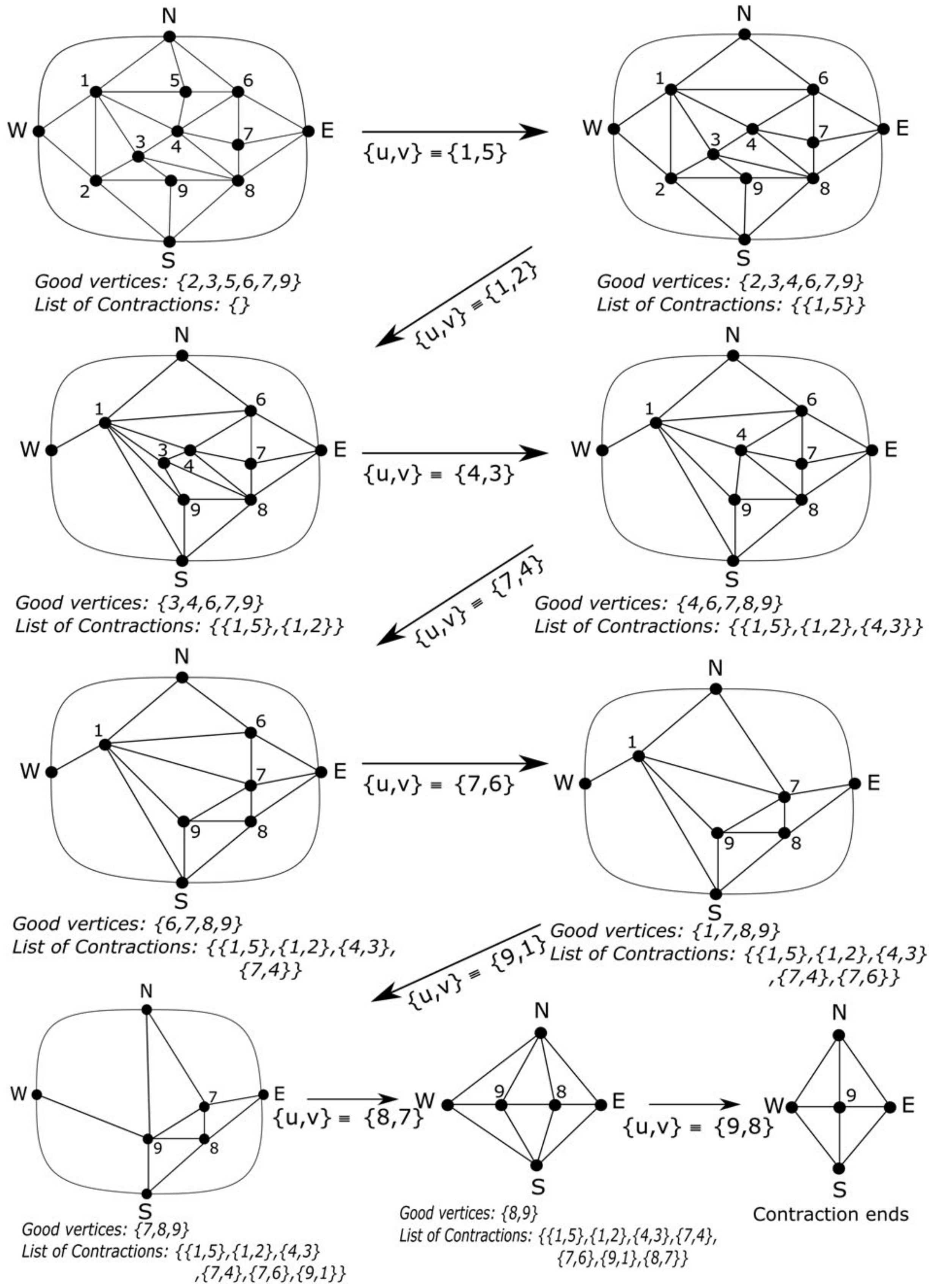

N
W
E
S
1
2
3
4
5
6
7
8
9
{u,v} = {1,5}
{u,v} = {1,2}
{u,v} = {4,3}
{u,v} = {7,4}
{u,v} = {7,6}
{u,v} = {9,1}
{u,v} = {8,7}
{u,v} = {9,8}
Good vertices: {2,3,5,6,7,9}
List of Contractions: {}
Good vertices: {2,3,4,6,7,9}
List of Contractions: {{1,5}}
Good vertices: {3,4,6,7,9}
List of Contractions: {{1,5},{1,2}}
Good vertices: {4,6,7,8,9}
List of Contractions: {{1,5},{1,2},{4,3}}
Good vertices: {6,7,8,9}
List of Contractions: {{1,5},{1,2},{4,3},{7,4}}
Good vertices: {1,7,8,9}
List of Contractions: {{1,5},{1,2},{4,3},{7,4},{7,6}}
Good vertices: {7,8,9}
List of Contractions: {{1,5},{1,2},{4,3},{7,4},{7,6},{9,1}}
Good vertices: {8,9}
List of Contractions: {{1,5},{1,2},{4,3},{7,4},{7,6},{9,1},{8,7}}
Contraction ends

Fig. 5. Edge contractions.

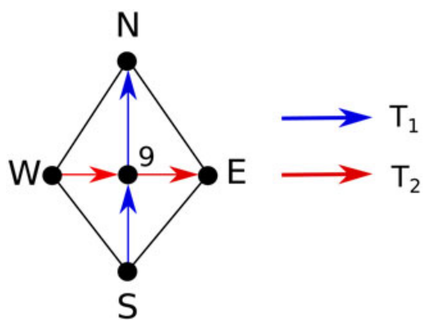


Fig. 6. Trivial regular edge labeling.

other. It represents the architectural program in terms of the set of rooms composing the plan and their relative position in terms of which spaces share walls with others. This is more informative of compositional concepts rather than the spatial relations between rooms (connections between rooms, which could be represented with a connectivity graph). In GPLAN, the user can give adjacency graph and connectivity graph simultaneously. For example, Figure 11c represents an adjacency graph (the complete graph with red and black edges) and a connectivity graph (graph induced by red edges) and Figure 11d represents the required floorplan where doors are corresponding to the edges of the connectivity graph.

Multiple floorplans

For a given adjacency graph, we aim to construct all possible floorplans. This can be done in two ways as follows:

(i) by considering each room adjacency in two ways as horizontal and vertical adjacency.

For example, refer to Figure 12 where floorplans corresponding to a given graph are topologically distinct because there exists at least a pair of rooms in these floorplans having different adjacency orientation (horizontal and vertical), that is, rooms 3 and 12 are horizontally adjacent in Figure 12b but they are vertically adjacent in Figure 12c. By looking at all possible combinations, GPLAN generates all topologically distinct floorplans. Eppstein *et al.* (2012) propose the concept of flippable edges and vertices to define a partial ordering on the family of all regular edge labeling of a given graph. A flippable edge is an edge e which is not adjacent to a degree-4 vertex, and the 4-cycle surrounding e is alternately labeled in the REL (edge (6, 9) is flippable in the REL in Fig. 13a). A flippable vertex v is a four-degree vertex such that the four-cycle surrounding v is alternately labeled in the REL (Fig. 13a has no flippable vertex). A new REL can always be obtained using a flippable edge or a flippable vertex (Eppstein *et al.*, 2012) (the REL in Fig. 13b is obtained from the REL in Fig. 13a by flipping the edge (6, 9) and corresponding floorplans are illustrated in Fig. 13c,d, respectively).

(ii) by considering different boundary constraints.

The vertices which lie on the exterior face of a graph (with a fixed embedding) correspond to the exterior rooms in a floorplan. Thus, the exterior vertices and corresponding rooms can be adjacent to the exterior through north, east, west, or south side. For example, rooms 3 and 12 are adjacent to the north in Figure 12b while in Figure 12c, rooms 4, 11, and 7 are adjacent to the north. Thus, GPLAN iterates through all possible boundaries to enumerate floorplans with different boundaries.

To generate all possible floorplans, we consider all possible boundary conditions and all combinations of horizontal and vertical room adjacencies. For example, for the graph in Figure 12a, GPLAN produces 6928 floorplans while considering 424 different boundaries as shown in Figure 12d. Similarly, for the graph in Figure 14, GPLAN constructs 101 different floorplans, a few of them are illustrated in Figure 15.

Time complexity

For a single floorplan generation, GPLAN first transforms the graph into a bi-connected triangulated graph (Berry *et al.*, 2004) which is an $O(N^2)$ algorithm where N refers to the number of rooms in the floorplan. GPLAN then performs 4-completion of the transformed graph, which takes linear time. The single floorplan is then generated using REL and merging horizontal and vertical floorplans using the algorithm proposed in Kant and He (1993). This algorithm runs in $O(N)$ time using an adjacency list as the data structure for the input graph. Combining these steps, GPLAN generates a floorplan in $O(N^2)$ time.

For multiple floorplans, GPLAN generates a family of all possible RELs for the graph. The algorithm proposed for the same by Eppstein *et al.* (2012) runs in polynomial time. These RELs are then transformed into floorplans by merging them using $O(N)$ algorithm proposed in Kant and He (1993). Combining these steps, GPLAN generates all possible topologically distinct floorplans in polynomial time.

GPLAN uses an adjacency list to store the input graph, which takes $O(N + M)$ space where M is the number of adjacencies in the floorplan. The output floorplan is stored in a list of tuples where each tuple contains the coordinates of the corner of each room. Eppstein *et al.* (2012) prove that the number of different RELs for a given graph is $O(N)$. For a single floorplan, the space complexity of GPLAN is $O(N + M)$. For multiple floorplans, GPLAN takes $O(N^2)$ space.

Dimensioned and symmetric floorplans

In Shekhawat *et al.* (2021), authors have presented a linear optimization model for producing dimensioned layouts corresponding to a given bi-connected PTG where dimensions are given in the form of minimum and maximum width, as well as the height of each room. In this paper, we have modified this model for 1-connected non-triangulated planar graphs. Further, in the previous model, the aspect ratio constraints for each room were not considered which may have led to very thin rooms which are not architectural acceptable. Hence, we have introduced aspect ratio constraints for each room. Refer to Figure 16, where for the given graph and dimensional constraints, a dimensioned floorplan is shown. Similarly, a dimensioned floorplan for the 1-connected graph in Figure 16d is shown in Figure 16e. Also, if the dimensional constraints are not feasible, GPLAN generates an error as shown in Figure 17d where dimensional constraints in Figure 17c are not feasible. The linear optimization model corresponding to horizontal and vertical st-graphs is given as follows:

$$\begin{aligned}\text{Minimize}: \quad & \sum w(e_{j,i}) - d_i^{\max} \\ \text{suchthat}: \quad & \sum w(e_{ji}) = \sum w(e_{ik}) \quad \forall i \in V(G) \quad , \\ & \min(d_i) \leq \sum w(e_{ji}) \leq \max(d_i) \quad \forall i \in V(G)\end{aligned} \tag{1}$$

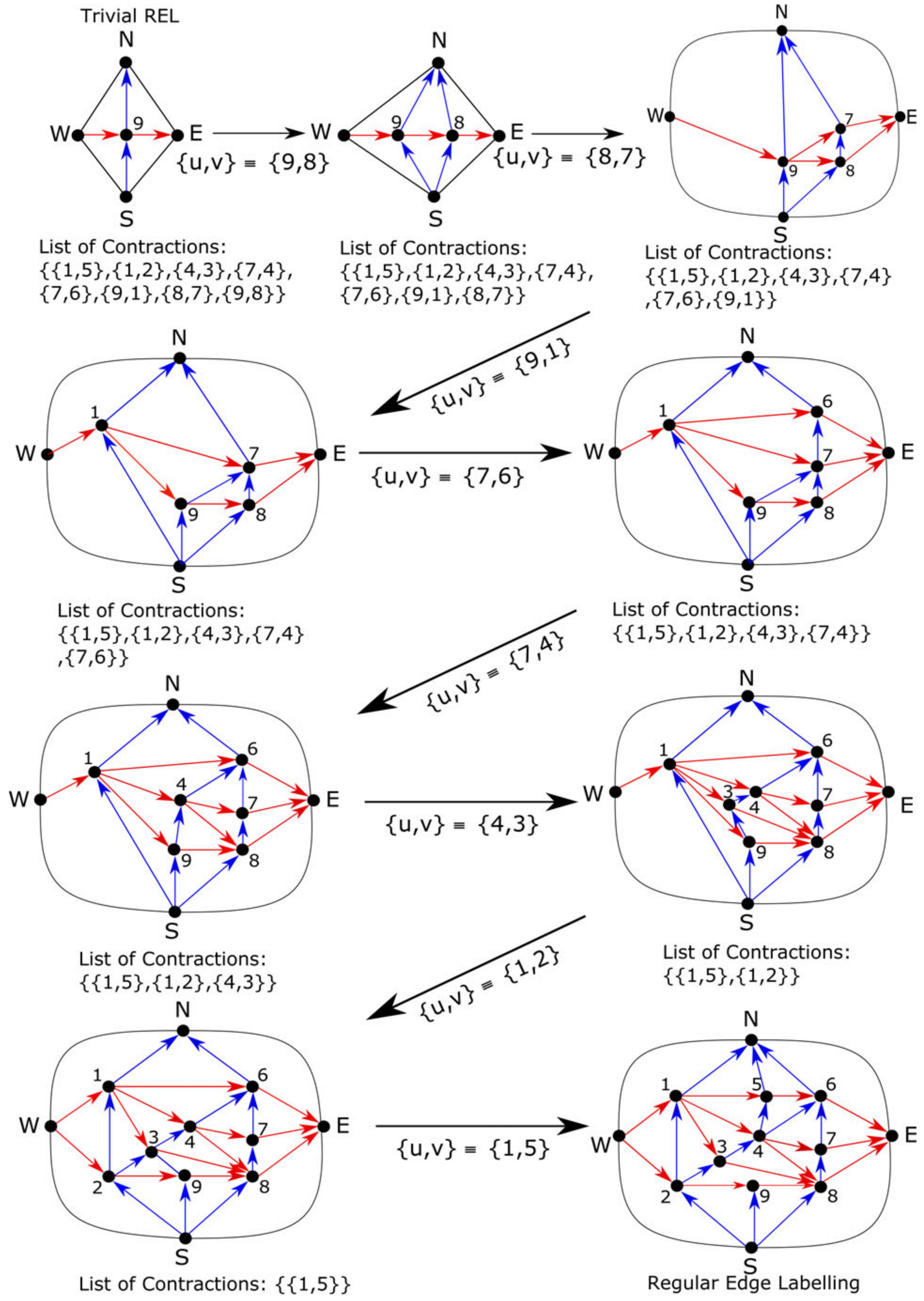

Trivial REL
N
W
E
S
9
8
7
{u,v} ≡ {9,8}
{u,v} ≡ {8,7}
List of Contractions: {{1,5},{1,2},{4,3},{7,4},{7,6},{9,1},{8,7},{9,8}}
List of Contractions: {{1,5},{1,2},{4,3},{7,4},{7,6},{9,1},{8,7}}
List of Contractions: {{1,5},{1,2},{4,3},{7,4},{7,6},{9,1}}
{u,v} ≡ {9,1}
{u,v} ≡ {7,6}
List of Contractions: {{1,5},{1,2},{4,3},{7,4},{7,6}}
List of Contractions: {{1,5},{1,2},{4,3},{7,4}}
{u,v} ≡ {7,4}
{u,v} ≡ {4,3}
List of Contractions: {{1,5},{1,2},{4,3}}
List of Contractions: {{1,5},{1,2}}
{u,v} ≡ {1,2}
{u,v} ≡ {1,5}
List of Contractions: {{1,5}}
Regular Edge Labelling

Fig. 7. Edge expansions.

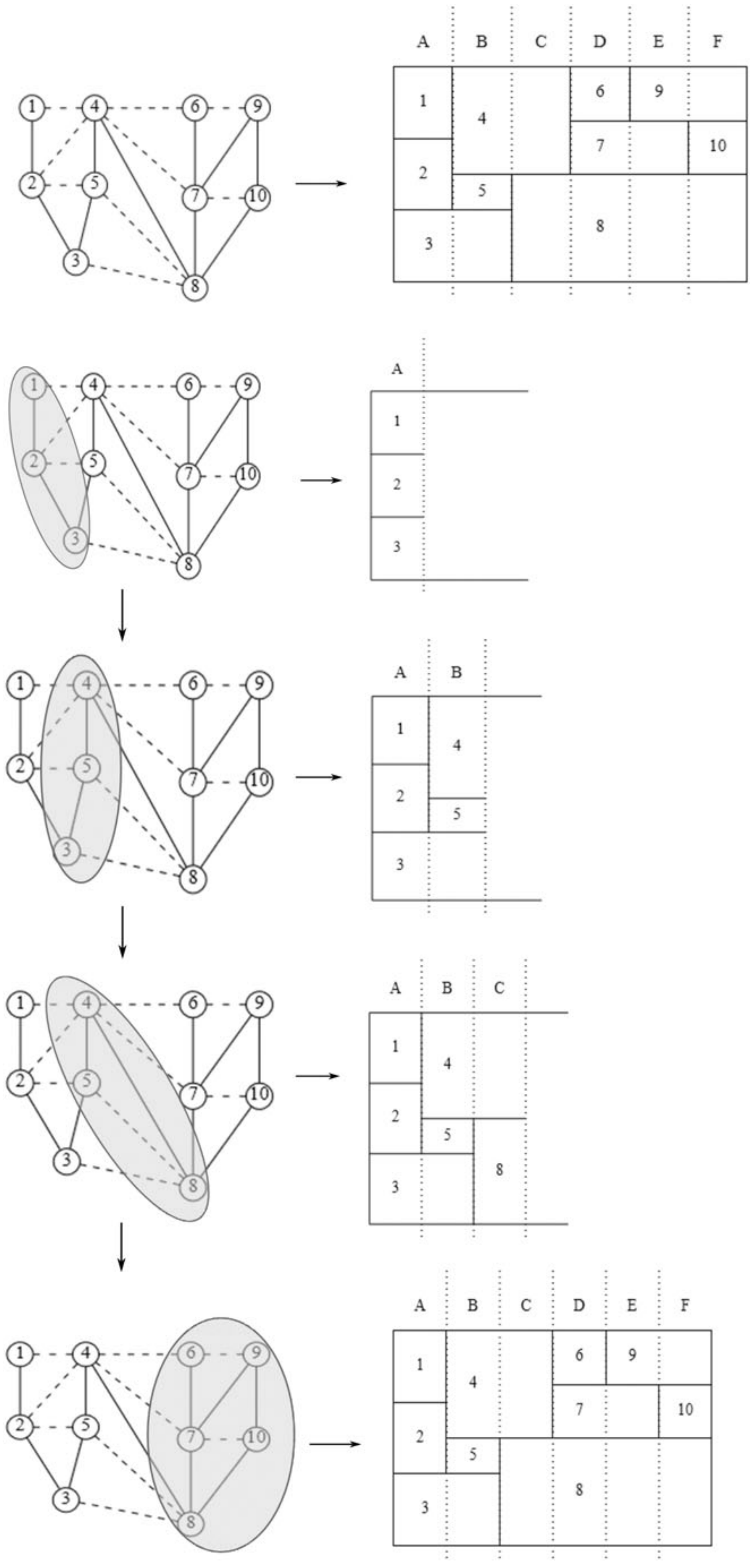


Fig. 8. The steps of construction of an RFP from the obtained REL.

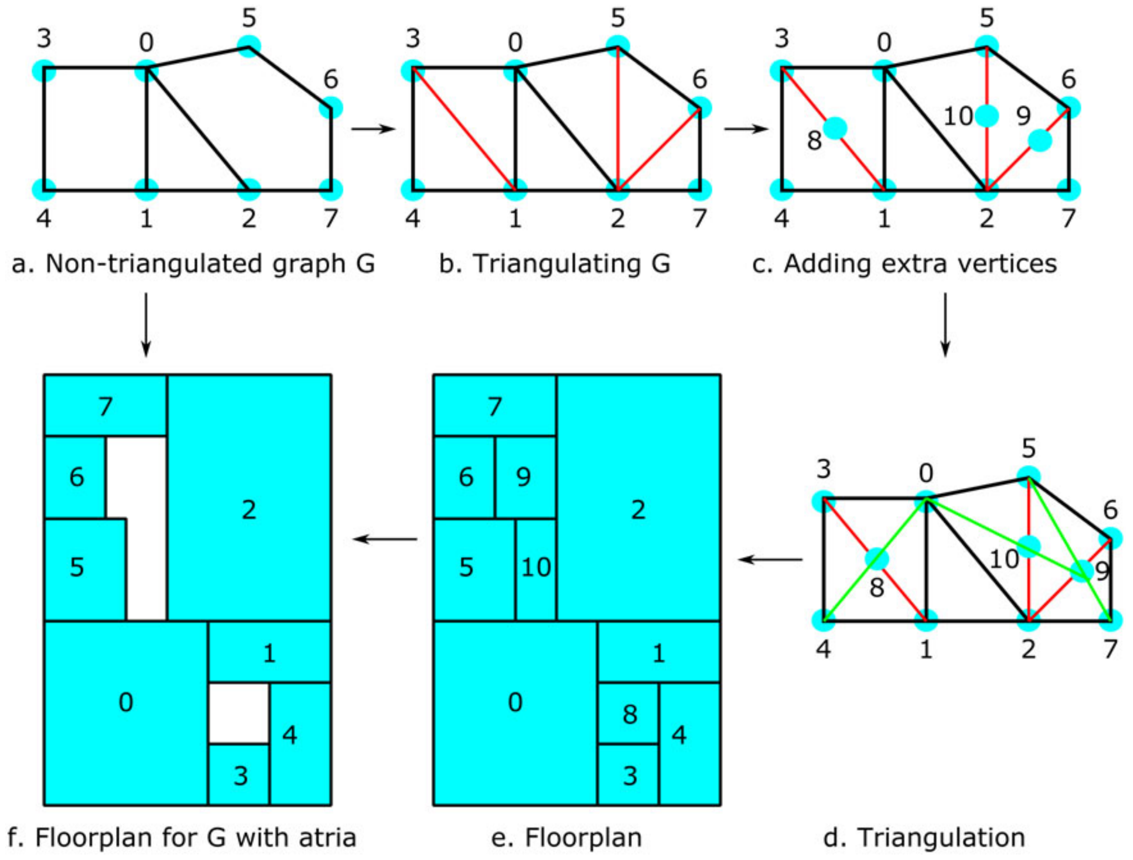


Fig. 9. Floorplan corresponding to a non-triangulated bi-connected PTG.

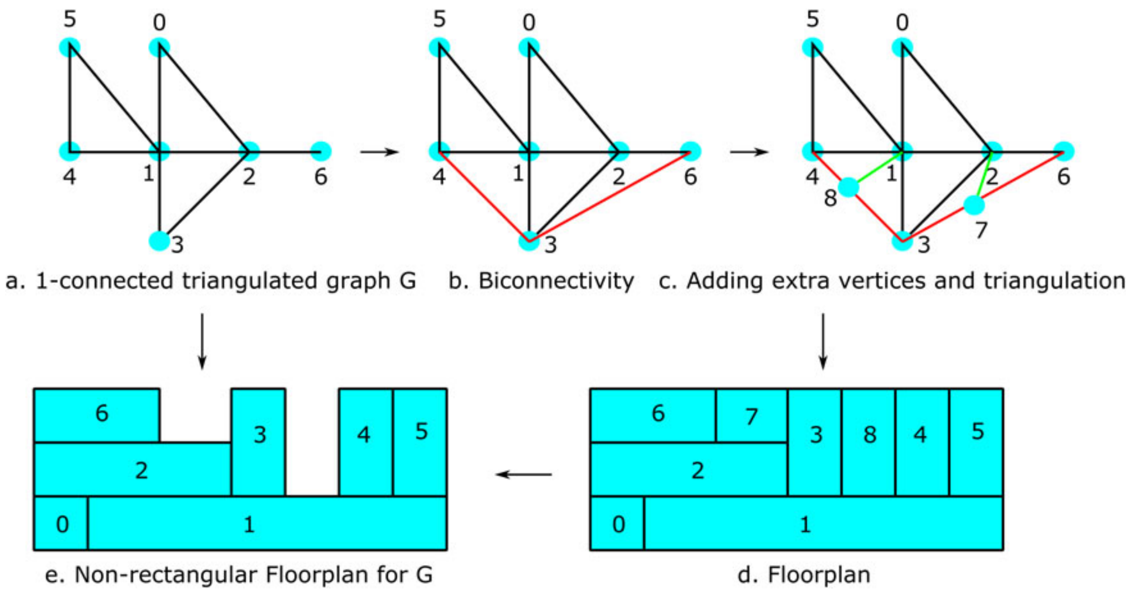

a. 1-connected triangulated graph G
b. Biconnectivity
c. Adding extra vertices and triangulation
d. Floorplan
e. Non-rectangular Floorplan for G

Fig. 10. Floorplan corresponding to a 1-connected PTG.

where $d_i^{\max}$ is the maximum dimension (width/height) of room i, $\sum w(e_{ji})$ denotes the total inflow, and $\sum w(e_{ik})$ denotes the total outflow from vertex i.

Taking these constraints into account, GPLAN optimizes width and height separately using the dual-simplex method to generate a feasible dimensioned floorplan. For a better understanding, refer to Tables 1–4 giving constraints corresponding to the st-graphs in Figure 16d,e and dimensional constraints in Figure 16b.

In addition to it, we introduce rooms symmetry using the linear optimization model where either we can make two rooms

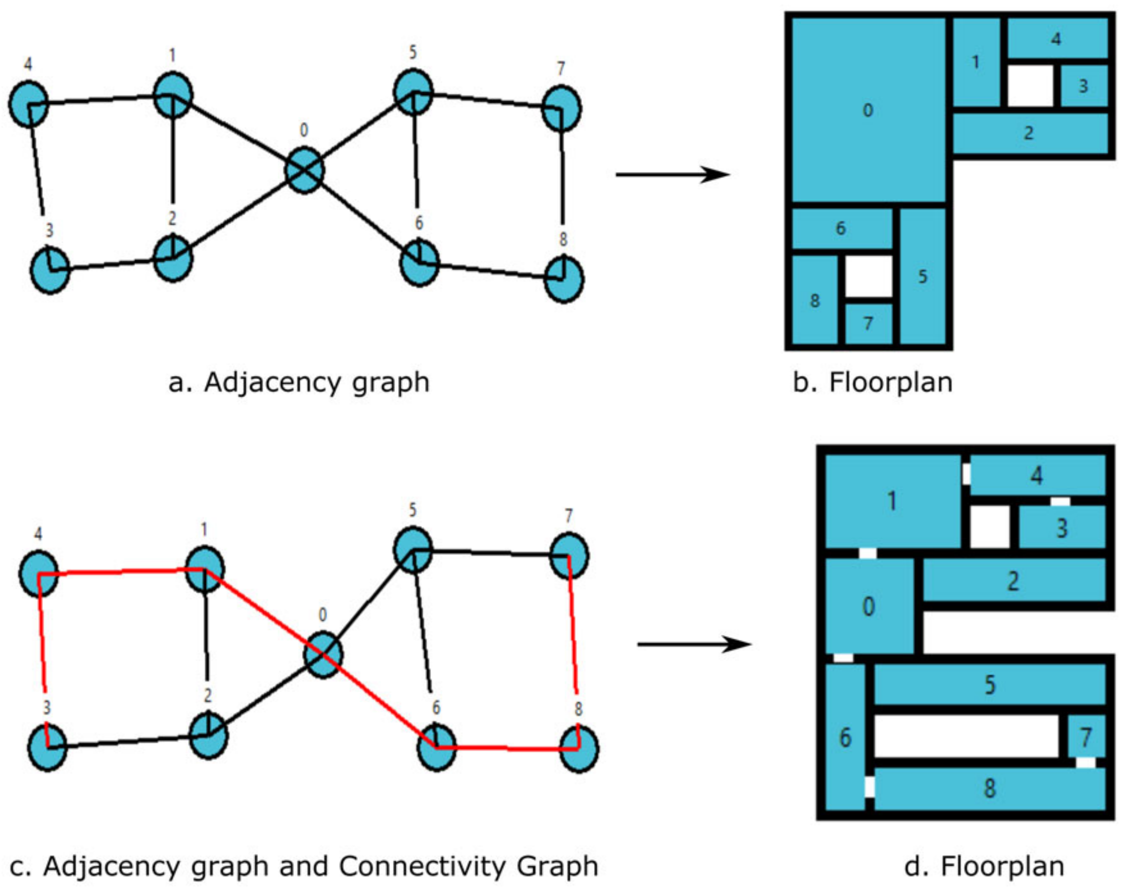
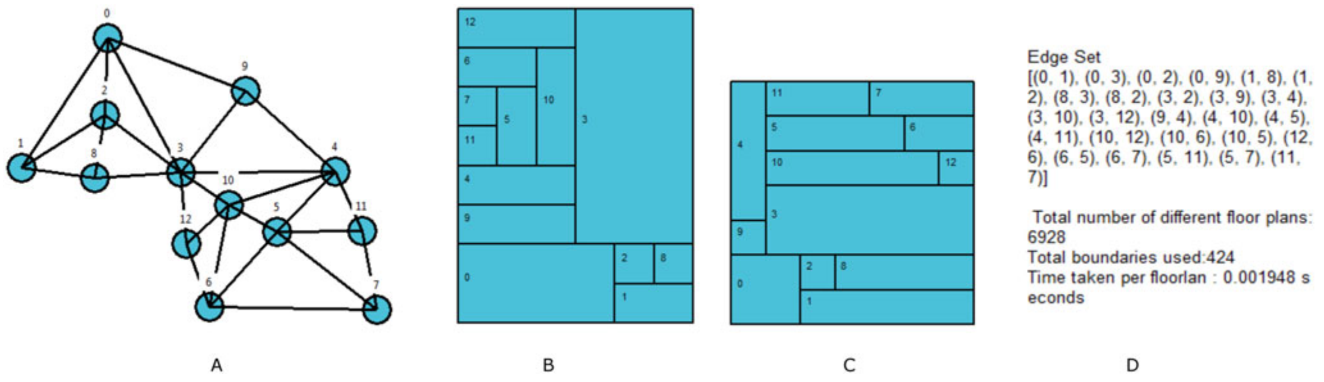


Fig. 11. Floorplans corresponding to adjacency graph and connectivity graph.



```
Edge Set
[(0, 1), (0, 3), (0, 2), (0, 9), (1, 8), (1,
2), (8, 3), (8, 2), (3, 2), (3, 9), (3, 4),
(3, 10), (3, 12), (9, 4), (4, 10), (4, 5),
(4, 11), (10, 12), (10, 6), (10, 5), (12,
6), (6, 5), (6, 7), (5, 11), (5, 7), (11,
7)]

 Total number of different floor plans:
6928
Total boundaries used:424
Time taken per floorlan : 0.001948 s
econds
```

Fig. 12. Different floorplans corresponding to a given graph.

symmetric to each other or we can make the combination of two adjacent rooms symmetric to a third room. Refer to the floorplan in Figure 16c where rooms 0 and 2 have the same dimensions as asked by the user. The concept of rooms symmetry can be very helpful in generating symmetric floorplans corresponding to symmetric graphs. The graph in Figure 17a is symmetric and if we ask rooms (0, 1), (2, 3), and (4, 5) to be symmetric, GPLAN produces a floorplan symmetric about x-axis and y-axis as shown in Figure 17b. In future, we try to develop an algorithm to identify symmetric graphs and to generate symmetric floorplans automatically.

Floorplans with boundary constraints

It has already been seen that GPLAN is capable of producing a lot of distinct layouts which are not easy to handle. To refine the set of all possible layouts, in this section, we impose boundary and dimensional constraints on these layouts. For the boundary constraints, users will be asked to choose the preferred location of each room based on cardinal and inter-cardinal directions. Also, among all possible layouts, we need to look for those layouts which satisfies dimensional restrictions. For example, for the

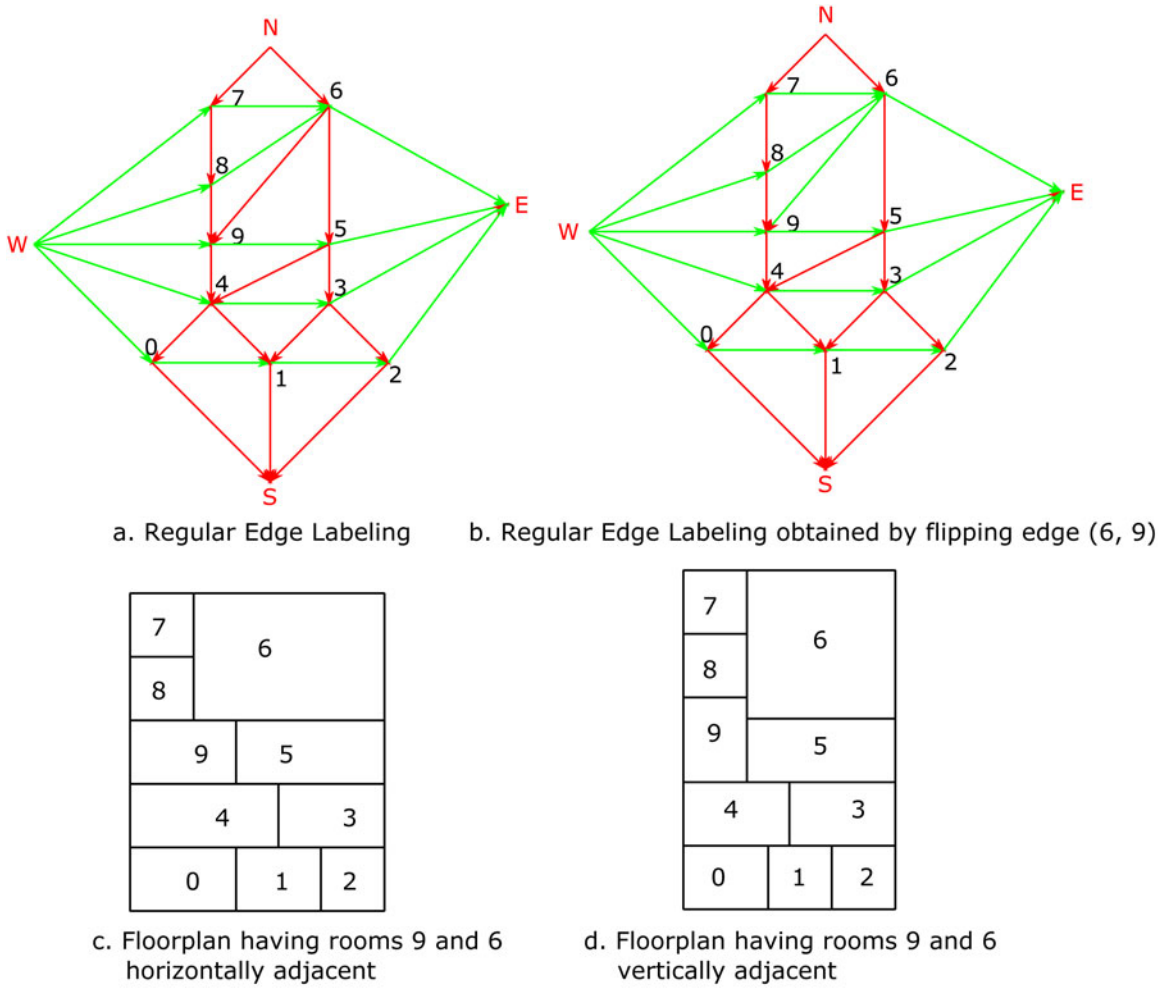
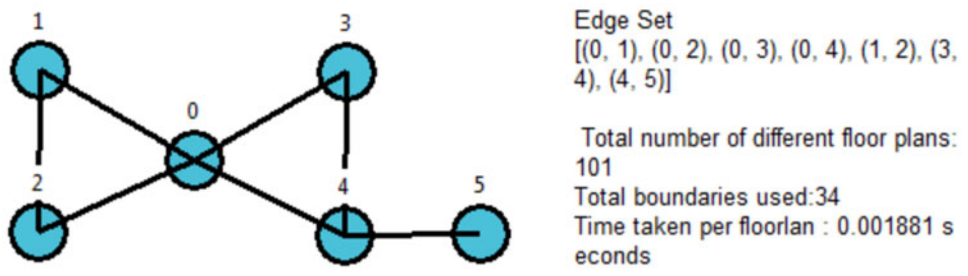

N
W
E
S
0
1
2
3
4
5
6
7
8
9
a. Regular Edge Labeling
b. Regular Edge Labeling obtained by flipping edge (6, 9)
c. Floorplan having rooms 9 and 6 horizontally adjacent
d. Floorplan having rooms 9 and 6 vertically adjacent

Fig. 13. Flippable edge and REL.



Edge Set
[(0, 1), (0, 2), (0, 3), (0, 4), (1, 2), (3, 4), (4, 5)]

Total number of different floor plans: 101
Total boundaries used:34
Time taken per floorlan : 0.001881 seconds

Fig. 14. Computing the number of different floorplans corresponding to a given graph.

graph in Figure 16a, GPLAN can produce 101 layouts but for the given boundary and dimensional constraints in Figure 18, there exist only 40 floorplans, a few of them are shown in Figure 18.

Spanning circulations

A circulation refers to the way people move through and interact within a building. A spanning circulation space stands for a single interior courtyard adjacent to each of the rooms of a floorplan.

In this section, we first present an algorithm for enumerating spanning circulation inside a floorplan assuming that floorplans have only one entrance.

Let f be the number of interior faces in $G(V, E)$ (dual of the floorplan), represented as $F_1, F_2, \dots, F_f$. Also, consider $E = \{e_1, e_2, \dots, e_m\}$ and $V = \{v_1, v_2, \dots, v_n\}$ are the edge set and vertex set of G, respectively. The steps of the algorithm for generating circulations are as follows (for understanding the steps of the algorithm, refer to Figure 19a,b, where a floorplan and corresponding dual graph are illustrated):

1. Choose a face, say F_1, of G having an exterior edge, say e_1 (edges adjacent to the exterior are called exterior edges).

   For example, corresponding to Figure 19b, we choose face F_1 consisting of vertices v_1, v_2, v_7.

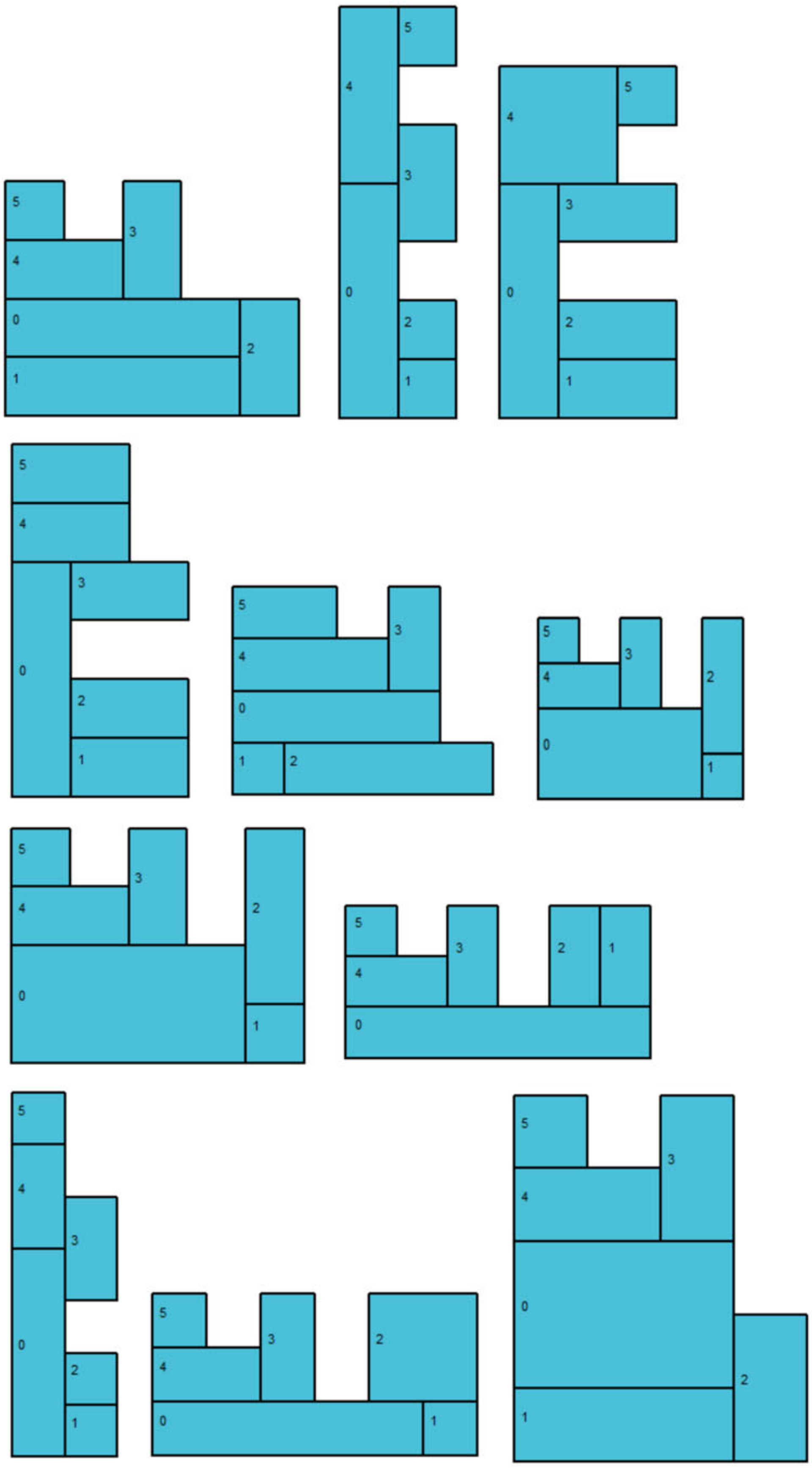
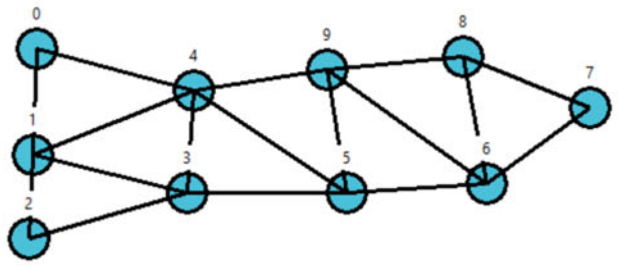
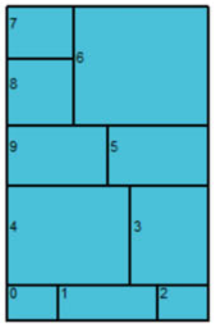


Fig. 15. Multiple floorplans.



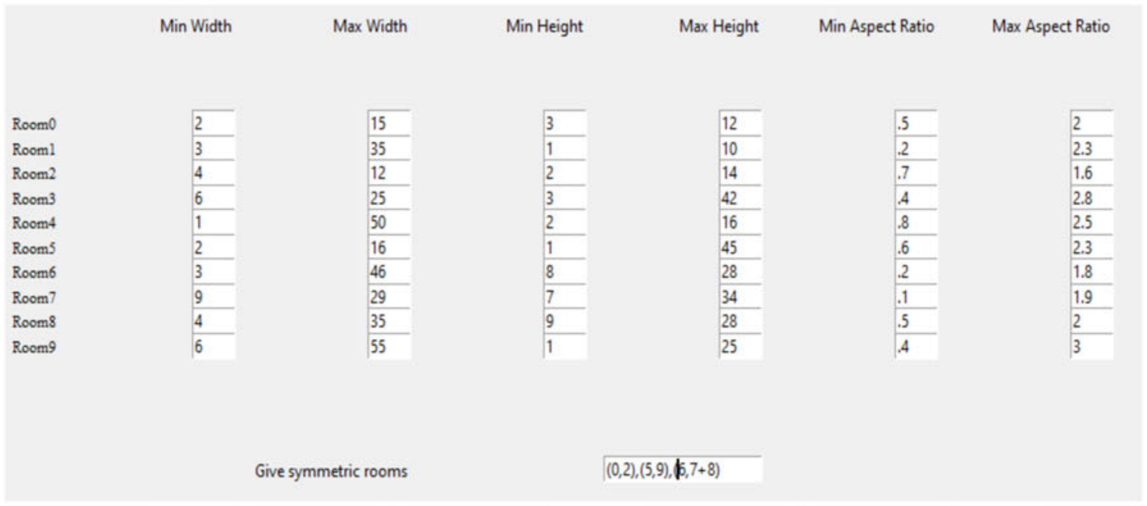
a. Given Graph



Dimensions of Each Room

	Height		Width
Room 0 :	4.75	:	6.79
Room 1 :	4.75	:	13.43
Room 2 :	4.75	:	6.79
Room 3 :	13.3	:	10.38
Room 4 :	13.3	:	16.62
Room 5 :	8.1	:	13.5
Room 6 :	16.0	:	18.0
Room 7 :	7.0	:	9.0
Room 8 :	9.0	:	9.0
Room 9 :	8.1	:	13.5

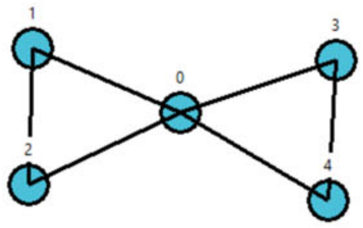
c. Dimensioned floorplan



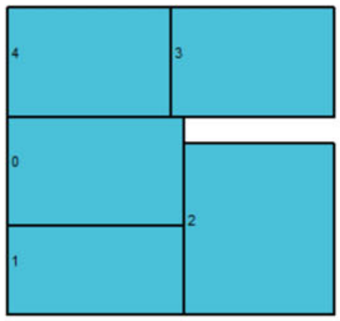
	Min Width	Max Width	Min Height	Max Height	Min Aspect Ratio	Max Aspect Ratio
Room0	2	15	3	12	.5	2
Room1	3	35	1	10	.2	2.3
Room2	4	12	2	14	.7	1.6
Room3	6	25	3	42	.4	2.8
Room4	1	50	2	16	.8	2.5
Room5	2	16	1	45	.6	2.3
Room6	3	46	8	28	.2	1.8
Room7	9	29	7	34	.1	1.9
Room8	4	35	9	28	.5	2
Room9	6	55	1	25	.4	3

Give symmetric rooms (0,2),(5,9),(6,7+8)

b. Dimensional Constraints



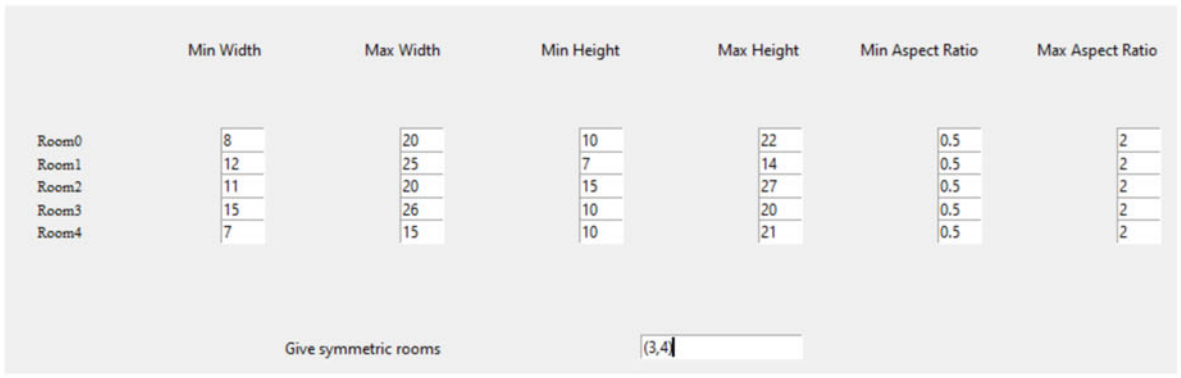
d. Given Graph



Dimensions of Each Room

	Height		Width
Room 0 :	10.0	:	16.22
Room 1 :	8.11	:	16.22
Room 2 :	15.63	:	13.78
Room 3 :	10.0	:	15.0
Room 4 :	10.0	:	15.0

e. Dimensioned floorplan

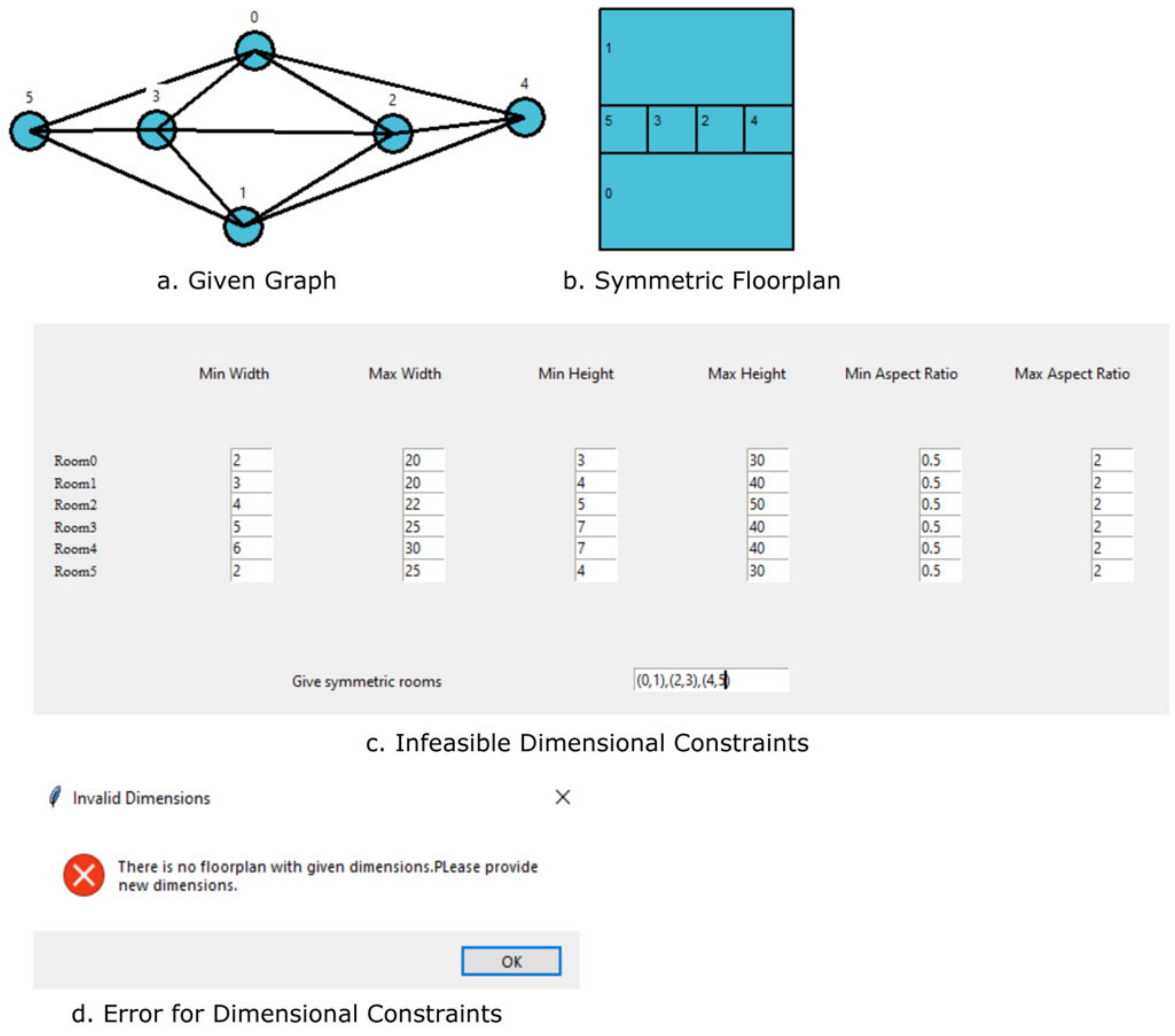


	Min Width	Max Width	Min Height	Max Height	Min Aspect Ratio	Max Aspect Ratio
Room0	8	20	10	22	0.5	2
Room1	12	25	7	14	0.5	2
Room2	11	20	15	27	0.5	2
Room3	15	26	10	20	0.5	2
Room4	7	15	10	21	0.5	2

Give symmetric rooms (3,4)

f. Dimensional Constraints

Fig. 16. Dimensioned floorplans for given graphs.


0
5
3
2
4
1
a. Given Graph
1
5
3
2
4
0
b. Symmetric Floorplan
c. Infeasible Dimensional Constraints
d. Error for Dimensional Constraints

	Min Width	Max Width	Min Height	Max Height	Min Aspect Ratio	Max Aspect Ratio
Room0	2	20	3	30	0.5	2
Room1	3	20	4	40	0.5	2
Room2	4	22	5	50	0.5	2
Room3	5	25	7	40	0.5	2
Room4	6	30	7	40	0.5	2
Room5	2	25	4	30	0.5	2

Give symmetric rooms (0,1),(2,3),(4,5)

Invalid Dimensions

There is no floorplan with given dimensions.PLease provide new dimensions.

OK

Fig. 17. A symmetric floorplan for a given graph.

Table 1. Constraints corresponding to vertical st-graph in Figure 4e and dimensional constraints in Figure 16b

Room	Equality constraints	Inequality constraints	Symmetry constraints
0	$w_{4,0} = w_{0,S}$	$2 \leq w_{4,0} \leq 15$	
1	$w_{4,1} + w_{3,1} = w_{1,S}$	$3 \leq w_{4,1} + w_{3,1} \leq 35$	
2	$w_{3,2} = w_{2,S}$	$4 \leq w_{3,2} \leq 12$	
3	$w_{5,3} = w_{3,1} + w_{3,2}$	$6 \leq w_{5,3} \leq 25$	
4	$w_{5,4} + w_{9,4} = w_{4,0} + w_{4,1}$	$1 \leq w_{5,4} + w_{9,4} \leq 50$	$w_{4,0} = w_{3,2}$
5	$w_{6,5} = w_{5,4} + w_{5,3}$	$2 \leq w_{6,5} \leq 16$	$w_{6,5} = w_{6,9} + w_{8,9}$
6	$w_{N,6} = w_{6,9} + w_{6,5}$	$3 \leq w_{N,6} \leq 46$	
7	$w_{N,7} = w_{7,8}$	$9 \leq w_{N,7} \leq 29$	
8	$w_{7,8} = w_{8,9}$	$4 \leq w_{7,8} \leq 35$	
9	$w_{6,9} + w_{8,9} = w_{9,4}$	$6 \leq w_{6,9} + w_{8,9} \leq 55$	

Table 2. Constraints corresponding to horizontal st-graph in Figure 4d and dimensional constraints in Figure 16b

Room	Equality constraints	Inequality constraints	Symmetry constraints
0	$h_{W,0} = h_{0,1}$	$3 \leq h_{W,0} \leq 12$	
1	$h_{0,1} = h_{1,2}$	$2 \leq h_{0,1} \leq 10$	
2	$h_{1,2} = h_{2,E}$	$2 \leq h_{1,2} \leq 14$	
3	$h_{4,3} = h_{3,E}$	$3 \leq h_{4,3} \leq 42$	$h_{W,0} = h_{1,2}$
4	$h_{W,4} = h_{4,3}$	$2 \leq h_{W,4} \leq 16$	$h_{9,5} = h_{W,9}$
5	$h_{9,5} = h_{5,E}$	$1 \leq h_{9,5} \leq 45$	$h_{7,6} + h_{8,6} = h_{W,7} + h_{W,8}$
6	$h_{7,6} + h_{8,6} = h_{6,E}$	$8 \leq h_{7,6} + h_{8,6} \leq 28$	
7	$h_{W,7} = h_{7,6}$	$7 \leq h_{W,7} \leq 34$	
8	$h_{W,8} = h_{8,6}$	$9 \leq h_{W,8} \leq 28$	
9	$h_{W,9} = h_{9,5}$	$1 \leq h_{W,9} \leq 25$	

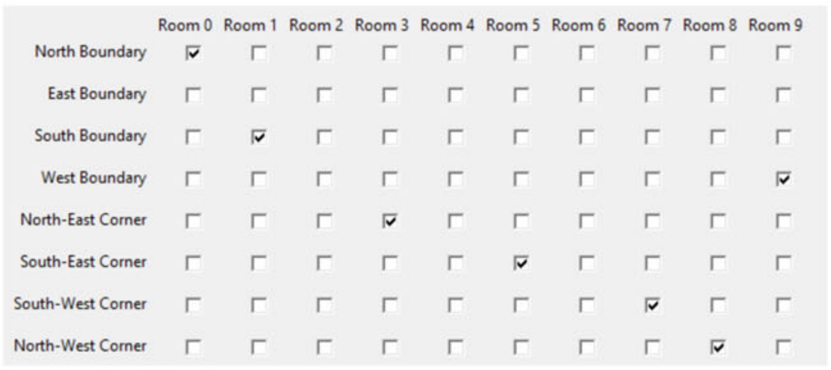
Table 3. Aspect ratio constraints corresponding to dimensional constraints in Figure 16b

Room	Intermediate width	Minimum AR	Minimum AR height	Maximum AR	Maximum AR height
0	6.79	0.5	3.4	2	13.58
1	13.43	0.2	2.69	2.3	30.89
2	6.79	0.7	4.75	1.6	10.86
3	10.38	0.4	4.15	2.8	29.06
4	16.62	0.8	13.3	2.5	41.56
5	13.5	0.6	8.1	2.3	31.05
6	18.0	0.2	3.6	1.8	32.4
7	9.0	0.1	0.9	1.9	17.1
8	9.0	0.5	4.5	2	18.0
9	13.5	0.4	5.4	3	40.5

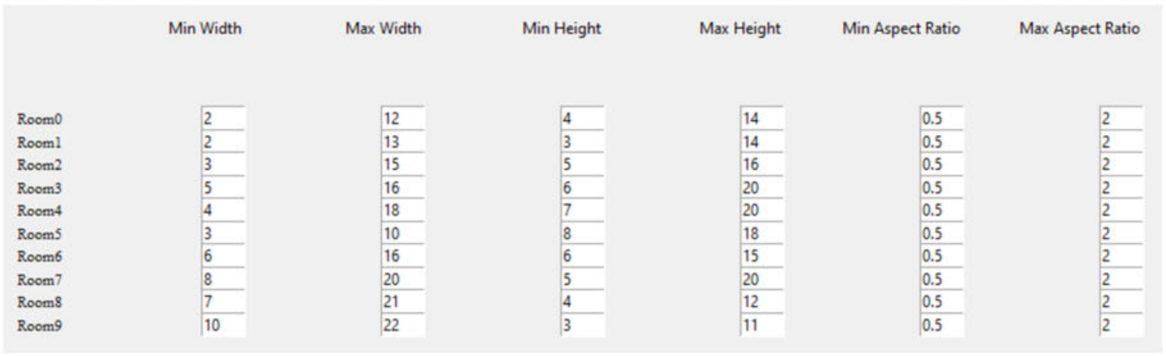
Table 4. Modified height constraints due to aspect ratio constraints corresponding to dimensional constraints in Figure 16b

Room	Equality constraints	Inequality constraints	Symmetry constraints
0	$h_{W,0} = h_{0,1}$	$3.4 \leq h_{W,0} \leq 12$	
1	$h_{0,1} = h_{1,2}$	$2.69 \leq h_{0,1} \leq 10$	
2	$h_{1,2} = h_{2,E}$	$4.75 \leq h_{1,2} \leq 10.86$	
3	$h_{4,3} = h_{3,E}$	$4.15 \leq h_{4,3} \leq 29.06$	$h_{W,0} = h_{1,2}$
4	$h_{W,4} = h_{4,3}$	$13.3 \leq h_{W,4} \leq 16$	$h_{9,5} = h_{W,9}$
5	$h_{9,5} = h_{5,E}$	$8.1 \leq h_{9,5} \leq 31.05$	$h_{7,6} + h_{8,6} = h_{W,7} + h_{W,8}$
6	$h_{7,6} + h_{8,6} = h_{6,E}$	$8 \leq h_{7,6} + h_{8,6} \leq 28$	
7	$h_{W,7} = h_{7,6}$	$7 \leq h_{W,7} \leq 17.1$	
8	$h_{W,8} = h_{8,6}$	$9 \leq h_{W,8} \leq 18.0$	
9	$h_{W,9} = h_{9,5}$	$5.4 \leq h_{W,9} \leq 25$	

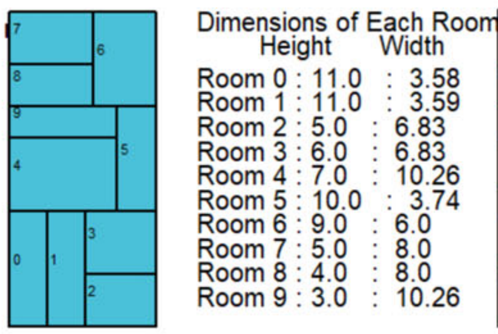
2. Consider an empty set S. Add all three vertices that form F_1 to S. For example, corresponding to Figure 19c, $S = \{v_1, v_2, v_7\}$.
3. Insert a new vertex, say V_{n+1}, on the exterior edge e_1. By inserting a new edge say E_k, make V_{n+1} adjacent to a vertex of F_1 that is not an end-point of e_1. For example, V_8 and E_{15} are shown in Figure 19c (in Fig. 19, inserted vertices are marked by red color while inserted edges are illustrated by dotted lines).
4. Insert a rectangular circulation space, say C_{n+1}, corresponding to V_{n+1} in the corresponding floorplan. For example, in Figure 19d, inserted circulation space C_8 is adjacent to rooms R_1, R_2, and R_7. The conditions for inserting a



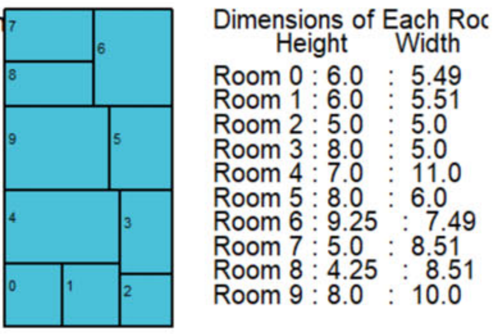
	Room 0	Room 1	Room 2	Room 3	Room 4	Room 5	Room 6	Room 7	Room 8	Room 9
North Boundary	✓	☐	☐	☐	☐	☐	☐	☐	☐	☐
East Boundary	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
South Boundary	☐	✓	☐	☐	☐	☐	☐	☐	☐	☐
West Boundary	☐	☐	☐	☐	☐	☐	☐	☐	☐	✓
North-East Corner	☐	☐	☐	✓	☐	☐	☐	☐	☐	☐
South-East Corner	☐	☐	☐	☐	☐	✓	☐	☐	☐	☐
South-West Corner	☐	☐	☐	☐	☐	☐	☐	✓	☐	☐
North-West Corner	☐	☐	☐	☐	☐	☐	☐	☐	✓	☐



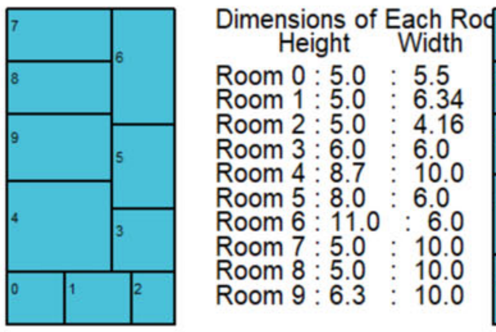
	Min Width	Max Width	Min Height	Max Height	Min Aspect Ratio	Max Aspect Ratio
Room0	2	12	4	14	0.5	2
Room1	2	13	3	14	0.5	2
Room2	3	15	5	16	0.5	2
Room3	5	16	6	20	0.5	2
Room4	4	18	7	20	0.5	2
Room5	3	10	8	18	0.5	2
Room6	6	16	6	15	0.5	2
Room7	8	20	5	20	0.5	2
Room8	7	21	4	12	0.5	2
Room9	10	22	3	11	0.5	2



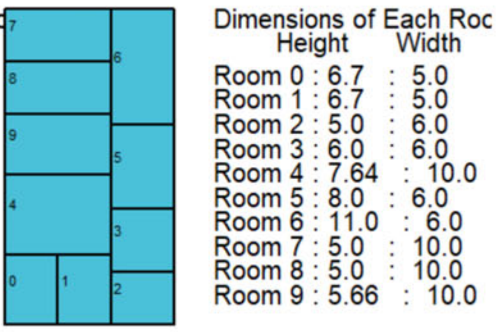
```
Dimensions of Each Room
    Height    Width
Room 0 : 11.0  :  3.58
Room 1 : 11.0  :  3.59
Room 2 : 5.0  :  6.83
Room 3 : 6.0  :  6.83
Room 4 : 7.0  :  10.26
Room 5 : 10.0  :  3.74
Room 6 : 9.0  :  6.0
Room 7 : 5.0  :  8.0
Room 8 : 4.0  :  8.0
Room 9 : 3.0  :  10.26
```



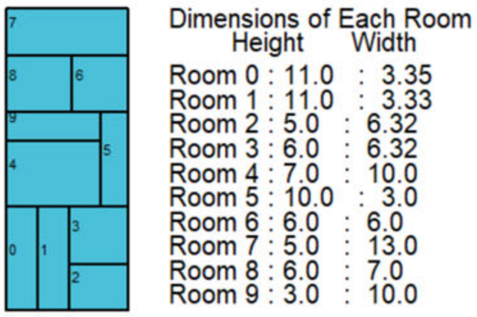
```
Dimensions of Each Roc
    Height    Width
Room 0 : 6.0  :  5.49
Room 1 : 6.0  :  5.51
Room 2 : 5.0  :  5.0
Room 3 : 8.0  :  5.0
Room 4 : 7.0  :  11.0
Room 5 : 8.0  :  6.0
Room 6 : 9.25  :  7.49
Room 7 : 5.0  :  8.51
Room 8 : 4.25  :  8.51
Room 9 : 8.0  :  10.0
```



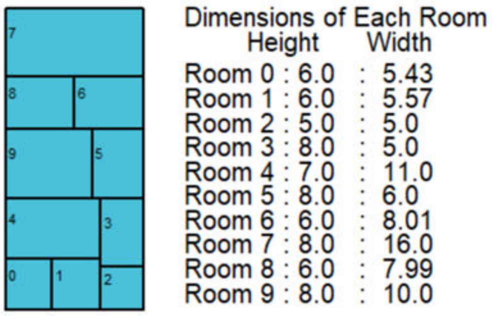
```
Dimensions of Each Roc
    Height    Width
Room 0 : 5.0  :  5.5
Room 1 : 5.0  :  6.34
Room 2 : 5.0  :  4.16
Room 3 : 6.0  :  6.0
Room 4 : 8.7  :  10.0
Room 5 : 8.0  :  6.0
Room 6 : 11.0  :  6.0
Room 7 : 5.0  :  10.0
Room 8 : 5.0  :  10.0
Room 9 : 6.3  :  10.0
```



```
Dimensions of Each Roc
    Height    Width
Room 0 : 6.7  :  5.0
Room 1 : 6.7  :  5.0
Room 2 : 5.0  :  6.0
Room 3 : 6.0  :  6.0
Room 4 : 7.64  :  10.0
Room 5 : 8.0  :  6.0
Room 6 : 11.0  :  6.0
Room 7 : 5.0  :  10.0
Room 8 : 5.0  :  10.0
Room 9 : 5.66  :  10.0
```



```
Dimensions of Each Room
    Height    Width
Room 0 : 11.0  :  3.35
Room 1 : 11.0  :  3.33
Room 2 : 5.0  :  6.32
Room 3 : 6.0  :  6.32
Room 4 : 7.0  :  10.0
Room 5 : 10.0  :  3.0
Room 6 : 6.0  :  6.0
Room 7 : 5.0  :  13.0
Room 8 : 6.0  :  7.0
Room 9 : 3.0  :  10.0
```



```
Dimensions of Each Room
    Height    Width
Room 0 : 6.0  :  5.43
Room 1 : 6.0  :  5.57
Room 2 : 5.0  :  5.0
Room 3 : 8.0  :  5.0
Room 4 : 7.0  :  11.0
Room 5 : 8.0  :  6.0
Room 6 : 6.0  :  8.01
Room 7 : 8.0  :  16.0
Room 8 : 6.0  :  7.99
Room 9 : 8.0  :  10.0
```

Fig. 18. Dimensioned floorplans with boundary constraints.

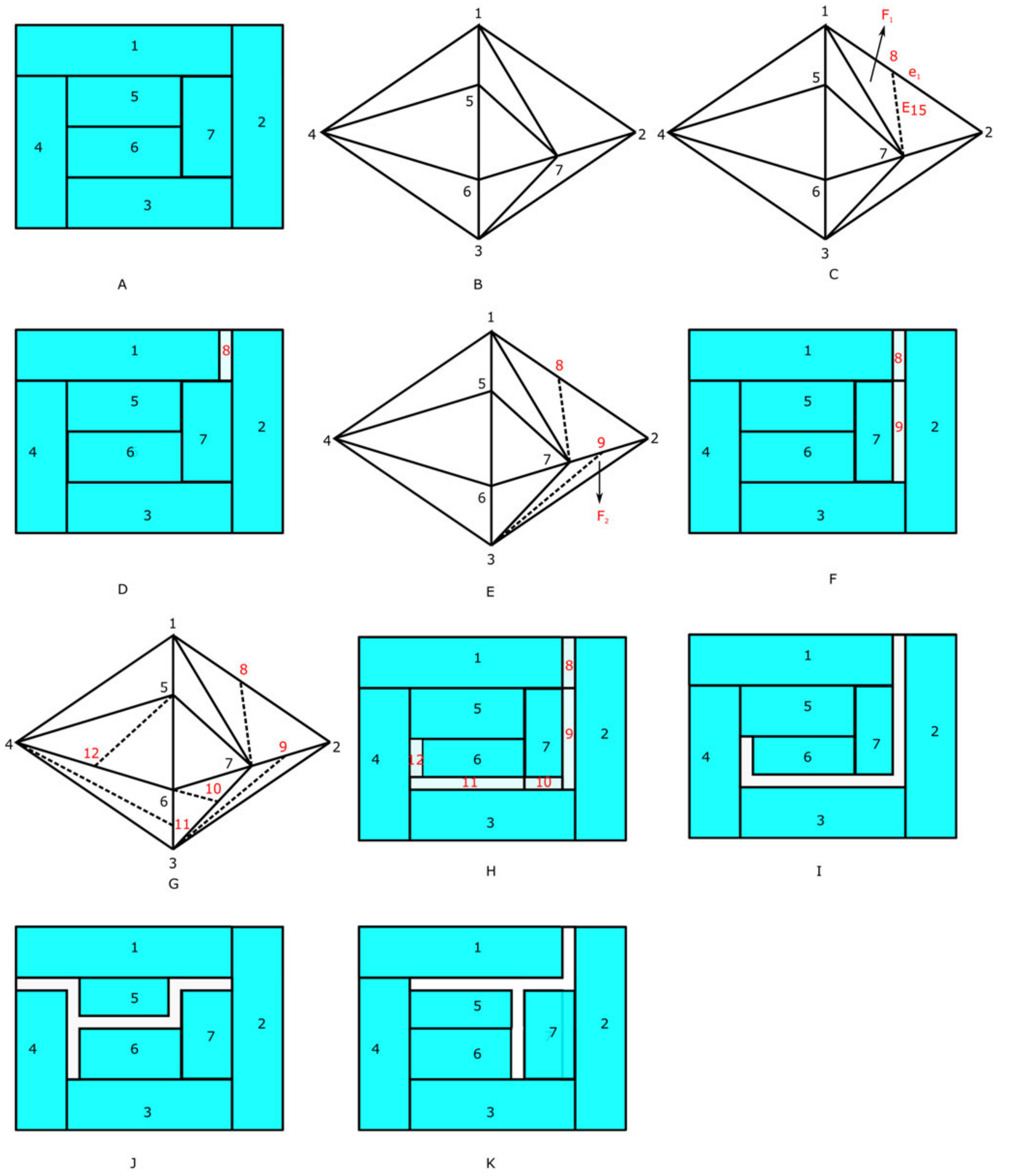
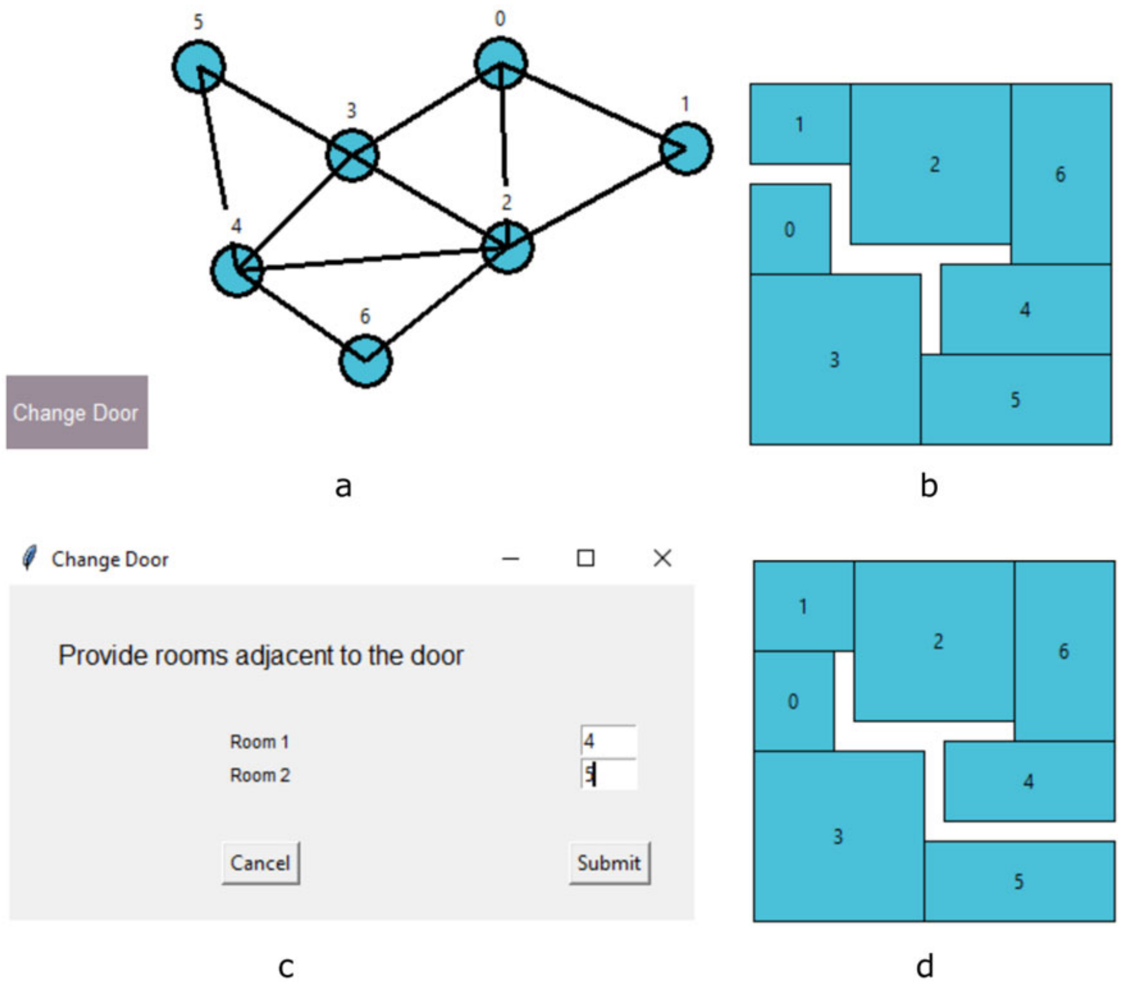


Fig. 19. Inserting circulation within a floorplan.

circulation space are given in Step 9 (the inserted circulation spaces are demonstrated by gray color).

5. Choose a face, say F_i, $2 \le i \le f$, to which an edge has not been inserted such that F_i is adjacent to at least one of the faces to which an edge has already been inserted. Let F_i be adjacent to F_j, $1 \le j \le f$. For example, in Figure 19e, we choose a face consisting of vertices v_2, v_7, v_3 and is adjacent to F_1.
6. Pick an edge e_j that belongs to both F_i and F_j. Insert a new vertex, say V_k, $k \ge n+2$, on e_j and make V_k adjacent to a



Change Door

Provide rooms adjacent to the door

Room 1	4
Room 2	5

Cancel Submit

Fig. 20. A floorplan with spanning circulations.

vertex of F_i that is not an end-point of e_j. For example, in Figure 19e, V_9 and E_{16} have been inserted.

7. Add all three vertices that form F_i to S. Corresponding to Figure 8e, $S = \{v_1, v_2, v_7, v_3\}$.
8. Insert a circulation space, say C_k, corresponding to V_k in the corresponding floorplan such that it preserves the adjacency relations of V_k, that is, it only satisfies the following adjacency requirements:
   (a) C_k is adjacent to rooms corresponding to the vertices that are adjacent to V_k (in Fig. 19f, C_9 is adjacent to R_2, R_3, and R_7),
   (b) C_k must be adjacent to one of the existing circulation space (in Fig. 19f, C_9 is adjacent to C_8),
   (c) Let the endpoints of e_j be v_j^1 and v_j^2. C_k is inserted in such a way that it makes the rooms corresponding to v_j^1 and v_j^2 non-adjacent (in Fig. 19f, after inserting C_9, rooms R_2 and R_7 are non-adjacent),
   (d) to satisfy adjacency requirements, C_k can be inserted horizontally or vertically. For example, in Figure 19f, C_9 has been inserted vertically.
9. If $S \subset V$, go to Step 6 otherwise stop. For example, corresponding to Figure 19b, a floorplan with all inserted circulation spaces is shown in Figure 19h.
10. Adjoin all inserted circulation spaces to form a spanning circulation. For example, a floorplan with spanning circulation is shown in Figure 19i. Similarly, floorplans with different positions of spanning circulations are illustrated in Figure 19j,k.

Floorplans with circulation produced by GPLAN are shown in Figure 20, where by default the entrance is between rooms 0 and 1 (Fig. 20b), but a user has a choice to change the entrance (Fig. 20c) to have a new circulation (Fig. 20d). Once we have a spanning circulation, in the future, we would like to provide an option to the user to add and remove parts of circulations (in Fig. 19h, among circulation 8, 9, 10, 11, and 17, the user would be able to delete any of them).

Conclusion and future work

In this work, we have proposed a graph-theoretic approach with a solid mathematical foundation which can handle many constraints simultaneously and able to generate a large amount of layouts for given adjacencies and dimensions. Also, along with adjacencies and dimensions, we have considered boundary and symmetric constraints, which help us to customize floorplans based on users requirements. At this stage, this work, in general, proposes floorplan construction, which has applications in housing design (Hu *et al.*, 2020), game layout design (Ma *et al.*, 2014), VLSI design (Kahng *et al.*, 2011), etc. In future, we aim to customize this work for housing design where the user only needs to present his requirements in terms of spaces and GPLAN would automatically provide a set of layouts. Also, to automate the process, one of the approaches is to derive data, mainly adjacency and dimensions, from the existing floorplans and use these data to produce multiple layouts (Wu *et al.*, 2019), which we aim to do in future. It can also be noticed that a very small amount of work has been done where boundary layout has been considered as input (Laignel *et al.*, 2021). In future, we aim to develop graph-based algorithms for generating floorplans with given boundary layouts.

We agree that architects can never be replaced by a software, at the same time, it may not be feasible for them to propose multiple drawings to the user. Here, we aim to assist architects by providing them the initial layouts. Also, in developing countries, architects are out of reach to a large amount of population and therefore most of the houses are built without getting any assistance from the architects. Hence, there is a need for such a software that can generate layouts for the users which are architecturally feasible (if not optimal and stylish). In future, we would try to customize GPLAN so that it can address the need for housing designs for the required population.

Supplementary material. The supplementary material for this article can be found at https://doi.org/10.1017/S0890060421000275

Conflict of interest. The authors report no conflicts of interest.

References

Alam MJ, Biedl T, Felsner S, Kaufmann M, Kobourov S and Ueckerdt T (2013) Computing cartograms with optimal complexity. *Discrete & Computational Geometry* **50**, 784–810. doi:10.1007/s00454-013-9521-1

Baybars I (1982) The generation of floor plans with circulation spaces. *Environment and Planning B* **9**, 445–456. doi:10.1068/b090445

Berry A, Blair JRS, Heggernes P and Peyton BW (2004) Maximum cardinality search for computing minimal triangulations of graphs. *Algorithmica* **39**, 287–298.

Bhasker J and Sahni S (1987) A linear time algorithm to check for the existence of a rectangular dual of a planar triangulated graph. *Networks* **17**, 307–317. doi:10.1002/net.3230170306

Bhasker J and Sahni S (1988) A linear algorithm to find a rectangular dual of a planar triangulated graph. *Algorithmica* **3**, 247–278. doi:10.1007/BF01762117

Duarte JP (2005) A discursive grammar for customizing mass housing: the case of siza's houses at malagueira. *Automation in Construction* **14**, 265–275. doi:10.1016/j.autcon.2004.07.013

Egor G, Sven S, Martin D and Reinhard K (2020) Computer-aided approach to public buildings floor plan generation. Magnetizing floor plan generator. *Procedia Manufacturing* **44**, 132–139. doi:10.1016/j.promfg.2020.02.214

Eppstein D, Mumford E, Speckmann B and Verbeek K (2012) Area-universal and constrained rectangular layouts. *SIAM Journal on Computing* **41**, 537–564. doi:10.1137/110834032

Hu R, Huang Z, Tang Y, Matias O, Kaick V, Zhang H and Huang H (2020) Graph2plan: learning floorplan generation from layout graphs. *ACM Transactions on Graphics* **39**, 1–14. doi:10.1145/3386569.3392391

Kahng AB, Lienig J, Markov IL and Hu J (2011) *VLSI Physical Design: From Graph Partitioning to Timing Closure*. Dordrecht: Springer Science & Business Media.

Kant G and He X (1993) Two algorithms for finding rectangular duals of planar graphs. In van Leeuwen J (ed.), *Graph-Theoretic Concepts in Computer Science. WG 1993. Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer, pp. 396–410. doi:10.1007/3-540-57899-4_69

Koźmiński K and Kinnen E (1985) Rectangular duals of planar graphs. *Networks* **15**, 145–157. doi:10.1002/net.3230150202

Laignel G, Pozin N, Geffrier X, Delevaux L, Brun F and Dolla B (2021) Floor plan generation through a mixed constraint programming-genetic optimization approach. *Automation in Construction* **123**, 1–21. doi:10.1016/j.autcon.2020.103491

Levin PH (1964) Use of graphs to decide the optimum layout of buildings. *The Architects' Journal* **7**, 809–815.

Liao CC, Lu HI and Yen HC (2003) Compact floor-planning via orderly spanning trees. *Journal of Algorithms* **48**, 441–451. 10.1016/S0196-6774(03)00057-9

Ma C, Vining N, Lefebvre S and Sheffer A (2014) Game level layout from design specification. *EUROGRAPHICS* **33**, 95–104. doi.org/10.1111/cgf.12314

Marson F and Musse SR (2010) Automatic real-time generation of floor plans based on squarified treemaps algorithm. *International Journal of Computer Games Technology* **2010**. doi:10.1155/2010/624817

Merrell P, Schkufza E and Koltun V (2010) Computer-generated residential building layouts. *ACM Transactions on Graphics* **29**, 1–12. doi:10.1145/1882261.1866203

Mitchel WJ (1990) *The Logic of Architecture (Design Computation and Cognition)*. Cambridge, MA: MIT Press.

Mitchell WJ, Steadman P and Liggett RS (1976) Synthesis and optimization of small rectangular floor plans. *Environment and Planning B: Planning and Design* **3**, 37–70. doi:10.1068/b030037

Müller P, Zeng G, Wonka P and Van GL (2007) Image-based procedural modeling of facades. *ACM Transactions on Graphics* **26**, 1–9. doi:10.1145/1276377.1276484

Nisztuk M and Myszkowski PB (2019) Hybrid evolutionary algorithm applied to automated floor plan generation. *International Journal of Architectural Computing* **17**, 260–283. doi:10.1177/1478077119832982

Rinsma I (1987) Nonexistence of a certain rectangular floorplan with specified areas and adjacency. *Environment and Planning B* **14**, 163–166. doi:10.1068/b140163

Rodrigues E, Gaspar AR and Gomes Á (2013) An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture, part 2: Validation and performance tests. *Computer-Aided Design* **45**, 898–910. doi:10.1016/j.cad.2013.01.003

Roth J, Hashimshony R and Wachman A (1982) Turning a graph into a rectangular floor plan. *Building and Environment* **17**, 163–173. doi:10.1016/0360-1323(82)90037-3

Shekhawat K (2018) Enumerating generic rectangular floor plans. *Automation in Construction* **92**, 151–165. doi:10.1016/j.autcon.2018.03.037

Shekhawat K and Duarte JP (2018) Introduction to generic rectangular floor plans. *AIEDAM: Artificial Intelligence for Engineering Design, Analysis and Manufacturing* **32**, 331–350. doi.org/10.1017/S0890060417000671

Shekhawat K, Upasani N, Bisht S and Jain RN (2021) A tool for computer-generated dimensioned floorplans based on given adjacencies. *Automation in Construction* **127**, 1–22. doi.org/10.1016/j.autcon.2021.103718

Shi F, Soman RK, Han J and Whyte JK (2020) Addressing adjacency constraints in rectangular floor plans using Monte-Carlo tree search. *Automation in Construction* **115**, 1–14. doi:10.1016/j.autcon.2020.103187

Steadman P (1973) Graph theoretic representation of architectural arrangement. *Architectural Research and Teaching* **2**, 161–172.

Upasani N, Shekhawat K and Sachdeva G (2020) Automated generation of dimensioned rectangular floorplans. *Automation in Construction* **113**, 1–11. doi:10.1016/j.autcon.2020.103149

Wang X-Y and Zhang K (2020) Generating layout designs from high-level specifications. *Automation in Construction* **119**, 1–12. doi:10.1016/j.autcon.2020.103288

Wang X-Y, Yang Y and Zhang K (2018) Customization and generation of floor plans based on graph transformations. *Automation in Construction* **94**, 405–416. doi:10.1016/j.autcon.2018.07.017

Wu F, Yan D-M, Dong W, Zhang X and Wonka P (2014) Inverse procedural modeling of facade layouts. *ACM Transactions on Graphics* **33**, 1–10. doi:10.1145/2601097.2601162

Wu W, Fan L, Liu L and Wonka P (2018) Miqp-based layout design for building interiors. *Computer Graphics Forum* **37**, 511–521. doi:10.1111/cgf.13380

Wu W, Fu X-M, Tang R, Wang Y, Qi Y-H and Liu L (2019) Data-driven interior plan generation for residential buildings. *ACM Transactions on Graphics* **38**, 1–12. doi.org/10.1145/3355089.3356556

Yeap KH and Sarrafzadeh M (1993) Floor-planning by graph dualization: 2-concave rectilinear modules. *SIAM Journal on Computing* **22**, 500–526. doi:10.1137/0222035

Krishnendra Shekhawat is presently working as an Associate Professor at Department of Mathematics, BITS Pilani, India. He joined BITS Pilani as an Assistant Professor in 2016. Prior to this joining, he worked as a post-doctoral fellow in the guidance of Prof. Jose P. Duarte at University of Lisbon. He has completed his PhD in Mathematics from University of Geneva in the guidance of Prof. Daniel Coray. He is working in the field of Geometric Graph Theory and Architectural Design.

Rahil N. Jain is about to complete an Integrated degree of M.Sc. (Hons.) Mathematics and B.E. (Hons.) Computer Science from BITS Pilani, India. He has a publication in the journal *Automation in Construction*. He has a keen interest in Graph Theory and Machine Learning.

Sumit Bisht is a senior at BITS Pilani, India where he is majoring in Mathematics and Computer Science. He has been working on solving architectural design problems using graph theory for the last 3 years under the guidance of Prof. Krishnendra Shekhawat and has co-authored a paper in the same. His research interests lie in Computational Graph Theory and Deep Learning.

Aishwarya Kondaveeti is currently pursuing a dual degree program in M.Sc. Mathematics and B.E Computer Science at BITS Pilani, India. Her research interests lie in Computational Graph Theory and Optimization.

Dipam Goswami is a final year undergraduate student currently pursuing a dual degree, a BE in Computer Science, and an MSc in Mathematics from BITS Pilani, India. His research interests include Graph-theoretic Algorithms, Artificial Intelligence, Machine Learning, and Computer Vision. He has worked on developing graph-theoretic algorithms for generating rectangular floor plans.