

Queue Layout of Planar 3–Tree

Digvijay Pandey

A Thesis Submitted to
Indian Institute of Technology Hyderabad
In Partial Fulfillment of the Requirements for
The Degree of Master of Technology



Department of Computer Science & Engineering

June 2019

Declaration

I declare that this written submission represents my ideas in my own words, and where ideas or words of others have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources that have thus not been properly cited, or from whom proper permission has not been taken when needed.




(Signature)


Digvijay Pandey
(Name)


CS17MTECH11024
(Roll No.)

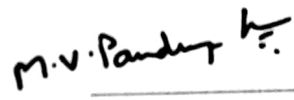
Approval Sheet

This Thesis entitled Queue Layout of Planar 3-Tree by Digvijay Pandey is approved for the degree of Master of Technology from IIT Hyderabad


Antony Franklin Examiner
Dept. of Computer Science & Eng
IITH


U. Ramakrishna Examiner
Dept. of Computer Science & Eng
IITH


(Dr. N. R. Aravind) Adviser
Dept. of Computer Science & Eng
IITH


(——) Chairman
Dept. of Computer Science & Eng
IITH

Acknowledgements

Firstly, I would like to express my sincere gratitude to my thesis advisor Dr. N.R. Aravind for the continuous support in my thesis work and related research, for his patience, motivation, and immense knowledge in theory. His guidance has helped me to think like a researcher, he has also reviewed presentations and this thesis work which resulted in writing a better content for readers. I would also like to thank the professors for their valuable comments and suggestions during the presentations. It has helped me to think in new directions and improve my research work. Finally, I would express my deep gratitude to my parents for providing me tremendous support throughout my research work. This achievement would not have been possible without them.

Dedication

Dedicated to:

My Parents

Divya and Shaivya (Sisters)

Priti Sharma (IITB)

Abhay Pratap Singh (Friend)

Prativa and Anamika (Friends)

Abstract

Graph drawing is essential for data representation. This thesis addresses various graph drawing techniques, their implementation, and enhancements. First, we discuss the 3D grid drawing techniques. The subsequent chapters address the Stack Layout and Queue layout of the graph. The application of Stack and Queue layout and its importance also discussed.

Section 4, dedicated to outerplanar Graph. In this chapter, we have discussed how outerplanar Graphs are implemented and their queue and track layouts. The most important part of this thesis is chapter 5, in which the implementation of planar 3-Tree is given. An outerPlanar graph and Planar 3-Tree are internally related. The known upper bound of the queue number of planar 3-Tree is 7. We have implemented the queue layout of 2-Layer planar 3-Tree using two queues and then generalized this experiment for any arbitrary number of levels.

Contents

Declaration	ii
Approval Sheet	iii
Acknowledgements	iv
Abstract	vi
Nomenclature	viii
List of Figures	1
List of Algorithms	2
1 Introduction	3
1.1 Aesthetic criteria	3
1.1.1 Area	3
1.1.2 Aspect Ratio	3
1.1.3 Sub-graph Separation	4
1.1.4 Closest Leaf	4
1.1.5 Farthest leaf	4
1.1.6 Size	4
1.1.7 Edge Length	4
1.1.8 Angular Resolution	4
1.1.9 Symmetry	4
1.2 Graph Drawing Techniques	5
1.2.1 Orthogonal Drawing	5
1.2.2 Poly Sub-tree Drawing	5
1.2.3 Upward and Non Upward Drawing	5
1.2.4 Planar Drawing	5
1.2.5 Grid Drawing	5
1.3 Application	6
1.4 Topology Shape Metrics	6
1.4.1 Topology	6
1.4.2 Shape	6
1.4.3 Metrics	6
2 Related Work	7

2.1	Planar Graph	7
2.2	Tree Drawing	7
2.3	Hierarchical layout algorithms	8
2.4	Orthogonal Drawing	8
2.5	Stack and Queue layout	9
2.6	Track layout	9
3	Track Layout	10
3.1	3D Grid Drawing	10
3.2	r-partite Graph Drawing	12
3.3	3-Track Layout Tree Drawing	13
4	Stack and Queue layout	15
4.1	Basics of layout	15
4.1.1	Layouts of Fixed Order	15
4.1.2	Graph with Queue number 1 and stack number 1	16
4.1.3	Queue number and Stack number Trade off	16
4.2	Queue number and Stack number in a nutshell	17
5	Outer-planar Graph	18
5.1	Implementation of outer-planar Graph	19
5.1.1	Terminology	19
5.1.2	Procedure	19
5.2	Algorithm	19
5.3	Track layout of outer-planar Graph	21
5.4	2-Queue Layout of outer-planar Graph	21
6	Planar 3-Tree	22
6.1	Definition	22
6.2	Implementation of planar 3-tree	23
6.3	5-queue Layout of 2-layer Planar 3-Tree	24
6.3.1	Terminology	24
6.3.2	Queue layout of 2-level planar 3-tree	24
6.3.3	Implementation	26
6.4	5 Queue layout of n -layer Planar 3-Tree	27
6.5	Result	28
6.5.1	n layer 3-tree	28
6.5.2	Converting into levels	28
6.5.3	Queue layout	29
7	Conclusion and Future work	30
7.1	Summary	30
	Bibliography	31

List of Figures

3.1	X - Crossing	11
3.2	3D Grid Drawing	11
3.3	3D Grid Drawing with a Prime number 'P'	12
3.4	3D Grid drawing for Bi-partite Graph	13
3.5	Edge Wrapping into the 3-tracks	13
3.6	3D Grid drawing for Tree	14
4.1	k - rainbow	15
4.2	k - twist	16
4.3	k - Necklace	16
5.1	Maximal Outer Planar Graph	18
5.2	Upper envelop and Lower envelop	20
5.3	Graph with height of neighboring vertices differ by at most 2	20
6.1	Planar 3-tree	22
6.2	2-Level Planar 3-Tree	23
6.3	Planar 3-Tree and level split	26
6.4	5- Queue layout of Planar 3-Tree	26
6.5	n - layer Planar 3-Tree	28
6.6	Level wise placement of vertices of planar 3-tree	28
6.7	Queue layout of Planar 3-Tree	29

List of Algorithms

1	Outer Planar Graph: Adjacent vertices are at a vertical height at most two.	19
2	Planar 3-Tree	23
3	Queue layout of 2-Layer Planar 3-Tree	24
4	Queue layout of n -Layer Planar 3-Tree	27

Chapter 1

Introduction

Interactive and scientific representation of data using a computer-supported tool so that a user can better understand it is called Information Visualization. Graph drawing [1] is a graphical representation of Information. With the help of the graph, the Human visual system can grasp a considerable amount of data (which is impossible without graph visualization) and also find the pattern in data. The limitation of the Human visual system is that it can only work with image data. Mapping data to an image in an interactive way is graph drawing [2].

The data structure for artificial intelligence based production of geometric representations of inter-related information is a graph where vertices and edges represent the entities and the relationships between them respectively. Placement of vertices and edges in a coordinate system is crucial. A graph is useful if it follows some aesthetic criteria [3].

1.1 Aesthetic criteria

1.1.1 Area

The area of a graph G is defined as coverage of total space inscribed within all vertices. Theoretically, the area of the graph should be minimum, but for a better resolution, the area should be substantial. The area of a graph should give flexibility scaling in both ways.

1.1.2 Aspect Ratio

Aspect ratio is defined as the fraction of the shortest edge of a graph to the longest edge of the graph. Ideally, the aspect ratio should be 1. The graph should follow the user controlled aspect ratio. For example, if an end user needs 2.5 Aspect ratio, then the graph should be scaled up.

1.1.3 Sub-graph Separation

Let G be a graph with the root vertex, left sub-tree, right sub-tree v , X and Y respectively then $G(X)$ and $G(Y)$ should not be overlapped. If we add a new vertex either to $G(X)$ or $G(Y)$ then the updated area of $G(X)$ or $G(X)$ should also not be overlapped. The importance of Sub-graph separation is that it should allow the user to focus on context.

1.1.4 Closest Leaf

A leaf which is at the shortest Euclidean distance from the root/central vertex.

1.1.5 Farthest leaf

A leaf which is at the longest Euclidean distance from the root/central vertex. Usually, the closest leaf should be placed far from the central vertex so that the user can visualize it in a better way. The furthest leaf should be kept close to the central vertex so that the area requirement of the graph is minimum.

1.1.6 Size

If the graph is inscribed in a rectangle, then the area of the rectangle is termed as the size of the graph. So for an informative graph, the area of the rectangle should be minimum.

1.1.7 Edge Length

The edge length is defined as the distance between any two vertices of the graph. The total edge length is the sum of all the edge lengths. Ideally, all the edges should be of the same length.

1.1.8 Angular Resolution

Let G be a graph with a vertex v . Let e_1 and e_2 be two edges such that both edges shared a common vertex v then the angle subtended by these two edges on v is called angular resolution. The angular Resolution for each vertex of the graph should be uniform.

1.1.9 Symmetry

The Graph should be symmetric around an axis so that it can be wrapped around.

All the aesthetic norms are not mutually exclusive, so it can happen that the graph specifically satisfies aesthetic criteria, can not satisfy other criteria. We need to consider these unavoidable trade-offs. Aesthetic criteria may look useful and applicable to the Graph, but from the implementation point of view it turns out to be tedious.

1.2 Graph Drawing Techniques

Based on user requirement, there are multiple graph drawing techniques. Some of them are described below

1.2.1 Orthogonal Drawing

Alternating series of horizontal and vertical sub-trees. The advantages of this drawing are that the user can visualize and understand the information stored in a sub-tree precisely. The disadvantage of this drawing it occupies more space.

1.2.2 Poly Sub-tree Drawing

Let G be the Graph and $G_1, G_2, G_3 \dots G_N$ be the sub-tree of the graph, placing all these sub-tree in a coordinate system and then connecting every sub-tree by an edge is called Poly Sub-tree Drawing. It may happen that by connecting any two sub-tree by a straight edge may create a cross edge or it violated sub-tree separation criteria. So to overcome this problem we need to *bend* the edge.

1.2.3 Upward and Non Upward Drawing

In Graph G , when the level of every child node does not precede the level of the root. This phenomenon is called Upward Drawing. On the other hand, when the level of every child node succeeds the level of the root, then it is called Non upward Drawing. This drawing technique commonly used for drawing trees.

1.2.4 Planar Drawing

In Graph G , no of two edges should intersect each other. Every Graph need not be Planar, so Planar drawing does not apply to all graphs.

1.2.5 Grid Drawing

In Grid Drawing, all the vertices of Graph G lie on integer Coordinates. The objective of this drawing area is to draw a rectangle such that every vertex placed at integer coordinate. So it will eventually turn out every pair of vertex maintain at least unit Euclidean distance. The rectangle in which vertices are enclosed is called Inclusive Rectangle.

1.3 Application

The application of graph drawing is used in setting up network topology, Chip Design, UML modeling, Share Market, Statistical Learning, Social Network Analysis, resource allocation in Operating System, flow of computation in a graph, Google Map, etc.

1.4 Topology Shape Metrics

This drawing is practically used in most of the filed like Databases, Software Engineering, Operating System, Computer Network, Social Network graph, etc. The topology shape matrix is used for drawing an orthogonal grid hybrid graph. The advantage of using this approach is that it will satisfy the majority of aesthetic criteria. There are three components to topology shape metrics.

1.4.1 Topology

Two orthogonal Graphs will have the same topology if and only if one can be transformed into another by preserving the same edge set for each sub-graph.

1.4.2 Shape

Two Orthogonal Graph is said to have the same shape if and only if they have the same topology and preserve an angle between the edges of segments which are orthogonal to each other.

1.4.3 Metrics

Two orthogonal graphs can have the same metrics if and only if they are the same under translations and rotational operations. All the above three Topology, Shape and metrics follow hierarchical manner. If the previous layers are satisfied, then only the current level has a chance to satisfy. The main objective of the Topology Shape Metrics approach is to minimize the number of cross edges. Shape and Topology and shape section emphasize the area section of the graph. It satisfies the majority of the aesthetic criteria but with a lower priority.

Chapter 2

Related Work

Graph drawing criterion has been extensively studied[1]. What should be the criterion for a good graph is discussed in the book. A brief summary of Quasi-Upward Planarity, 3D-Orthogonal Box-Drawings, visualization of interconnected data, Metrics graph drawing algorithms, Planarity Checking, 3D Orthogonal drawing using splash and push approach, geometric thickness of graphs also given in the book. Apart from the basic aesthetic criterion here are related works based on the type of graph:

2.1 Planar Graph

G. Kant *et al.*[4] introduced a method to minimize the area, number of bends and angle between edges in the planar 3D Grid drawing. Time taken by this algorithm is $O(n)$ and space complexity is also $O(n)$.

2.2 Tree Drawing

Reingold *et al.*[5] talked about Tree Drawing algorithms. They have given a spaced optimal algorithmic approach for drawing trees in a tidy way such that they follow aesthetic criteria. Aesthetic criterion are not mutually exclusive so they have also given a new algorithm for this. Forest and trees which are not binary binary also discussed in this paper. Crescenzi *et al.*[6] proved that the Space Optimal Upward Drawing of binary is possible. Family of binary tree need $\Omega(n \log n)$ are and this bound is tight. They have also given a space linear ($O(n)space$) algorithm for the Upward Drawing of a binary tree and extended this result for Fibonacci trees. P. Crescenzi *et al.*[7] in an another paper talked about space linear AVL tree construction. In algorithm, each parent was placed above their child and no two edge intersect. T. M. Chan *et al.*[8] showed that for a n node binary tree which is also order-preserving, planar, strictly upward, straight-line can take $O(n^{1+\epsilon})$ area in the worst case. Here $\epsilon > 0$ can be any real number.

2.3 Hierarchical layout algorithms

Hierarchical layout algorithms [9] are used for presentation of data. In this paper they talked about data structures and algorithms, basic concepts of computer graphics and computational geometry, planarization technique for drawing Hierarchical graphs. Some open problems like performance-planarization trade-off, testing palanarity, line drawing approach, dynamic drawing algorithms [10], bend and area minimization had also discussed in the same paper. Directed graphs are useful for representing the information flow. Peter and Lin *et al.*[11] gave an algorithm for drawing directed graphs. The main three aesthetic criterion taken into consideration were:

- No arc pointing upward
- The Uniform distribution of nodes
- Minimum arc crossing

Emden *et al.*[12] gave a four pass fast algorithm for drawing directed graphs. These four passes are:

- Use network simplex algorithm for finding Optimal rank assignment.
- Vertex ordering within ranks for reducing crossings
- Optimal coordinates for nodes
- Spline edges

This algorithm basically avoids the common visual anomalies like edge crossings and sharp bends. Aesthetic criteria like Symmetry and balance were the secondary thing for this algorithm. Kozo *et al.*[13] discussed a two-step approach for drawing hierarchical graphs. In the first step, order the vertices in different levels to avoid edge crossing. In second step, place the vertices to enhance readability. Nature of problem and scalability is determined by theory and heuristic methods respectively.

2.4 Orthogonal Drawing

Roberto *et al.*[14] gave a network flow technique algorithm which minimizes the total number of bending edges. Output of this algorithm is the region preserving rectilinear grid embedding. This algorithm was also extended to k - region graphs. For n node graph algorithm takes $O(n \log n)$ time. Roberto *et al.*[15] gave another linear algorithm for planar grid embedding. Here vertices are placed at integer grid points. There are four silent features of this drawing:

1. For a graph G with n nodes, the total number of bends in planar grid embedding of G is $2.4n + 2$
2. Maximum number of bend per edge is 4
3. Length of each edge is linear, i.e. $O(n)$
4. The total area of planar grid embedding of G is $O(n^2)$

2.5 Stack and Queue layout

Stack layout and the Queue layout of a graph is extensively discussed in [16]. Let G be a graph with n vertices. Graph G can have $O(1) \times O(1) \times O(n)$ iff queue number of graph G is bounded by some positive constant. Actually, this question is same as asking "whether every planar graph has constant queue number?" [17] Track and queue number of some graphs are well known. Every tree and outer-planar can be simulated in 1, 2 queue(s) respectively.

2.6 Track layout

Di Giacomo *et al.*[18] proved that if a graph G has queue number which is bounded by some constant then track number of the graph G is also bounded by some constant. In same paper track number of some graph families are also discussed along with the finding track number of a graph in linear time.

Chapter 3

Track Layout

According to Dujmovic *et al.*[19] a (K, T) -Track layout of the graph G consists of a proper vertex T coloring of G , a total order of each vertex class, and a non-proper edge K coloring such that between each pair of color classes no two monochromatic edges cross. The Track layout is used to draw a 3-Dimensional Graph.

We first fix the linear set of vertices of the graph in the hierarchical manner such that no two vertices at the next level share the same color. The number of colors and tracks are inversely proportional to each other. A t -Track Layout for the graph G is the minimum number of tracks in G when the number of colors is minimized. A vertex t coloring of the Graph G is a partition.

A vertex t -colouring [20] of a graph G is a partition $V_i : i \in t$ of $V(G)$ such that for every edge $vw \in E(G)$, if $v \in V_i$ and $w \in V_j$ then $i \neq j$. The elements of t are colours, and each set V_i is a colour class. The chromatic number of G , denoted by $\chi(G)$, is the minimum number k such that G has a vertex k -colouring. Let there be two levels i and j and there are two edges ab and cd such that vertices a, c are at level i and b, d are at level j . If a precedes c and b precedes d then edges ab, cd are called X -crossing edges.

3.1 3D Grid Drawing

The 3D Grid drawing [21] is defined as placement of vertices at integer coordinates without X -crossing. For 2D Graph, an area is a concern, but for a 3D grid, the volume is considered. Now in 3D Grid drawing the goal is to minimize the total volume. The volume of the 3D grid is the volume of a cube that contains the 3D graph. Let i be any integer if we place vertices at (i, i^2, i^3) then no two edges will produce X -crossing. For any two edges to cross or intersect all four endpoints should be in the same plane. By placing vertices at (i, i^2, i^3) no four endpoints will not be co-planar. This can be proved using **Vander Monde** Matrix Equation.

Cross edges are not allowed in 3D grid drawing. Two edges will cross if they are in the same plane but reverse may always be true. Here we are choosing vertices from the different plane so there will not be any edge crossings.

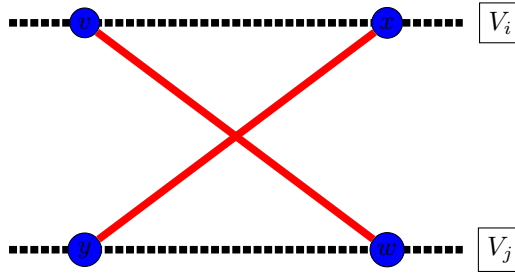


Figure 3.1:
X – Crossing

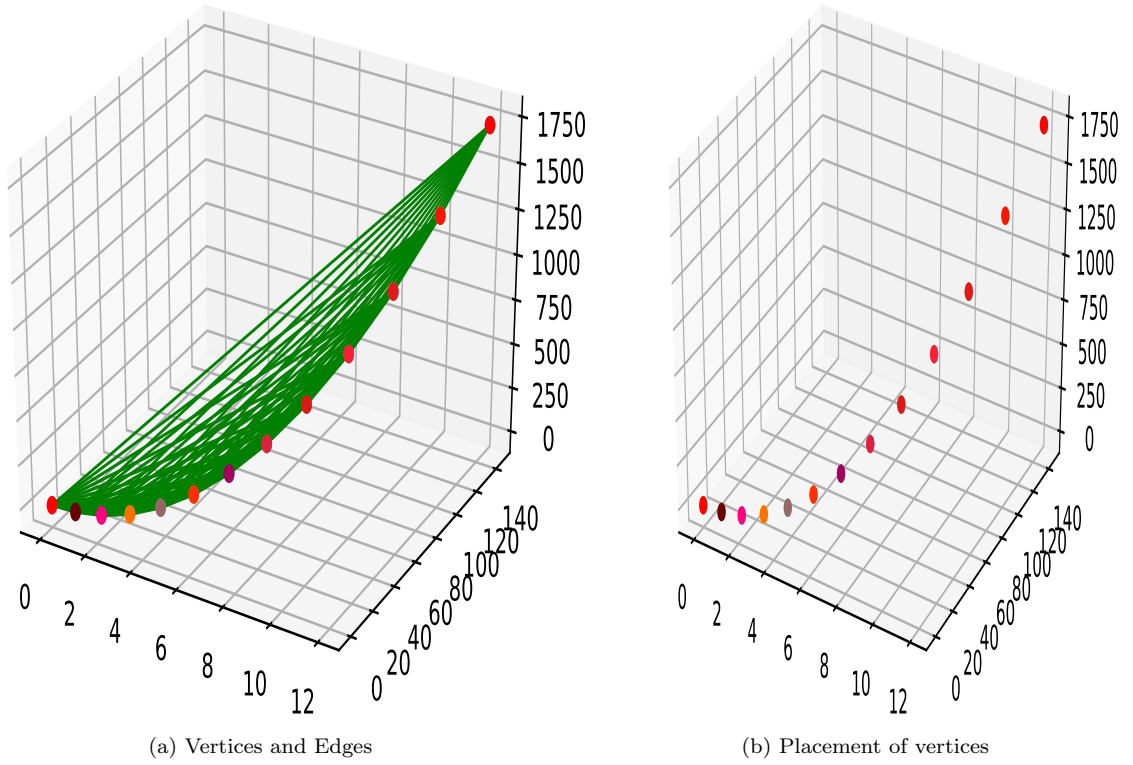


Figure 3.2: 3D Grid Drawing

Total Volume of the above drawing is $(n \times n^2 \times n^3) = O(n^6)$. To reduce the volume we select a prime number p such that $n < p < 2n$. Now if we placed vertices at [22] $(n\%p, n^2\%p, n^3\%p)$

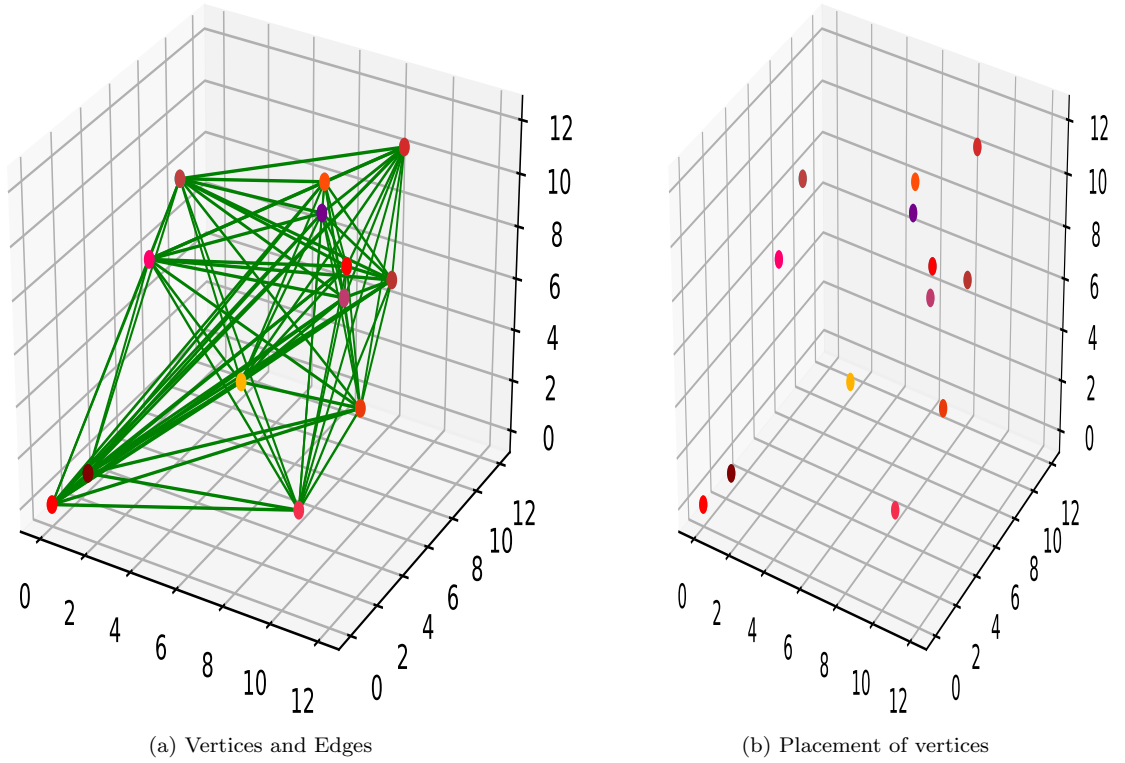


Figure 3.3: 3D Grid Drawing with a Prime number 'P'

Total Volume of the above drawing is $(n \times 2n \times 2n) = O(n^3)$.

3.2 r-partite Graph Drawing

Let the total number of vertices be n . A complete r-partite graph $K_r(n)$ will have a 3D dimensional Grid drawing [23] if we place vertices at

$$V_i = \{(i, t, it) : 0 \leq t < n \equiv i^2 \pmod{p}\}$$

where $p \geq 2r - 1$, $N = pn/r$ and $0 \leq i \leq r - 1$.

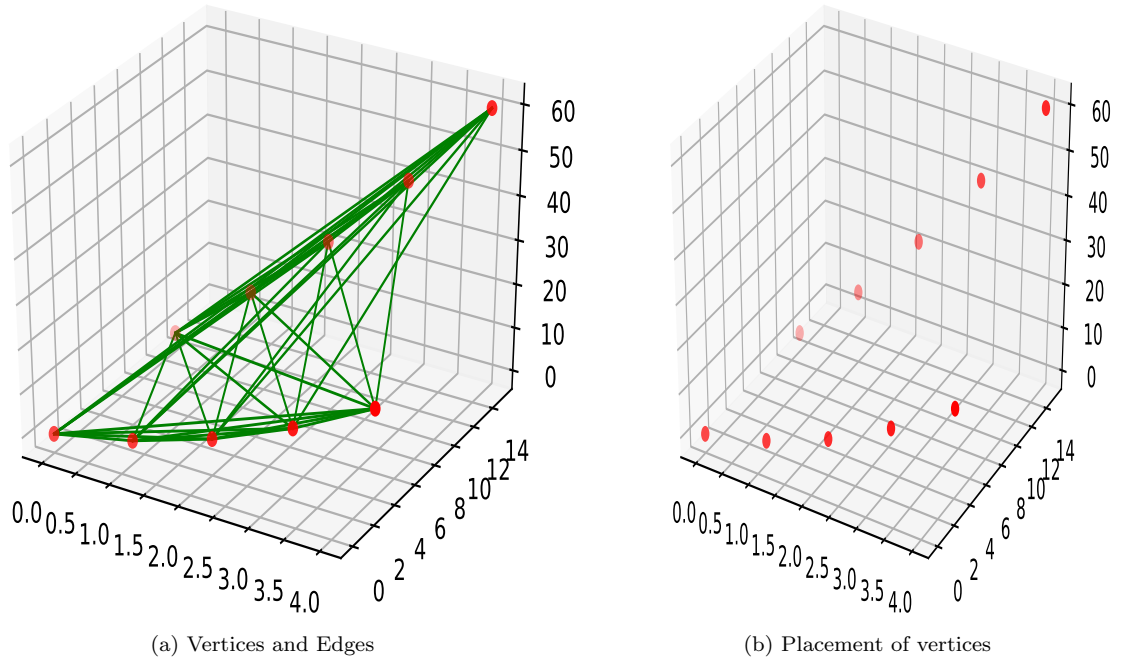


Figure 3.4:
3D Grid drawing for Bi-partite Graph

Total Volume of the above drawing is $(r \times 4n \times 4nr) = O(n^2r^2)$.

3.3 3-Track Layout Tree Drawing

A tree is a minimally connected graph. Each internal vertex is cut vertex, and every edge is cut edge. Depth of a tree d is the number of edges in the longest path from the root to the longest leaf. Following the 3D dimensional grid drawing convention, a tree can be drawn with three tracks [18].

$$V_i = \{(i, t, it) : d = i\%3, t \equiv i^2\%3\}$$

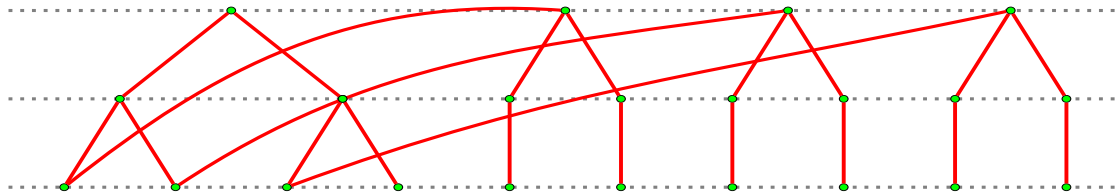
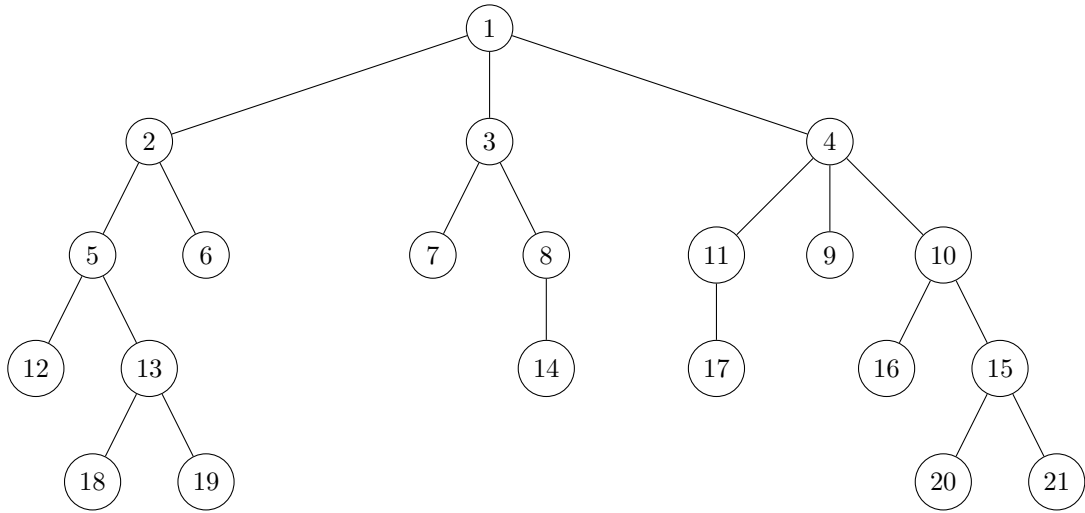


Figure 3.5:
Edge Wrapping into the 3-tracks



- GREEN: Edges from Track0 to Track1
- BLACK: Edges from Track1 to Track2
- ORANGE: Edges from Track2 to Track0
- Track0 Vertices
- Track1 Vertices
- Track2 Vertices

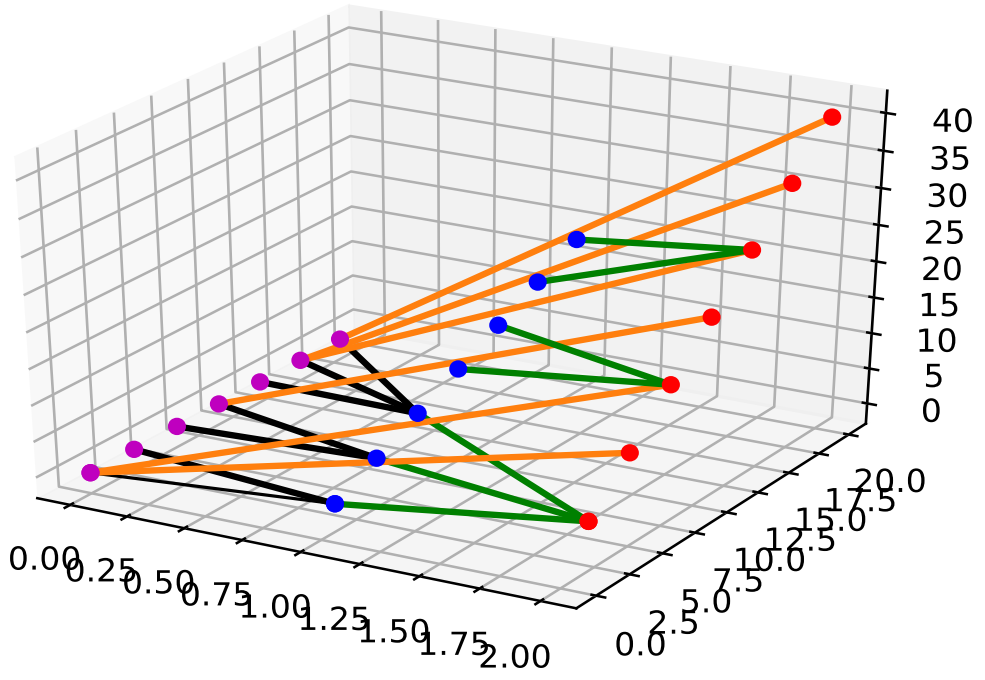


Figure 3.6:
3D Grid drawing for Tree

The total Volume of the above drawing is $(r \times 4n \times 4nr) = O(n \times t \times t) = O(t^2n) = O(n)$.

Chapter 4

Stack and Queue layout

4.1 Basics of layout

4.1.1 Layouts of Fixed Order

Let G be a Graph with Vertices $V_1, V_2, V_3 \dots V_n$ and corresponding edges $a_1 b_1, a_2 b_2 a_3 b_3 \dots a_n b_n$. There are some edges in Graph G , which may create issues when we need to minimize stack/Queue number.

A k -rainbow [24] is

$$e_p = (v_p w_p), 1 \leq p \leq n$$

such that

$$v_1 < v_2 < v_3 \dots v_{(n-1)} < w_n < w_{n-1} \dots w_2 < w_1$$

.

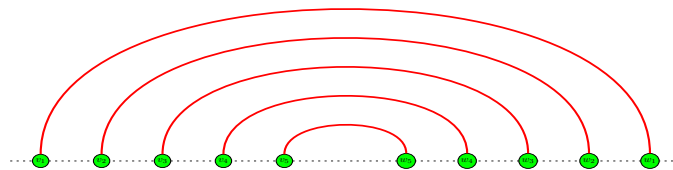


Figure 4.1:
 k - rainbow

A k -twist [25] is

$$e_p = (v_p w_p), 1 \leq p \leq n$$

such that

$$v_1 < v_2 < v_3 \dots v_{n-1} < v_n < w_1 < w_2 \dots w_{n-1} < w_n$$

.

K rainbow creates an issue for Queue Layout because it creates nested edges which cannot be placed

in the same queue. If graph G has k -rainbow, then Queue Layout of that graph cannot be less than K .

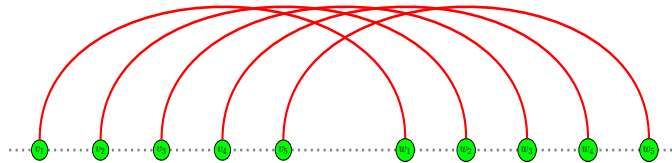


Figure 4.2:
 k -twist

K twist creates an issue for Stack Layout because it creates intersecting edges which cannot be placed in the same stack. If a Graph G has K -twist, then the Stack Layout cannot be less than k . The largest size of k -rainbow and k -twist in a graph G decides Queue number and Stack number of G respectively.

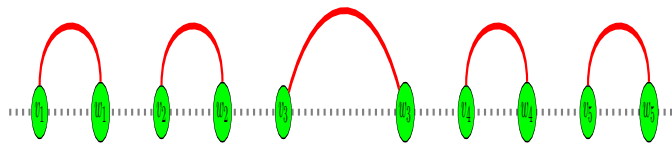


Figure 4.3:
 k -Necklace

4.1.2 Graph with Queue number 1 and stack number 1

A graph G is arc planar if and only if the number of Queue in Queue Layout is 1. A Graph G is outer-planar if and only if the number of the stack in stack Layout of Graph G is 1.

4.1.3 Queue number and Stack number Trade off

Let $1, 2, 3, \dots, n$ be a vertex ordering of Graph G . If we cut the above ordering orthogonality then the maximum number of edges that affected is called cut width. Cut width is the maximum of all cut sets. The valence of a graph G is defined as maximum degree of any vertex in a Graph G . Let $1, 2, 3, \dots, n$ be a vertex ordering of a Graph G , then the relation between Queue number and Stack number can be summarized as

$$\text{Stack number} \times \text{Queue number} \geq \frac{\text{Cut width}}{\text{Valence of } G}$$

4.2 Queue number and Stack number in a nutshell

A stack is a data structure which follows Last In First out Policy, where as Queue is a data structure which follows First in First out Policy. The stack layout is the minimum number of stacks needed to draw a linear ordering of vertices such that there is no X crossing as well as nested edges within the same stack.

Let endpoints of edges be denoted as L,R and there are two edges x,y ,

- if $Lx <_p Ly <_p Rx <_p Ry$, then x and y will intersect. This is called X crossing
- if $Lx <_p Ly <_p Ry <_p Rx$, then x and y will intersect. This is called $Nest$ crossing.

The Queue layout is the minimum number of the queue needed to draw a linear ordering of vertices such that there is no X crossing as well as nested edges within the same queue.

Let endpoints of edges be denoted as L,R and there are two edges x,y ,

- if $Lx <_p Ly <_p Rx <_p Ry$, then x and y will intersect. This is called X crossing
- if $Lx <_p Ly <_p Ry <_p Rx$, then x and y will intersect. This is called $Nest$ crossing.

Every Queue Layout graph can be converted into Track Layout Graph, but the reverse may not be true. For every bounded width graph, the track number is constant.

Chapter 5

Outer-planar Graph

In the planar graph, if every vertex is at the outer boundary, then it is called outer-planar Graph [26]. The maximal outer-planar graph is an outer-planar graph in which if we add an edge it destroys the property of outer-planarity. For n vertex, outer-planar graph the total number of edges are $2N - 3$. Every face of the outer-planar graph is bounded by exactly three edges or a triangle. If a graph G is outer-planar, then every bi-connected component of G is also outer-planar. Any maximal outer-planar graph can be implemented in a way such that every adjacent vertex (vertices connected by an edge) are placed at a vertical distance at most 2. Every vertex of the outer-planar graph lies on the outer boundary so if we delete a vertex and its connecting edges, then the resultant graph is also outer-planar. Every outer-planar graph has at least two vertices with the degree 2.

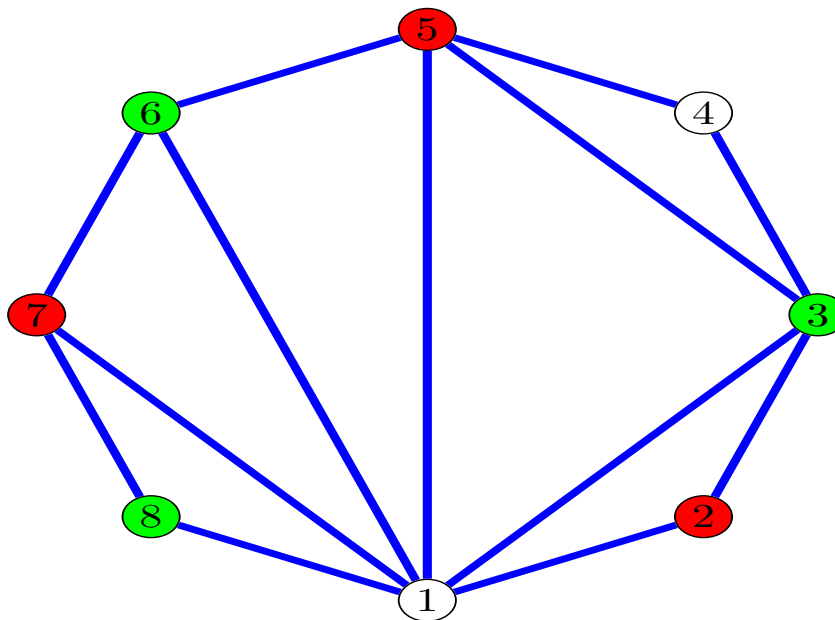


Figure 5.1:
Maximal Outer Planar Graph

5.1 Implementation of outer-planar Graph

5.1.1 Terminology

1. Upper Envelope: The part of a graph that is visible from $(0, +\infty)$.
2. Lower Envelope: The part of a graph that is visible from $(0, -\infty)$.
3. Co-ordinates of end point of an edge uv are $(X(u), Y(u))$ and $(X(w), Y(w))$.

5.1.2 Procedure

To construct a maximal outer-planar graph with neighboring vertices at height at-most two, recursively delete vertices which are having degree 2 and add it to the graph. Here the only concern is the vertical distance. For any face uw where $Y(u) < Y(w)$, v can be placed at following places:

- If uw is the face visible from $(0, +\infty)$ $Y(w) = Y(u) + 1$, v should be placed at $(\frac{1}{2}X(w) + \frac{1}{2}X(u), Y(w) + 1)$.
- If uw is the face visible from $(0, +\infty)$ $Y(w) = Y(u) + 2$, v should be placed at $(\frac{3}{4}X(w) + \frac{1}{4}X(u), Y(w) + 1)$.
- If uw is the face visible from $(0, -\infty)$ $Y(w) = Y(u) + 1$, v should be placed at $(\frac{1}{2}X(w) + \frac{1}{2}X(u), Y(u) - 1)$.
- If uw is the face visible from $(0, -\infty)$ $Y(w) = Y(u) + 2$, v should be placed at $(\frac{3}{4}X(w) + \frac{1}{4}X(u), Y(u) + 1)$.

5.2 Algorithm

As procedure mentioned above, below is an algorithm[19] for drawing maximal outer planar graph in which adjacent vertices are separated by the vertical distance at most two. This algorithm will take a maximal outer planar graph as input and return a maximal outer planar graph having property that every pair of neighbouring vertices at a vertical distance at most two.

Algorithm 1 Outer Planar Graph: Adjacent vertices are at a vertical height at most two.

```
1:  $G$  : Outer Planar Graph
2:  $G'$  : Null Graph
3:  $V_i$  : Set of vertices
4: procedure DRAW( $V_i, G$ )
5:   Draw a triangle in  $G'$ 
6:   for  $i = 1$  to  $n - 4$  do
7:     Randomly select a triangular region  $\Delta$  in  $G$ 
8:     Randomly select a point inside  $\Delta$ 
9:     Add edge between  $V_i$  to each vertices of  $\Delta$ 
10:  end for
11:  Return  $G'$ 
12: end procedure
```

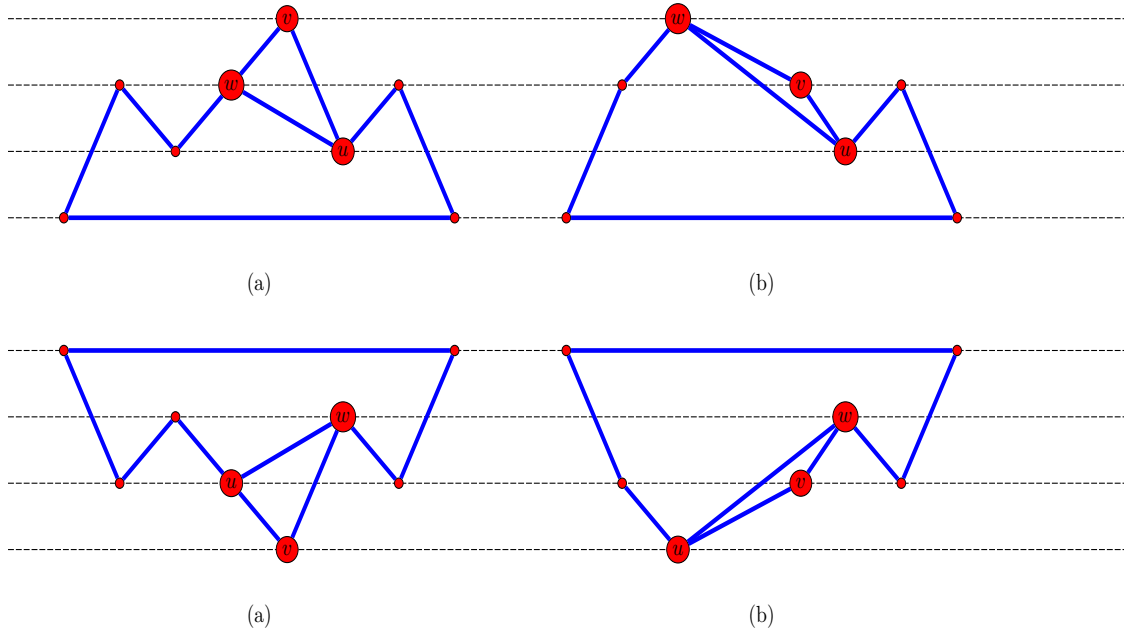


Figure 5.2: Upper envelop and Lower envelop

Every outer-planar graph can only be bounded by a triangle so adding a vertex v to the already existing outer-planar graph G will not affect outer-planarity.

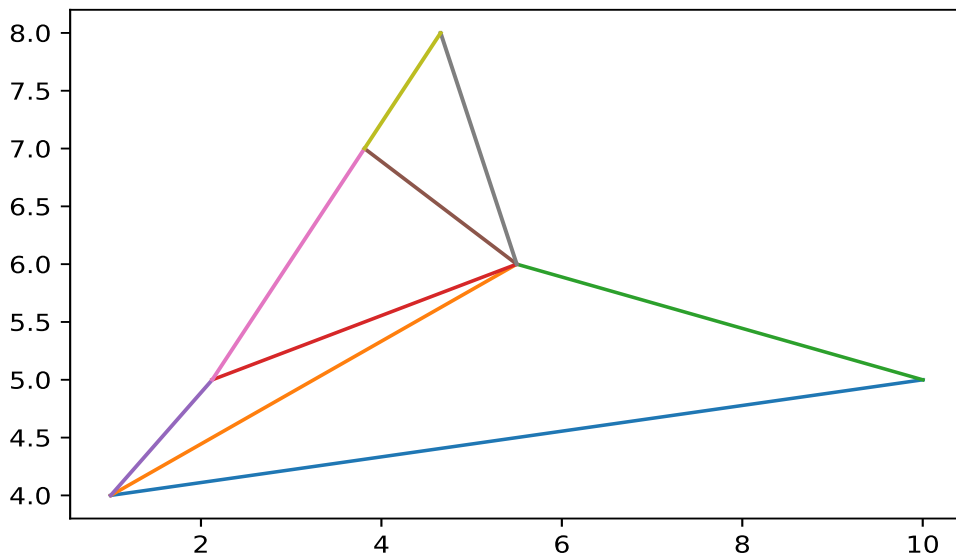
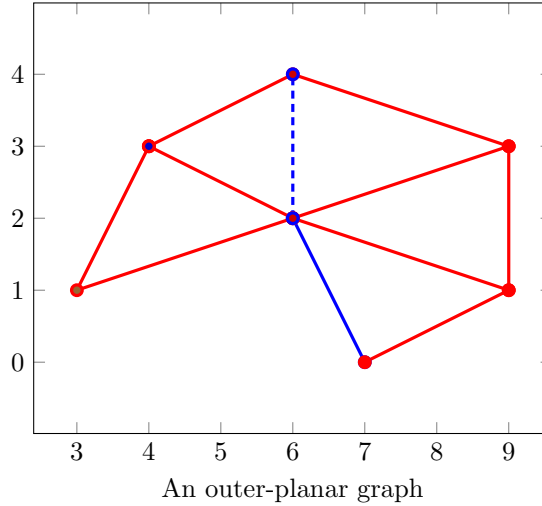


Figure 5.3:
Graph with height of neighboring vertices differ by at most 2

5.3 Track layout of outer-planar Graph

Every maximal outer-planar graph can be implemented in 5 tracks [27]. To prove this, we can use the relation between track-layout and span. If a graph has a maximum span of s then the number of track in the track layout of G can be represented by the following inequality:

$$t \leq 2s + 1$$



Let us say we have two vertices P_1 and P_2 whose coordinates are (x_{p1}, y_{p1}) and (x_{p1}, y_{p1}) . If there is an edge between P_1 and P_2 , then $|y_{p1} - y_{p1}| \leq 2$. So in the worst case a vertex $V : (x_{p1}, y_{p1})$ can be a neighbour of $V : (x_{p1}, y_{p1} + 2)$ and $V : (x_{p1}, y_{p1} - 2)$. The total span of y coordinates for any vertex is five here. We can use *Wrapping Lemma* and simulate any maximal outer-planar graph with five tracks.

5.4 2-Queue Layout of outer-planar Graph

Queue Layout of an outer-planar graph is a total ordering of vertices such that no two independent edges create a rainbow. The motive of queue layout is partitioning of the edges into the queues. G is a 1-stack graph if and only if G is outerplanar [28]. Every 1-stack graph can be simulated with 2-queue. So, the queue number of outerplanar graph is 2. The total span of maximal outerplanar is 2. This fact is also forcing an outerplanar graph to has queue number two.

Chapter 6

Planar 3-Tree

6.1 Definition

A graph is said to be planar such that edges in the drawn graph which lies in the same plane so that it does not intersect each other. A planar graph divides the plane into bounded and unbounded regions. The unbounded region is called the outer face, and the bounded region is called inner faces. The vertices which lie on the outer faces are called outer vertices, while the vertices which lie inside the inner faces are called inner vertices. A bounded region bounded by three edges is called a triangular region. If every face is bounded by three edges is called a triangulated graph [29]. Every outer-planar graph is internally triangulated. A maximum outer-planar graph is internally triangulated outer-planar graph with the maximum number of edges [30]. A layer i in a planar 3-tree is the graph after removing vertices and its connecting edges at layer $i - 1$.

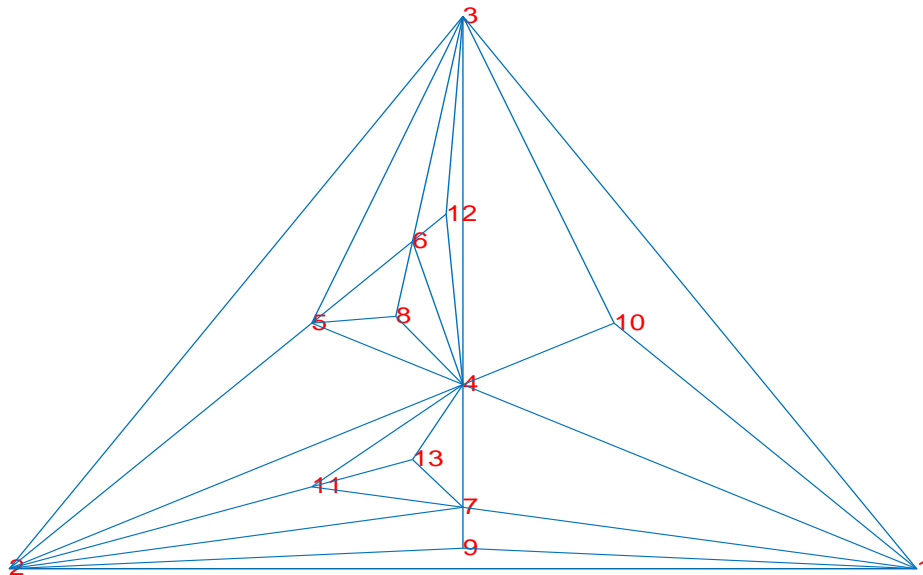


Figure 6.1:
Planar 3-tree

6.2 Implementation of planar 3-tree

Here is the algorithm for planar 3-tree.

Algorithm 2 Planar 3-Tree

```

1:  $G$  : Null Graph
2:  $V_i$  : Set of vertices
3: procedure DRAW( $V_i, G$ )
4:   Draw a triangle  $G$ 
5:   for  $i = 1$  to  $n - 4$  do
6:     Randomly select a triangular region  $\Delta$  in  $G$ 
7:     Randomly select a point inside  $\Delta$ 
8:     Add edge between  $V_i$  to each vertices of  $\Delta$ 
9:   end for
10:  Return  $G$ 
11: end procedure

```

The total number of vertices is n . Draw a triangle. Now, we have $n - 3$ edges. Select any bounded region and place vertex V_i anywhere in that region. Connect every vertex of that region with V_i . Previously the region was a triangle. After placing vertex V_i in that region, whole region becomes a set of the triangle where every region is at the outer boundary for region Δ_{123} .

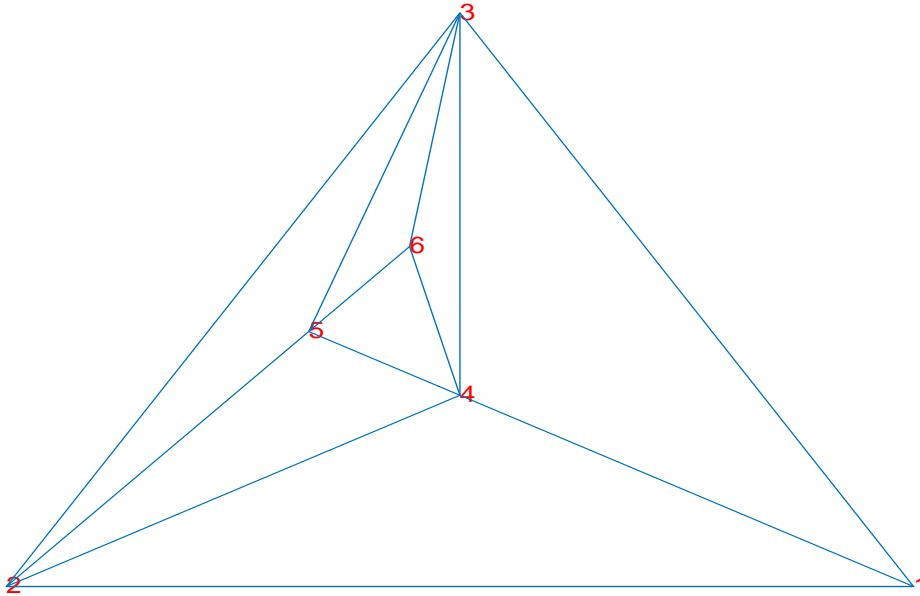


Figure 6.2:
2-Level Planar 3-Tree

In the above figure we had region Δ_{123} . After placement of vertex 4 Δ_{123} becomes $\Delta_{124}, \Delta_{143}, \Delta_{243}$. For the next vertex, region Δ_{243} was selected and vertex 5 was placed there. After placement of vertex 5, Δ_{243} becomes $\Delta_{245}, \Delta_{253}, \Delta_{453}$. Same for vertex 6. Placement of each vertex divides a Δ into 3, Δ and each new region placed at the outer boundary corresponding to their parent Δ . Vertices $\{1, 2, 3\}$ are at the outer boundary so these vertices are at level 0. If we remove these outer

vertices and its connecting edges, then vertices $\{4, 5, 6\}$ will lie on the new outer boundary and this set of vertices are at level 1.

In fig 6.1, 3-Level Planar 3-Tree where the vertex set $\{1, 2, 3\}$ is at level 0, the vertex set $\{4, 5, 6, 7, 9, 10, 11, 12\}$ is at the level 1, vertex set $\{13, 8\}$ is at level 2. Every layer of planar 3-tree consist of internally triangulated outer-planar graph. Property of outer-planar graph like track number and queue number is useful for finding the Queue layout of planar 3-tree.

6.3 5-queue Layout of 2-layer Planar 3-Tree

6.3.1 Terminology

- **Layer:** Layer i of a planar 3-tree is the set of vertices which are at the boundary after removing all edges which are the level $i - 1$. The vertices of a triangle are said to be at layer 0.
- **Binding edges:** Binding edges are those edges which are connecting two layers of planar 3-tree.
- **Level edges:** Edges which are used to connect the vertices of the same layer is called level edges.
- **Anchor:** Let $\langle u, v, w \rangle$ is a triangle with neighbour vertices having maximum vertical height at-most 2. Let v and w are top and bottom vertices, then vertex u is called the anchor vertex of face $\langle u, v, w \rangle$.

6.3.2 Queue layout of 2-level planar 3-tree

Let graph G_0 is the graph obtained due to layer L_0 and graph G_1 is the graph obtained due to layer L_1 . We need to follow procedure [31]:

Algorithm 3 Queue layout of 2-Layer Planar 3-Tree

- 1: G : 2-level planar 3-tree
 - 2: L_i : Set of vertices at Layer i
 - 3: Q_i : Set of 5-queues
 - 4: **procedure** 5-QUEUE LAYOUT(G, L_i, Q_i)
 - 5: Total ordering of vertices such that vertices which are at layer L_0 placed before vertices of layer L_1
 - 6: Placement of edges of same layers can be done with the help of one queue each. For layer L_0 we need 1 Queue and another queue for layer 1. Total 2-Queues
 - 7: Three additional queues needed for the edges which are connecting two successive layers i.e. layer L_0 and layer L_1
 - 8: Return Q
 - 9: **end procedure**
-

2 Queues for level edges:

Place vertices of L_0 and L_1 in a total order such that $L_0 < L_1$. For each layer order of vertices would be decided by the property of the triangulated outer-planar graph. Every triangulated outer-planar graph can be converted into a graph such that neighboring vertices differ by the vertical height at most 2. Two queues are sufficient to simulate a triangulated outer-planar graph. If there are multiple connected components at any level, then each connected component either precede or succeed. In that case, we use the previously used queue for them also.

3 Queues for binding edges:

Let L_1 having a face $\langle u, v, w \rangle$ that is inscribed in L_0 . We will allocate top, bottom and anchor vertex to different queues. If L_1 having more than one connected component let's say, c_1 and c_2 , then find ordering between these two and then-then place in into the queues.

Finding Ordering:

Let's say, L_1 has two connected components $c_1 = \langle u, v, w \rangle$ and $c_2 = \langle u', v', w' \rangle$ inscribed in L_0 . u, u' are anchor vertices, v, v' are top and w, w' are bottom vertices of connected components c_1 and c_2 .

1. If the position of anchor vertex is not same i.e. uu' then either c_1 precedes or succeed depending upon the relative positions of u and u' . If $u \prec u'$ then c_1 precedes c_2 and place c_1 in the queue before C_2 in partial order of L_1 .
2. If the position of anchor vertex is same i.e. $u = u'$ then depending upon the relative positions of v and v' either c_1 precedes or succeed. If $v \prec v'$ then c_1 precedes c_2 and place c_1 in the queue before c_2 in partial order of L_1 .
3. If the position of anchor vertex is same i.e. $u = u'$ and $v = v'$ then depending upon the relative positions of w and w' either c_1 precedes or succeed. If $w \prec w'$ then c_1 precedes c_2 and place c_1 in the queue before C_2 in partial order of L_1 .

Proof that no two edges of the same queue nested:

Let's say two edges e_1, e_2 whose end vertices at L_0 and L_1 are $(p, q), (p', q')$ respectively and are placed in the same queue. Anchor vertices $p, p' \in L_0$ with the face Δ and $q, q' \in L_1$ with face Δ' . If $p \prec p'$ then Δ should be placed before Δ' . If all edges of Δ placed before every edge of Δ' that means there will not be any nesting edges. We can use the same analogy for top and bottom vertices also. For avoiding nesting edges we need three queues. One each for:

- Anchor vertex
- Top vertex
- Bottom vertex

6.3.3 Implementation

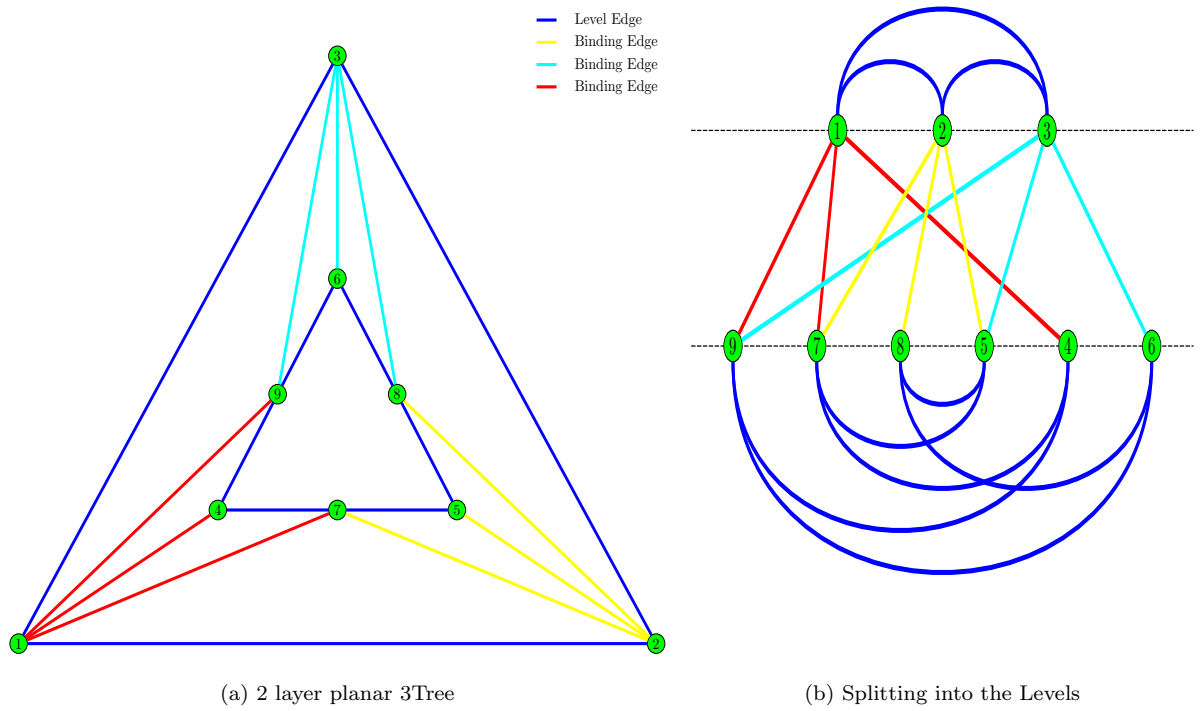


Figure 6.3:
Planar 3–Tree and level split

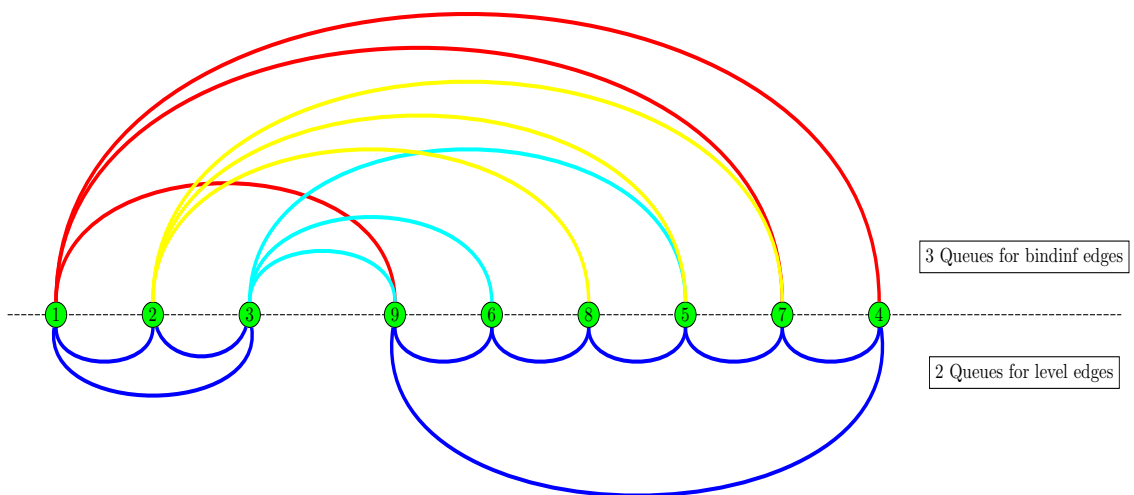


Figure 6.4:
5– Queue layout of Planar 3–Tree

6.4 5 Queue layout of n -layer Planar 3-Tree

We have proved that five queues are sufficient enough to simulate 2-layer planar 3-tree. For a graph G having n - layers $L_0, L_1, L_2, \dots, L_n$. Each layer L_i is a set of internally triangulated outer-planar connected components. These connected components may be bi-connected so add extra edges in order to make it bi-connected. For each layer i the graph G_i must be inscribed in G_{i-1} or vice versa. Edges connecting vertices of the same layer are called level edges and edges connected connecting the vertices of consecutive layers are called binding edges. Algorithm[31] for placement of edges into the 5-queues for n - layer planar 3- Tree ($n > 2$):

Algorithm 4 Queue layout of n -Layer Planar 3-Tree

```

1:  $G$  : 2-level planar 3-tree
2:  $L_i$  : Set of vertices at Layer  $i$ 
3:  $Q$  : Set of 5-queues
4: procedure 5-QUEUE LAYOUT( $G, L, Q$ )
5:   for  $i = 0$  to  $n - 1$  do
6:     Total ordering of vertices such that vertices which are at layer  $L_i$  placed before vertices
       of layer  $L_{i+1}$ 
7:     Placement of edges of two consecutive layers can be done with the help of one queue
       each. For layer  $L_i$  we need 1 Queue and for layer  $L_{i+1}$  another queue needed. Total
       2-Queues
8:     Three additional queues needed for the edges which are connecting two successive layers
       i.e. layer  $L_i$  and layer  $L_{i+1}$ 
9:      $i = i + 1$ 
10:  end for
11:  Return  $Q$ 
12: end procedure

```

For each level pair L_i and L_{i+1} , we first linearly place the vertices of each layer such that they will follow total order. Ordering of vertices is decided by layer number. We need two queues for level edges and three queues for binding edges. Once we are done with layer pair L_i and L_{i+1} , the same procedure will be applicable to layer pair L_{i+1} and L_{i+2} . Here the queue used for level edges of layer i can be reused for level edges of layer $i + 2$. Binding edges also follow the same rule.

No Nesting Edges in same queue

Lets say two edges e_1, e_2 whose end vertices at L_i and L_{i+1} are $(p, q), (p', q')$ respectively and are placed in same queue. Anchor vertices $p, p' \in L_i$ with face Δ and $q, q' \in L_{i+1}$ with face Δ' . If $p \prec p'$ then Δ should be placed before Δ' . If all edges of Δ placed before every edge of Δ' that means there will not be any nesting edges. We can use the same analogy for top and bottom vertices also. Finally, we need three queues. One each for:

1. Anchor vertex
2. Top vertex
3. Bottom vertex

and there will not be any nesting edges or k -rainbow.

6.5 Result

6.5.1 n layer 3-tree

Here is the 3-layer planar 3-tree. Blue edges are level edges and violet edges are binding edges. We will first place this graph in multiple levels. Vertex 1 is at level 1. If we delete the vertex 1 then the next level vertex set is 2, 3, 6, 10.

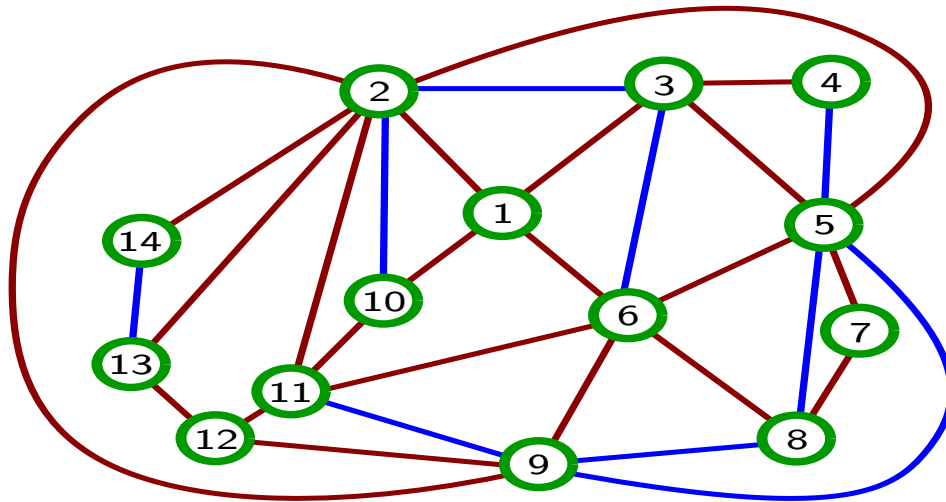


Figure 6.5:
 n - layer Planar 3-Tree

6.5.2 Converting into levels

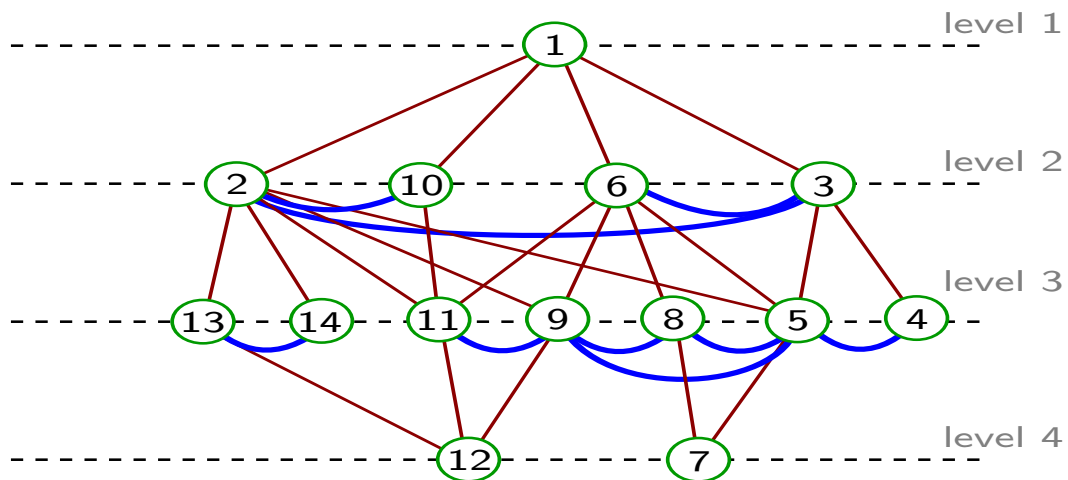


Figure 6.6:
Level wise placement of vertices of planar 3-tree

6.5.3 Queue layout

At level 0 only one vertex 1 is there. So no level edge for L_0 . At Level 1 there are four vertices 2, 3, 6, 10. Create a total order of vertices and then place on a number line. We have 5 queues Q_0, Q_1, Q_2, Q_3, Q_4 . Queues Q_0, Q_1 for level edges and Q_2, Q_3, Q_4 for binding edges.

For Level L_0 no level edge for Q_0 would be empty and for L_1 all edges $2 - 10, 2 - 3, 6 - 3$ placed in Q_1 . Binding edges uses 3 queues and edges $1 - 3, 3 - 6$ are placed in Q_2 , edges $1 - 2, 1 - 6, 1 - 10$ are placed in Q_3 , edges $1 - 3, 3 - 2, 10 - 2$ are placed in Q_4 . Once this process is completed Q_0 can be used for level L_2 and binding edges can be reused accordingly. Here is implementation:

Level Edges are blue colored, binding edges are violet coloured and the vertices inscribed in the grey coloured area is the set of vertices at the same level.

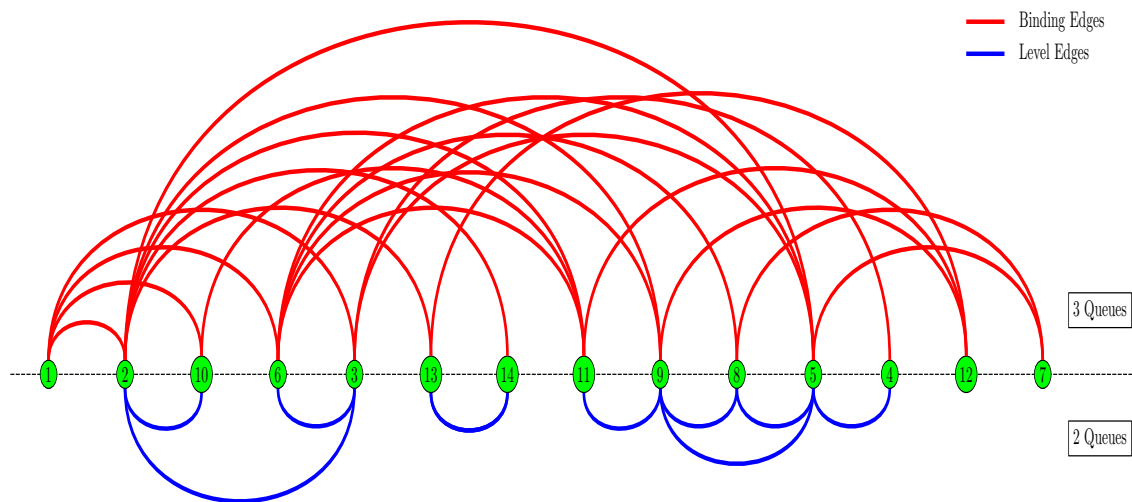


Figure 6.7:
Queue layout of Planar 3-Tree

Chapter 7

Conclusion and Future work

7.1 Summary

Queue Layout		
Graph Class	Lower Bound	Upper Bound
Tree	1	1
Outer-Planar	2	2
Planar 2–tree	3	3
Planar 3–tree	4	5
Planar	4	$O(\log n)$

Implementation of an outer-planar graph was the basis of finding queue layout of planar 3-tree. Any maximal outer-planar graph can be converted into outer-planar graph with neighboring vertices kept at the vertical distance at most two. This result helps in finding track layout and queue layout of outer-planar graph. Every planar 3-tree consist of internally triangulated outer-planar graphs. Each layer of planar 3-tree is set of connected component of triangulated outerplanar graph.

The queue layout of a tree, outer-planar graph and planar 2-tree is 1, 2, 3 respectively. Previous upper and lower bound of planar 3–tree was 3 and 7 respectively. We reduced the upper bound to 5 and there exist a planar 3–tree whose lower bound is 4. This result will help us to find queue layer out of any planar graph. Till now, the lower bound of the queue layout planar graph is 4 and upper bound is $O(\log n)$. These bounds are still not tight. Finding queue layout of planar graph is still an open problem. There are paper which shows that if tree width of graph is bounded by some constant then the requirement of the number of queues in queue layout of planar is also bounded by some constant. Finding queue layout of queue layout is an open optimization problem.

Bibliography

- [1] Sue H Whitesides. *Graph Drawing: 6th International Symposium, GD'98 Montreal, Canada, August 13-15, 1998 Proceedings*. Number 1547. Springer Science & Business Media, 1998.
- [2] Fabian Beck, Michael Burch, and Stephan Diehl. Towards an aesthetic dimensions framework for dynamic graph visualisations. In *2009 13th International Conference Information Visualisation*, pages 592–597. IEEE, 2009.
- [3] Helen C Purchase. Metrics for graph drawing aesthetics. *Journal of Visual Languages & Computing*, 13(5):501–516, 2002.
- [4] Goos Kant. Drawing planar graphs using the canonical ordering. *Algorithmica*, 16(1):4–32, 1996.
- [5] Edward M. Reingold and John S. Tilford. Tidier drawings of trees. *IEEE Transactions on software Engineering*, (2):223–228, 1981.
- [6] Pierluigi Crescenzi, Giuseppe Di Battista, and Adolfo Piperno. A note on optimal area algorithms for upward drawings of binary trees. *Computational Geometry*, 2(4):187–200, 1992.
- [7] Pierluigi Crescenzi and Adolfo Piperno. Optimal-area upward drawings of avl trees. In *International Symposium on Graph Drawing*, pages 307–317. Springer, 1994.
- [8] Timothy M Chan. A near-linear area bound for drawing binary trees. *Algorithmica*, 34(1):1–13, 2002.
- [9] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G Tollis. Algorithms for drawing graphs: an annotated bibliography. *Computational Geometry*, 4(5):235–282, 1994.
- [10] Jürgen Branke. Dynamic graph drawing. In *Drawing graphs*, pages 228–246. Springer, 2001.
- [11] Peter Eades and Lin Xuemin. How to draw a directed graph. In *[Proceedings] 1989 IEEE Workshop on Visual Languages*, pages 13–17. IEEE, 1989.
- [12] Emden R Gansner, Eleftherios Koutsofios, Stephen C North, and K-P Vo. A technique for drawing directed graphs. *IEEE Transactions on Software Engineering*, 19(3):214–230, 1993.
- [13] Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiro Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(2):109–125, 1981.

- [14] Roberto Tamassia and Ioannis G Tollis. Planar grid embedding in linear time. *IEEE Transactions on circuits and systems*, 36(9):1230–1234, 1989.
- [15] Roberto Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM Journal on Computing*, 16(3):421–444, 1987.
- [16] David R Wood. Queue layouts, tree-width, and three-dimensional graph drawing. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 348–359. Springer, 2002.
- [17] Lenwood S Heath, Frank Thomson Leighton, and Arnold L Rosenberg. Comparing queues and stacks as machines for laying out graphs. *SIAM Journal on Discrete Mathematics*, 5(3):398–412, 1992.
- [18] Emilio Di Giacomo and Henk Meijer. Track drawings of graphs with constant queue number. In *International Symposium on Graph Drawing*, pages 214–225. Springer, 2003.
- [19] Vida Dujmovic, Attila Pór, and David R Wood. Track layouts of graphs. *Discrete Mathematics and Theoretical Computer Science*, 6(2), 2004.
- [20] E Sampathkumar and VN Bhave. Partition graphs and coloring numbers of a graph. *Discrete Mathematics*, 16(1):57–60, 1976.
- [21] Hubert De Fraysseix, János Pach, and Richard Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990.
- [22] Robert F. Cohen, Peter Eades, Tao Lin, and Frank Ruskey. Three-dimensional graph drawing. *Algorithmica*, 17(2):199–208, 1997.
- [23] János Pach, Torsten Thiele, and Géza Tóth. Three-dimensional grid drawings of graphs. In *International Symposium on Graph Drawing*, pages 47–51. Springer, 1997.
- [24] Boštjan Brešar, Michael A Henning, Douglas F Rall, et al. Rainbow domination in graphs. *Taiwanese Journal of Mathematics*, 12(1):213–225, 2008.
- [25] Indra Rajasingh, Bharati Rajan, and R Sundara Rajan. Embedding of hypercubes into necklace, windmill and snake graphs. *Information Processing Letters*, 112(12):509–515, 2012.
- [26] Prosenjit Bose. On embedding an outer-planar graph in a point set. *Computational Geometry*, 23(3):303–312, 2002.
- [27] Franz Brandenburg, David Eppstein, Michael T Goodrich, Stephen Kobourov, Giuseppe Liotta, and Petra Mutzel. Selected open problems in graph drawing. In *International Symposium on Graph Drawing*, pages 515–539. Springer, 2003.
- [28] Lenwood S Heath and Arnold L Rosenberg. Laying out graphs using queues. *SIAM Journal on Computing*, 21(5):927–958, 1992.
- [29] Donald J Rose. Triangulated graphs and the elimination process. *Journal of Mathematical Analysis and Applications*, 32(3):597–609, 1970.
- [30] Giuseppe Di Battista, Fabrizio Frati, and Janos Pach. On the queue number of planar graphs. *SIAM Journal on Computing*, 42(6):2243–2285, 2013.

- [31] Jawaherul Md. Alam, Michael A. Bekos, Martin Gronemann, Michael Kaufmann, and Sergey Pupyrev. Queue layouts of planar 3-trees. In Therese Biedl and Andreas Kerren, editors, *Graph Drawing and Network Visualization*, pages 213–226, Cham, 2018. Springer International Publishing.