PROJECT REPORT

ON

ENUMERATION OF MAXIMAL RECTANGULAR FLOOR PLANS

Under the guidance of Dr. Krishnendra Shekhawat

Assistant Professor Department of Mathematics

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

(Sept 2019)

Submitted by:

Lakshya Agarwal 2017B4A70630P

Rahil N Jain 2017B4A70541P

Acknowledgment

We want to take this opportunity to express our sincere gratitude and deepest regards to our teacher and guide Prof. Dr. Krishnendra Shekhawat for his ideas, guidance and the way he helped us traverse alternate routes to success whenever we faced major roadblocks. Without his insight and brain-storming, the project could not be completed.

Abstract

First, the task is to generate all inner graphs on ‘n’ vertices for which Rectangular floor plan is possible. The idea is to first construct a path graph and then to add edges between two vertices if the length of the shortest path between them is found to be two. This way we will obtain different triangulated graphs and then we have to check various conditions on it to make sure if there exists a corresponding floor plan for it.

Given a dual graph of an RFP, the second task is to add 4 more vertices to it so that it forms a maximal dual graph. This method is called “IO” where ‘I’ refers to the inner vertices and ‘O’ refers to the outer ones.The initial graph is referred to as the inner graph with n vertices and the final result as the outer graph with n+4 vetices. This method gives all possible non-isomorphic maximal graph for n+4 vertices for a given inner graph of n vertices.

Table of Contents

Introduction

In architectural design, the first step is to build a floor plan as per the adjacency constraints. Mathematically, the problem is to draw a rectangular floor plan(RPF) corresponding to the planar graph satisfying all the adjacency constraints; in which each rectangle corresponds to a vertex in the planar graph and connectivity of this rectangle is defined by the edges in the planar graph.In a rectangular floor plan (RFP) each room is rectangle along with its boundary. This conversion of planar graph into a rectangular floor plan is called Rectangular Dualization. Rectangular Dualization was originally introduced to generate rectangular topologies for floor planning of integrated circuits.

A graph which is the dual graph of a rectangular floor plan is called rectangular floor plan graph.Rectangular floor plan graph is said to be maximal if drawing a rectangular floor plan is not possible of the graph obtained by adding any new edge to it.Two rectangular floor plans are said to be isomorphic if and only if one can be derived from the other using translation, rotation, reflection and scaling of the rooms in the RFP.

In our work, we first studied various algorithms by which a maximal rectangular floor plan could be obtained on ‘n’ vertices. The first being introduced by Dr. Krishnendra Shekhawat for enumerating all maximal rectangular floor plan(MRFP) on n vertices by using the MRFP on ‘n-1’ vertices. Dr. shinichi Nakano also wrote a paper on Enumeration floor plans on n rooms which could be used to draw MRFP by adding four rooms to the outer boundary of RFP. As the time complexity for it being (n)! It couldn’t be used for enumeration of maximal RFP for larger than 8 vertices.
Then we tried to come up with our own algorithm to enumerate MRFP for ‘n’+4 vertices by firstly drawing all dual graphs on ‘n’ vertices and then adding 4 boundaries to make a MRFP for n+4 vertices . We also used some of the conditions of the paper “Heuristic method to check the realisability of a graph into a rectangular plan” written by A. Recuero, M. Alvarez and O. Rio to check the realisability of inner graph.

Problem Statement

Given number of vertices(n+4), the problem is to enumerate all Maximal rectangular floor plan having ‘n+4’ rooms.
A RFP(n) is said to be best connected if and only if it has only four rooms on the boundary. So, the idea is to construct a rectangular floor plan on ‘n’ vertices and then later to add four rooms on the outer boundary to convert it into maximal rectangular floor plan(MRFP). The RFP to which the outer four rectangle are being added is termed as inner rectangle floor plan. And the dual of this floor plan is termed as inner graph.

Problem 1: Inner graphs

To enumerate all inner graphs for n vertices.

Problem 2 : IO Method

To add 4 boundaries to the inner graph giving an output maximal dual graph for n+4 vertices.

1. Inner Graph

A. Construction

To find all Inner graphs on ‘n’ vertices, firstly, a path graph is drawn having ‘n-1’ edges and then an edge is added between two vertices if the length of the shortest path between them is two. And this will result into formation of triangulated graph. If we continue the process we will find all different triangulated graphs for a given n. This way all non-isomorphic inner graphs were drawn.

B. Elimination of cases

After constructing inner graphs on ‘n’ vertices, we have many graphs which are redundant and many others for which Rectangular floor plan(RFP) doesn’t exist. The conditions to eliminate these graphs are listed below -

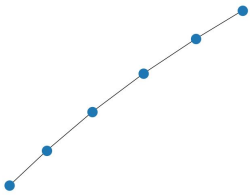
1. **Planarity** - If the inner graph obtained through the algorithm is non-planar, then we will ignore the case as RFP doesn’t exist for a non- planar graph.
2. **K-4 as subgraph** - If any of the inner graph has k-4 graph (complete graph on 4 vertices) as a subgraph, then also RFP for that graph doesn’t exist.
3. **Cycle case -I** : If the wrapping of any vertex forms a cycle, then that vertex can have only one component i.e. the cycle. If it has more than one component, then RFP for it is not possible as the cycle component will alone form a closed wrapping and the other component couldn’t be connected to it.
4. **Cycle case-II :** If 2 vertices have the same wrapping cycle then the RFP of any dual graph containing this kind of subgraph doesn’t exist.
5. **Three-Component case** : If we consider any two cut vertex in a graph and after their removal, graph has more than ‘2’ components, then RFP for it doesn’t exist. Similarly, if we choose any two non-cut vertices and see, if after their removal graph results into more than ‘2’ components, then also RFP for it doesn’t exist.
6. **Isomorphism** - In each step of construction of inner graph, we will check if the new inner graph obtained is isomorphic to any of the previous graph. If it is we will ignore the graph as it is redundant and already stored in the list of possible inner graphs.

Below is the simulation of inner graph using the algorithm discussed above for ‘n+4’= 10, where ‘n’ is the number of vertices in the inner graph.

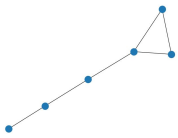
Enumeration of all inner graphs when n=6

Starting with Path graph on ‘n’ vertices which would be having 5 edges and adding edges between two vertices if the length of the shortest path is 2. And then drawing the non-isomorphic graphs and eliminating ones for which RPF doesn’t exist, we obtain different inner graph. Here, inner graphs having same no. of edges are clubbed together.

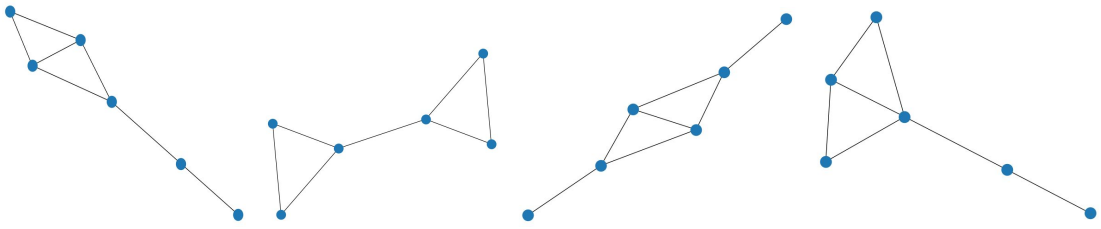
Number of edges = 5



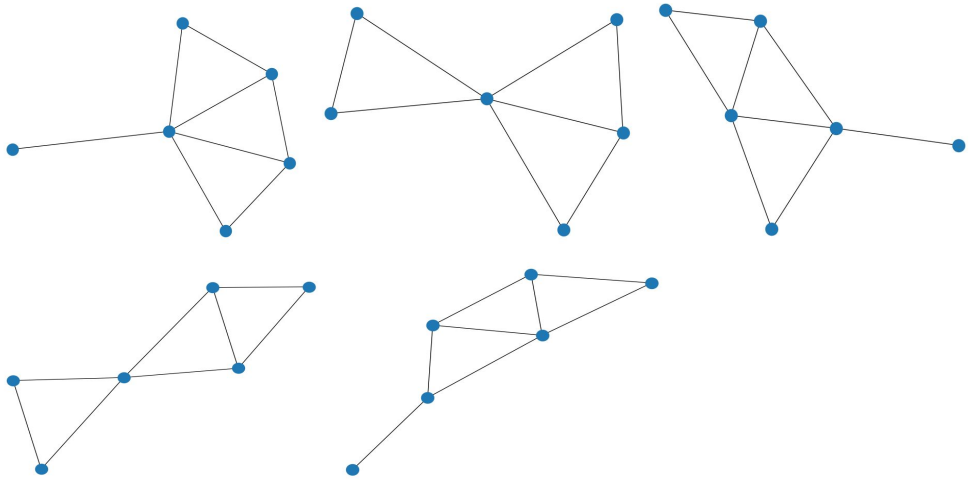
Number of edges = 6



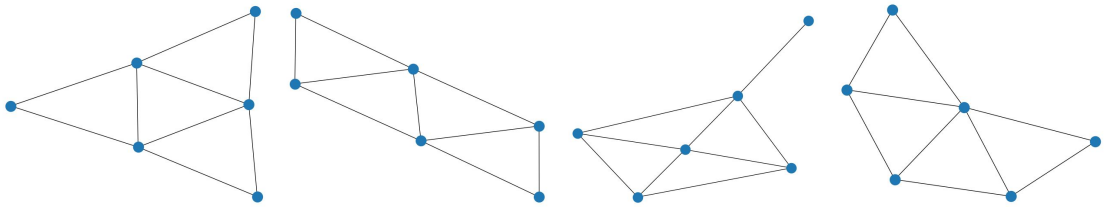
Number of edges = 7



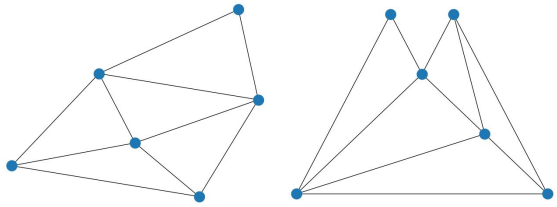
4. Number of edges = 8



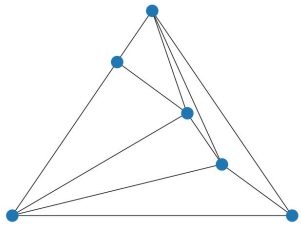
5. Number of edges = 9



6. Number of edges = 10



7. Number of edges = 11



Here, **total no. of inner graph** generated by the code came to be **18** which matches with the solution drawn by hand. The code is checked for all the in between 1 and 9 and it is giving the same manually expected answer.

New Issues

The fig 3. Given below does not exist when manually verified. So the possible reason could be that there is a quadrangle in the dual graph or another possible reason is that vertex 3 is connected to both the vertices of the common edge between the two wrapped cycles of 5 and 1.

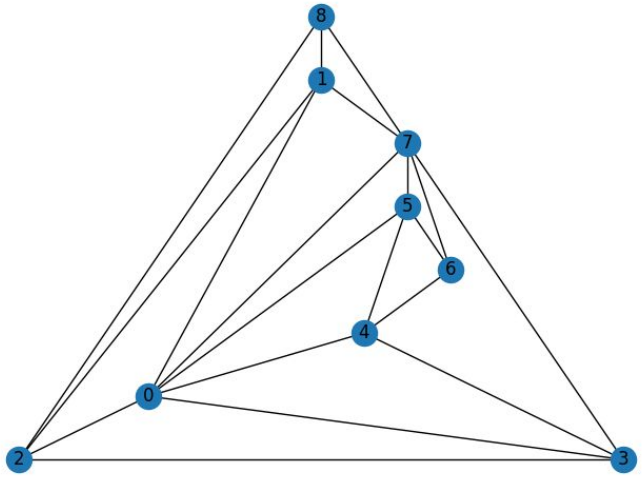


Fig 3. Represents a planar graph whose RFP does not exist.

New Heuristics

There could be many other possible heuristics such as

- There should not be a vertex connected to the common edge between 2 wrapped cycles.
- Finding the outer vertices and checking if there exists a connected path among the outer vertices.

2. The IO Method

Definition 1: Inner Graph

The initially given dual graph of the given RFP on which we have to add four rooms.

Definition 2 : NESW vertices

The four vertices to be added are names as NESW being ‘North’ , ‘East’ , ‘South’ ,’West’ respectively.

Definition 3 : Outer Graph

The final graph after adding the NESW vertices is called the outer graph.

Definition 4 : Wrapped Vertices

Those vertices which have a cycle in the subgraph formed from its neighbors. In the fig 4. The vertex 0 is a wrapped vertex because the subgraph formed from its neighbors 1,2,3,4, has a cycle i.e, 1-2-3-4-1.

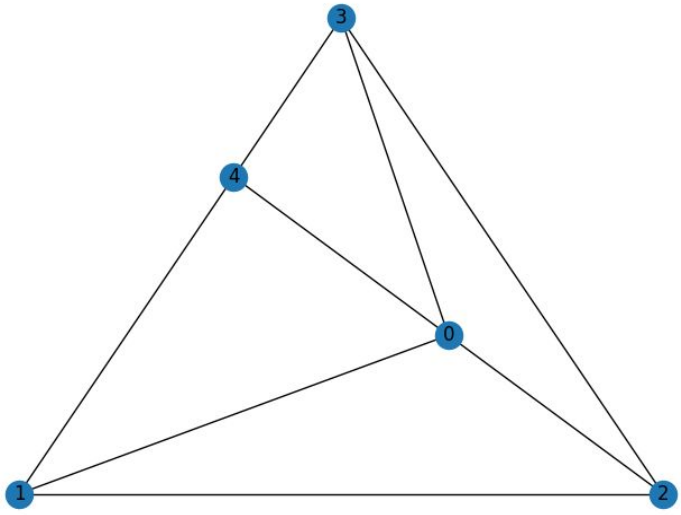


Fig 4. Representation of a wrapped cycle.

Remark 1 : -

Each vertex in the inner graph must have minimum degree of 4 in the outer graph. This is because a rectangular block has 4 sides and hence is obvious.

Part 1 : Basics of edge counting :-

In 2014, Shekhawat [1] proposed the enumeration of a particular class of best connected RFP, i.e., the RFP having 3n−7 edges in their dual graphs. Let n be the number of vertices in the inner graph. So the outer graph will contain n+4 vertices. A best connected RFP is also a maximal RFP so a maximal RFP will have 3n-7 edges in its dual graph. This implies that the outer graph will have 3*(n+4)-7 edges. Let K be 3*(n+4)-7 which is 3n+5.

The basis of this method is to remove the evident edges from K and find all non-isomorphic combinations with the rest, thus giving us all different kinds of outer graphs for a given inner graph.

Below is the list of all different kinds of edges.

1. I-I edges:
    - Edges of the given inner graphs. Let this be ‘m’.Remaining number of edges after removal of I-I is K-m.
2. O-O edges:
    - Edges between NESW vertices. They are 4 . Remaining number of edges after removal of O-O is K-m-4.
3. I-O edges:
    - This category of edges create different combinations of non-isomorphic maximal dual graphs. But still there are some evident cases of special vertices which can be identified and assigned edges to them.
    - Edges of cut vertices: A cut vertex separates the graph into components. There can’t be more than 2 components formed because a rectangular block has 4 sides and 2 pairs of non adjacent sides and there isn’t any set of sides more than 2 which are mutually non adjacent. So therefore the cut vertices in a dual graphs form 2 disconnected components on their

removal. Thus the number of I-O edges to be added to a cut vertex in the inner graph is 2.

- Degree 1 vertices: Degree 1 vertices in the inner graph have 1 side covered and 3 more sides left to cover, so there are 3 I-O edges for a degree 1 vertex.
- Wrapped vertices: There could be few vertices which are wrapped by more than 4 vertices. If we look into the RFP of such graph we see that all sides of the wrapped vertices are covered by the wrapping cycle and its obvious that the cycle covers it. So the number of I-O edges for a wrapped vertex in the inner graph would be 0.

After counting on all the above edges, let the remaining edges be Y.
Now our aim is to add the remaining edges to the outer graph. So we find out the maximum and minimum number of I-O edges to be added to the rest of the inner graphs

Maximum number : As the inner graph is connected, each vertex has at least one of its sides block by the blocks of the inner graph. So as each vertex has 4 sides to be covered, the maximum number of I-O edges that can be added to the rest of the vertices is 3.

Minimum number : Assume a generic vertex v, which doesn't come under the above criteria of vertices. Let the degree of v be d. Now the maximum number of sides the neighbors of v can take is d. So the minimum number of I-O edges for v to be added must be 4-d. Now if d is greater than or equal to 4, then the minimum number must be zero.

Now let us have a count of all the minimum required edges for all the vertices. Let the remaining number of edges to be added be Z.

We have the maximum number of I-O edges to be added to all the vertices. So we can have all combinations of adding Z I-O edges while satisfying the maximum number.

Let us illustrate the above method with examples.

Example 1:-

Let the graph given in fig 3. Be the inner graph with 6 vertices. Our aim is to add 4 outer boundaries to make it into a maximal dual graph with 10 vertices.

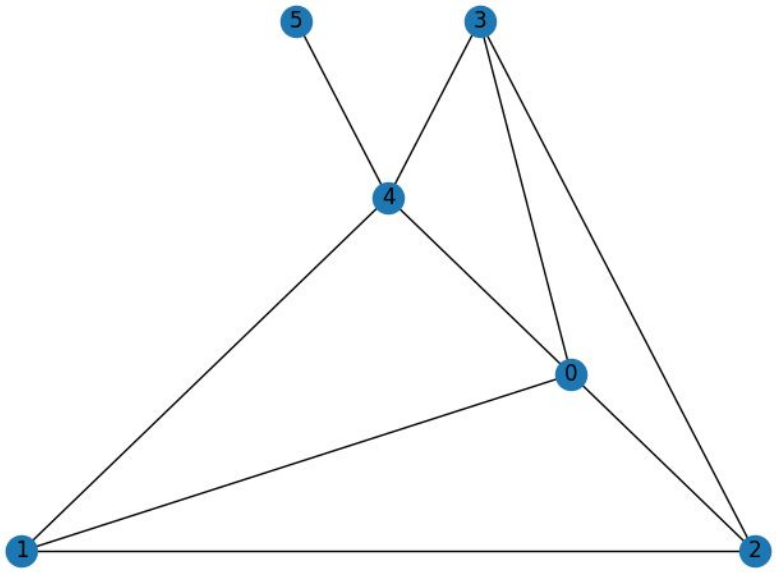


Fig 5. Inner graph illustration.

Firstly, we find the total number of edges in the outer graph.
K= 3*(n+4)-7
K= 3*(6+4)-7
K=23 edges

Stage 1:- Initializing the outer graph

Now we count I-I and O-O edges into the outer graph
I-I edges = 9
O-O edges = 4

Remaining edges = 23 - 9 - 4
= 10 edges

At this current stage the outer graph looks like fig 4. Which as 9+4 edges.

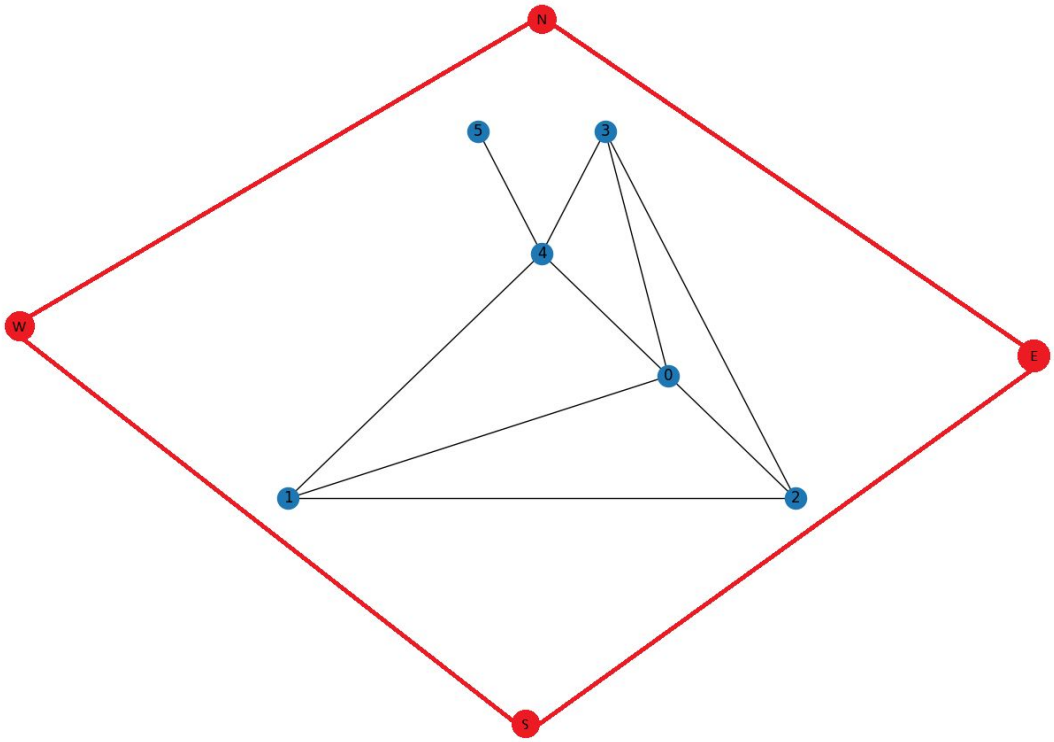


Fig 6. Representation of outer graph after stage 1.

Stage 2: Assigning number of I-O edges

Special vertices:
Vertex 5 is of degree 1 so it takes 3 I-O edges .
Vertex 4 is a cut vertex so it takes 2 I-O edges.
Vertex 0 is a wrapped vertex so it take 0 I-O edges.

Other vertices:
Vertex 1 has degree 3, so min= 4-3 = 1, and max=3.
Vertex 2 has degree 3, so min = 4-3=1, and max=3.
Vertex 3 has degree 3, so min = 4-3=1, and max=3.
Now we count on the minimum required I-O edges for each inner vertex and mark it on our outer graph. Fig 4. Represents the graph after completion of this stage
Remaining edges after stage 2 is 10-3-2-0-1-1-1= 2.

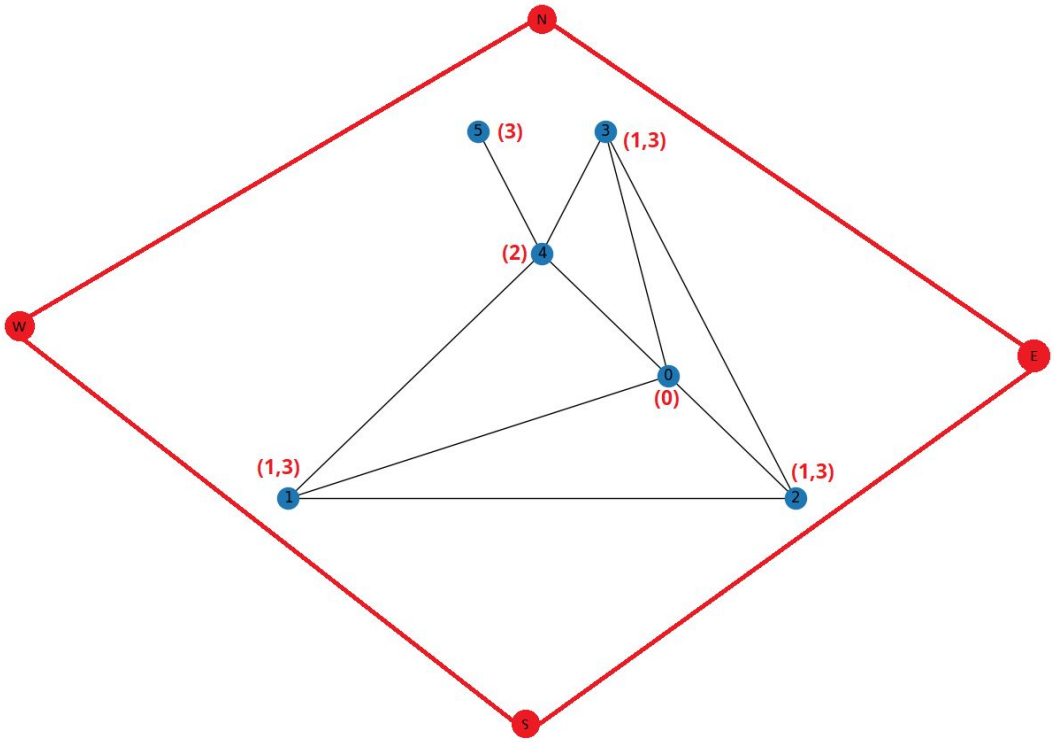


Fig 7. Representation of outer graph after stage 2.

Now we have to add 2 I-O edges. So we find all possible non-isomorphic combinations of adding 2 edges to vertices 1,2,3.
So the possible non-isomorphic combinations of adding 2 edges are
1,1
1,2
1,3
2,2.
So therefore there are 4 non-isomorphic outer graphs for the given inner graph.
End of example.
End of basics of counting.

Part 2 : Completion problem

Let us consider the outer graph after taking the combination 1,2. Let this graph be G.
Fig 5. Represents G

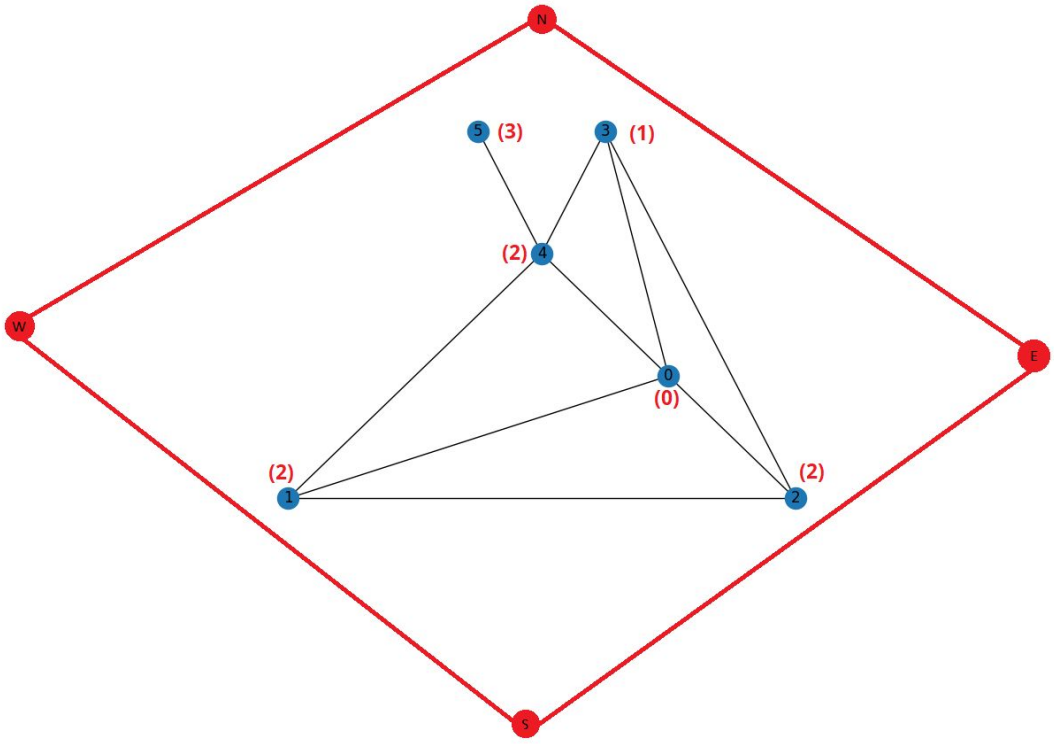


Fig 8. Representation of a partially completed outer graph.

Remark 3:

All cut vertices must be connected to the same pair of outer vertices.
The above remark can be proved by contradiction.

There are few steps to add the I-O edges.
Divide the inner graph into components of biconnected subgraphs.

- Find all cut vertices and make a set of them.
- Create biconnected subgraphs by creating a separation between the cut vertices, but the cut vertices exists in both the biconnected subgraphs.
- As the all cut vertices must have 2 I-O edges with the same outer vertices, without loss of generality let us take all cut vertices to be connected to West and East.

- Now the subgraphs containing only the first or the last cut-vertex from the set of cut-vertices must have I-O edges towards North-East-West and South-East-West respectively.
- Rest all the subgraphs must have I-O edges towards West and East.

In the above graph the cut vertex set contains only 1 element i.e, vertex 4.
The newly formed subgraphs are (5,4) and (4,3,2,1,0) . Now the subgraph (5,4) must be connected to North-East-West vertices and the subgraph (4,3,2,1,0) must be connected to South-West-East vertices.

The final outer graph for the given example is shown in fig 6.

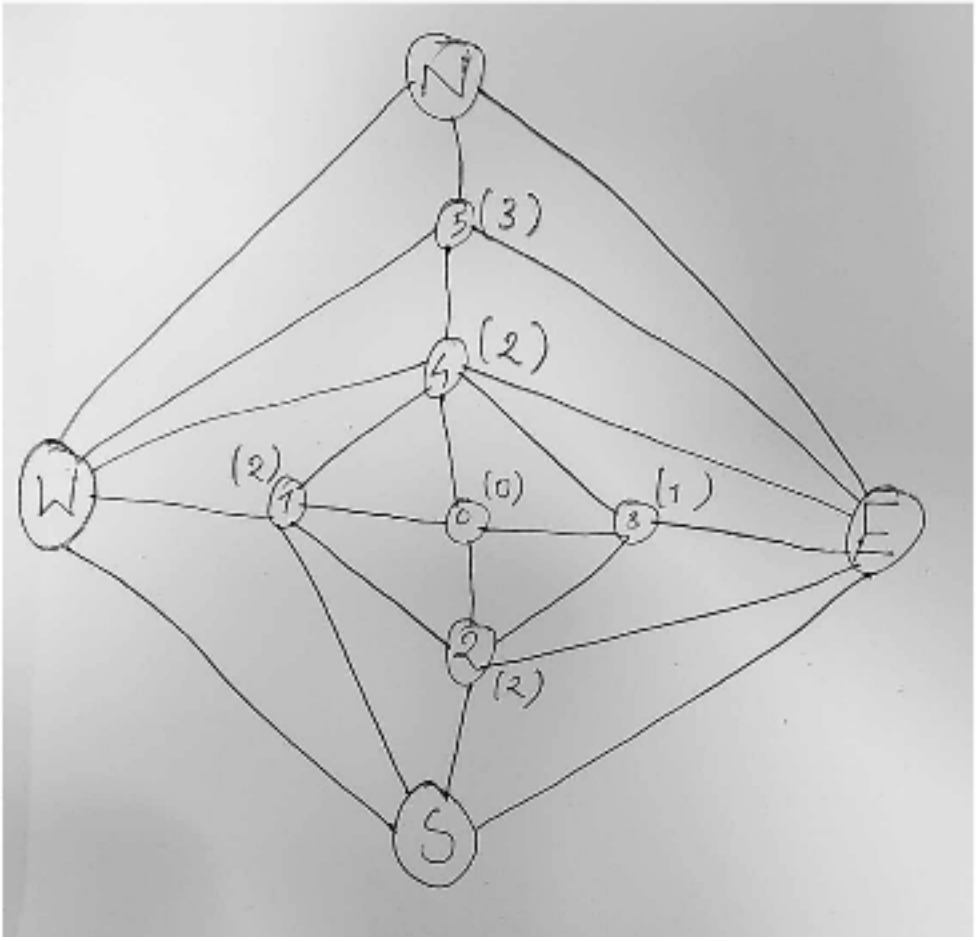


Fig 9. Representation of final outer graph.

Conclusions

Inner graphs:

The inner graphs method was initially found to use it to enumerate maximal RFPs but it can also be used to enumerate all RFPs for a given number of vertices.
The heuristics that are found are observing manually for the inner graphs being generated till n=10. There is no proof for the existence of many of the heuristics of the inner graphs. There are some more heuristics found but not included in the report as enough work has not been done on the new heuristics. Overall it is evident that completion of the heuristics in itself is tough so till then the results from this could be used but not with assurance.

IO Method:

The logic of the method is graphic theoretically correct. But its implementation is not yet complete. This method can be used in various other problems and fields of Rectangular floor planning.

Link to the generated RFPs
https://drive.google.com/open?id=1rz1l684gOu_popuaxz1MZN0F9UvHx40x

Link to github repository
https://github.com/thunderbolt06/Enumeration-of-RFPs

Recommendations

Inner graphs:

- Completion of all the new heuristics found need to be done.
- Complexity of many heuristics needs to be reduced
    - By merging the heuristics which are redundant
    - By some better logical solution for a given heuristic
- Writing a modular program which can add and remove heuristics easily
- Writing a program to check it there exists an RFP for the given input dual graph
- Finding applications for this concept in various areas

IO Method:

- Completion of the program of this method.
- Using this program to find the existence of RFP for a given dual graph.

Appendix

ALGORITHM TO GENERATE INNER GRAPHS

Size of a graph i.e., number of edges is represented by |E|

Input-
Number of vertices of the inner graph say it 'n'. (n is a finite natural number)

Output-

A. The number on inner graphs of n vertices.
B. Edgelist of all inner graphs possible for n.

Procedure-

```
/*Main */
Create a path graph with n vertices and name it G
Print Edges of G
Glist = Null                    //Glist is a list of graphs
Append G in Glist               //G is added to Glist
While Glist is non-empty Do
        For iterGraph in Glist do
                Print edges of iterGraph
        /*End of for loop*/
        Glist= MakeBig (Glist)
/*End of Main*/
/*MakeBig Function: -
        Parameters: List of Inner graphs of size |E|
        Output: List of Inner graphs of size |E+1|
*/
MakeBig (GraphList)
For loop graphs 'G' in GraphList
        Len2List = Array of all pair of vertices of distance 2
        /* Len2List contains tuples of vertices (x, y) which are at distance 2*/
        NewGraphArray = Null
        For Edgetuple in Len2List
Create graph 'G1' from graph g by adding Edgetuple
/* G1 has 1 more edge than G*/
Choose = 1
/*Planarity*/
```

```
        If (G1 is non-planar) then
                Choose =0
        /*K4 Subgraph*/
        Else If (G1 has k-4 as subgraph) then
                Choose = 0
        /* Isomorphic to Previous Graph Check*/
        Else If (G1 is isomorphic to any graph in 'NewGraphArray') then
                Choose = 0
        /* Bridge Case and Cut vertex Case*/
        Else
                CutVertexList = Array that contains all the cutvertices in Graph G1
                For (a, b) edge in edgelist of G1:
                        Copy1 = copy of graph G1
                        If both a, b belong to CutVertexList or both a, b don't belong to
        CutVertexList then
                                Remove vertices a and b from Copy1 graph
                                If (Number of components of Copy1 > 2) then
                                Choose = 0
        DoubleCycleList = Null
        /*Wrapping Check*/
        For vertices V in Graph G1:
                If degree of V>4 then
                        Copy1= Subgraph induced by neighbors of V
                        CycleList = List of all cycles in Copy1
                        For Cycle in CycleList:
                                If len (Cycle)! = len (Copy1) and len (Cycle)>3:
                                        Choose = 0
                                Else
                                        Append to DoubleCycleList

        /*Double Cycle Check*/
        J=0
        For i1 in DoubleCycleList
                J = J +1
                For i2 in DoubleCycleList [ J:]
                        If i1==i2 then
                                Choose = 0
        /* Append if above conditions satisfy*/
        If Choose==1 then
                Append Graph G1 to NewGraphArray list
Return NewGraphArray
/*End of MakeBig function*/
/*End*/
```

References

1. Recuero, A., Rio, O. and Alvarez, M. (2000). Heuristic method to check the realisability of a graph into a rectangular plan.

2. Shekhawat, K. and kumar, V. (2018). An Algorithmic Approach for Constructing Floor Plans with Circulations

3. Reading, N. (2012). Generic rectangulations.