CONSTRUCTION OF ARCHITECTURAL FLOOR PLANS FOR GIVEN ADJACENCY REQUIREMENTS

KRISHNENDRA SHEKHAWAT¹ and PINKI PINKI²
¹² Birla Institute of Technology & Science, Pilani-333031
¹ krishnendra.iitd@gmail.com ²p20170003@pilani.bits-pilani.ac.in

Abstract. For most of the architectural design problems, there are underlying mathematical sub-problems, they may require to consider for generating architectural layouts. One of these sub-problems is to satisfy adjacency constraints for obtaining an initial layout. But in the literature, there does not exist a mathematical procedure that can address any given adjacency requirements, i.e., there does not exist a tool for generating a floor plan corresponding to any given adjacency (planar) graph (there exist algorithms for constructing floor plans for planar triangulated graphs only). In this paper, we are going to present an algorithm that would generate a floor plan corresponding to any given planar graph. The larger aim of this research is to develop a user-friendly tool that can generate a variety of initial layouts corresponding to a given graph, which can be further modified by the architects/designers.

Keywords. Floor plan; Algorithm; Graph Theory.

1. Introduction

One of the primary tasks in most of the architectural design processes is to build a layout while satisfying the given adjacency constraints. It is easy to see that adjacency constraints need to be addressed using the mathematical tools only, in particular the graph theoretical tools. Even we can think of generating a layout only if we know there exists a layout for the given adjacency constraints (there may or may not exists a floor plan corresponding to a given adjacency graph). Hence, the generation of an initial layout can be seen as a difficult mathematical problem and if it is addressed, it can be a helpful tool for the architects.

In the literature, there exists a sufficient amount of work for the construction of floor plans based on graph theoretic approach. One of the initial works in this direction is given by Levin [8] in 1964. He gave an intuitive approach for mapping adjacency graphs to architectural floor plans. Then a lot of work has been proposed in this direction. This work can be categorized on the basis of rectangular floor plans and orthogonal floor plans. A brief overview of the construction of floor plans using graph theoretic approach is as follows:

In 1980, Baybars and Eastman [1] gave a systematic procedure for obtaining a floor plan for a given underlying maximal planar graph (MPG), where the boundary of the obtained floor plan is not necessarily rectangular (a graph is called maximal if no edges can be added to it without losing planarity). In 1982, Roth et al. [13] presented the construction of a dimensioned rectangular floor plan for a given planar triangulated graph (PTG). In 1985, Kozminski & Kinnen [7] developed the necessary and sufficient conditions for the existence of a rectangular floor plan for properly triangulated planar graphs (PTPG). Using the concepts of [7], in 1987, Bhasker & Sahni [2] gave a linear time algorithm for constructing a rectangular floor plan for bi-connected PTPG.

In 1990, Rinsma et al. [12] were the first ones to demonstrate the automated generation of an orthogonal floor plan corresponding to a given MPG. In 1993, Sun and Sarrafzadeh [18] found that there exist PTG for which rectangular floor plans do not exist, and for those PTG, there may exist a floor-plan having 0-CRM and 1-CRM (a \(k\)-concave rectilinear module, i.e., \(k\)-CRM is a module with \(k\) bends). In the same year, Yeap and Sarrafzadeh [21] showed that if only 0-CRM and 1-CRM are allowed, there are PTG’s that do not admit a floor-plan, i.e., 2-CRMs are sufficient and necessary for graph dualization floor-planning.

Using the concept of orderly spanning trees, in 2003, Liao et al. [9] gave a linear time algorithm for constructing an orthogonal floor plan for any \(n\)-vertex PTG which require fewer module types i.e., the algorithm uses only 1-modules, L-modules (could be flipped horizontally), and T -modules, but Z-modules are not required.

In 2009, Eppstein et al [3] proved that a rectangular floor plan is area universal if and only if it is one sided (a rectangular floor plan is area universal if any assignment of areas to rooms can be realized by a combinatorial equivalent rectangular floor plan).

In 2011, Jokar and Sangchooli [6] introduced the concept of area of a face of a graph and used it for the construction of orthogonal floor plans, where the given graphs are maximal planar.

In 2014, Shekhawat [14] gave the enumeration of best connected rectangular floor plans. In 2015, Shekhawat [15] gave an algorithm for constructing plus-shaped floor plans. In 2018, Wang et al. [20] developed a prototype to regenerate well-known existing rectangular floor plans having an underlying graph as bi-connected PTPG. In 2019, Nisztuk et al. [11] developed an application, based on the evolutionary algorithms, for the automated generation of floorplans satisfying the given adjacency and dimensional requirements. But the proposed work does not cover the graphs for which it is not possible to construct floor plans with rectangular rooms only.

It can be observed from above discussion that most of the work done related to the existence and construction of a floor plan is restricted to planar triangulated graphs only, i.e., to the best of our knowledge, there does not exist an algorithm for checking the existence of a floor plan and for its construction corresponding to non-triangulated graphs (in 2019, Shekhawat et al. [16] gave an algorithm for constructing a floor plan for any given graph but for the algorithm, it is required to
have all generic rectangular floor plans and enumeration of generic rectangular floor plans for a large amount of rooms is computationally very demanding. Hence, we present an algorithm for generating floor plans for non-triangulated graphs (which can obviously be used for triangulated graphs).

2. Methodology

In this section, we present an algorithm for constructing a floor plan for a given planar adjacency graph. For a better understanding of the algorithm, we require the following terminologies.

A planar graph that can be drawn in a plane is called a plane graph, which divides the plane into connected components called faces/regions. The unbounded region is called the external face. Except for the external face, all other faces are internal faces. A planar graph $G$ is said to be triangulated if all of its faces are triangular. If only interior faces are triangular, and the outer face is a cycle of length $k$, $k > 3$, then $G$ is called internally triangulated.

A floor plan corresponding to a graph $G$ is a closed polygon in which every vertex of $G$ is replaced by its component polygons called rooms. Two rooms in a floor plan are said to be adjacent if they share a wall or some part of it, where wall of a room refers to the edges forming its perimeter.

An ordering of a planar graph $G_n$ whose vertices are denoted and ordered as $v_1, v_2, \ldots, v_n$ is a canonical ordering of $G_n$ if the following conditions hold, when $1 < k < n$.

- $G_k$ is biconnected and internally triangulated.
- $\langle v_1, v_2 \rangle$ is an outer edge of $G_k$.
- Each vertex say $v_{k+1}$ where $k+1 \leq n$ must be adjacent to at least two vertices of $G_k$ such that it forms a cycle having at least one edge that lies on the exterior face of $G_{k+1}$.

For example, the canonical ordering of graph in Figure 3(b) is shown in Figure 3(c).

*Horizontal Adjacency:* Two rooms are horizontally adjacent if and only if there is a horizontal line segment connecting them (see Figure 1).

![Figure 1. Horizontal Adjacency.](image)

*Vertical Adjacency:* Two rooms are vertically adjacent if and only if there is a vertical line segment connecting them (see Figure 2).
A 2-visibility drawing of a plane graph is a drawing where all the vertices are drawn as rectangular boxes and an edge is drawn either as a horizontal line segment or as a vertical line segment [4]. For example, in Figure 4 the steps of 2-visibility drawing of the graph in Figure 3(c) are illustrated.

Figure 2. Vertical Adjacency.

Figure 3. Triangulation of a given graph and canonical ordering of the triangulated graph.
2.1. ALGORITHM FLOORPLAN

**Input:** A planar graph $G$ with its planar embedding.

**Output:** A Floor plan

**Procedure:**

1. **2-Visibility Drawing of the given graph $G$.**
   a) If $G$ is not internally triangulated, add a minimum number of edges so that $G$ would be internally triangulated.
   
   **Example:** The graph given in Figure 3(a) is not triangulated however after adding the red edges it is transformed into a triangulated graph as shown in Figure 3(b).
   
   b) Compute the canonical ordering of $G$, denoted as $v_1, v_2, \ldots, v_n$.
      
      i) Choose any edge on the outer face of $G_n$ and mark its endpoints as $v_1$ and $v_2$. Let the graph having only adjacent vertices $v_1$ and $v_2$ be called as $G_2$.
      
      ii) For $k=2, n-1$ do
      
      iii) Choose $v_{k+1}$ where $k+1 \leq n$ in such a way it must be adjacent to at least two vertices of $G_k$.
      
      iv) Add $v_{k+1}$ to $G_k$ such that it forms a cycle having at least one edge that lies on the exterior face of $G_{k+1}$.
      
      **Example:** Start with any edge on the outer face of $G$ and mark its endpoints as $v_1$ and $v_2$. Let this graph be $G_2$. For $k=2, k+1=3$ and $3 \leq n$. Hence, $v_3$ must be located on the outer cycle of $G_3$ and $G_3$ is biconnected. Similarly, we obtain the canonical ordering of $G$, as shown in Figure 3(c).
      
   c) Represent each vertex $v_i$ of $G$ by a rectangular box $R_i$, where $1 \leq i \leq n$.
   
   d) Place $R_1$ and $R_2$ of size $1 \times 2$ and $1 \times 1$ such that they must be horizontally adjacent as shown in Figure 4(a).
   
   e) For $i=3, n$ do
   
   f) Consider every $R_i$ of size $1 \times 1$.
   
   g) Let $R_{p_1}, R_{p_2}, \ldots, R_{p_j}$ be the neighbors of $R_i$ that are present in $G_{i-1}$ and
are arranged in counterclockwise order where \( 1 < j < n \).

**Example:** In Figure 4(ii), for \( i = 3 \), \( R_3 \) is of size 1 x 1 and neighbors are \( R_1 \) and \( R_2 \). In Figure 4(iv), the neighbors of \( R_4 \) in counterclockwise order are \( R_1, R_3 \), and \( R_2 \).

**h)** Place \( R_i \) above or to the right of its neighbors given in the drawing of \( G_{i-1} \) such that \( R_{p_1}, R_{p_2}, \ldots, R_{p_j-1} \) is horizontally adjacent to \( R_i \) and \( R_{p_j} \) is vertically adjacent to \( R_i \).

**Example:** In Figure 4(iii), \( R_3 \) is drawn above \( R_2 \) and to the right of \( R_1 \) in such a way that it is horizontally adjacent to \( R_1 \) and vertically adjacent to \( R_2 \). In Figure 4(iv), \( R_4 \) is placed to the right of \( R_2 \) and \( R_3 \). Here, \( R_4 \) is horizontally adjacent to \( R_2 \) but not to \( R_3 \).

**i)** Stretch each of the drawn rectangular boxes \( R_j \) to the right or upside such that they must be visible to \( R_i \) if \( v_j \) in \( G \) are adjacent to \( v_i \).

**Example:** In Figure 3(c), vertices 3 and 4 are adjacent but in Figure 4(iii), \( R_4 \) is not adjacent to \( R_3 \). Hence, stretch \( R_3 \) to the right in such a way that it is vertically adjacent to \( R_4 \).

By following the above steps, we obtain a 2-visibility drawing of \( G \) as shown in Figure 4(vi).

2. **Horizontal Expansion**

Remove the horizontal line segments by expanding each \( R_i \) preferably from right to left to maintain the rectangularity of rooms. Alternate layouts can be obtained by expanding the boxes to their left.

**Example:** Horizontal expansion of a 2-visibility drawing of \( G \) is shown in Figure 5(b).

3. **Vertical Expansion**

Remove the vertical line segments by expanding \( R_i \) preferably from top to down to maintain the rectangularity of rooms.

**Example:** Vertical expansion of Figure 5(b) is shown in Figure 5(c).

For a further illustration of the algorithm, refer to Figures 6 and 7.

Figure 5. Obtaining horizontal expansion and vertical expansion from the 2-visibility drawing.
3. Conclusion

We have known that architectural design is a multi-constraint problem and it cannot be addressed using the mathematical tools. At the same time, we can notice that there are some stages of architectural design where mathematicians can help designers, in particular, for developing an initial layout. In many cases, the initial layout needs to be developed while satisfying the given adjacency constraints, which mainly falls in the domain of graph theory. The problem of obtaining a floor plan for a given adjacency graph can be categorized into many mathematical sub-problems, where each one of them is quite difficult to handle. For example, how to identify the existence of a rectangular floor plan for a given adjacency graph. If it exists, how to construct it. If it does not exist, how to construct an orthogonal floor plan for the given graph. If there does not exist a floor plan for the given graph, how to transform the graph into a new graph for which a floor plan exists (for further details, about the use of graph theory in architectural design and related problems, refer to Nassar [10]). In the literature, there only exist few algorithms for constructing either a rectangular floor plan or an orthogonal floor plan for the given planar triangulated graphs, i.e., it is required to have an algorithm that can always generate a floor plan (either a rectangular floor plan or an orthogonal floor plan) for the given planar graph. To address this problem, we proposed an algorithm that generates rectangular and, if required, orthogonal floor plans from a given adjacency graph. It can be argued that the presented work is restricted to adjacency constraints only, whereas to have a floorplan with architectural meaning, at least dimensional constraints need to be considered. Recently, when given a rectangular layout, Upasani et al. [19] propose a method based on linear-optimization to adjust the geometric dimensional constraints of a given rectangular layout while keeping the topological adjacency
relations unchanged. This means that, once we can generate a layout, we can think of using a similar approach for addressing the dimensional constraints. Our ultimate goal is to build a general system for the automatic generation of floor plans, i.e., to provide architects with design aids, that is, algorithms that can generate good candidate solutions, taking adjacency requirements into account and that can be further improved and adjusted by them, to provide better solutions to the user.

![Image](image.png)

Figure 7. Obtaining horizontal expansion and vertical expansion from the 2-visibility drawing given in Figure 6.

**Acknowledgement**

The research described in this paper is supported by BITS Pilani and is evolved as a part of the research project Mathematics-aided Architectural Design Layouts (File Number: ECR/2017/000356) funded by the Science and Engineering Research Board, India.

**References**


