GRAPH VISUALIZATION

Pravi Malviya

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

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Under the guidance of
Dr. N.R.ARAVIND

Deparment Of Computer Science and Engineering
I.I.T. Hyderabad

June 2016

Declaration

I, Pravi Malviya, declare that this thesis titled, 'Graph Visualization' and the work presented in it are my own. I confirm that this work was done wholly or mainly while in candidature for a thesis research.Any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.I have consulted the published work of others, this is always clearly attributed.I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.I have acknowledged all main sources of help.Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

[illegible] Signature:

PRAVI MALVIYA Name:

CS14MTECH11012 Roll No.:

Approval Sheet

This Thesis entitled Graph Visualization by Pravi Malviya is approved for the degree of Master of Technology from IIT Hyderabad

[illegible] 15/6/2016

(U.RAMAKRISHNA) Examiner Dept. of CSE IITH

[illegible] 15/6/16

Examiner Dept. of CSE IITH

[illegible]

(Dr. N.R. Aravind) Adviser Dept. of CSE IITH

[illegible]

SUBRAHMANYAM KALYANASUNDARAM Chairman Dept. of CSE IITH

Acknowledgements

I would like to express my deepest appreciation to all those who provided me the possibility to complete my thesis. A special gratitude to my thesis adviser Dr. N.R.Aravind for his constant encouragement, patience and immense knowledge. Without his guidance and persistent help, this dissertation would not have been possible.I thank the Almighty, my family and friends for their support and constant encouragement.

Dedication

This work is dedicated to my parents and all of my friends without whom none of my success would be possible.

Abstract

Visualization of graph is the most important area. It helps in numerous applications like in web computing,database handling, machine learning stuff and many more,also it provides a way of representing data, showing large scale information, analysing the data, communication with peers and people. Visualization of graph creates a problem when nodes are very large in number and the area to visualize all of them is less also when the graph is small even and if the graph is non-planar or having cycle in it create problems.

We implemented a scenario to obtain a planar two-dimensional subgraph view of non-planar graph showing two-color class nodes at a time and each set of two-color class node will surely be a planar representation.

Also for the visualization of graph in three-dimension we have work on some class of graph which are having geometric thickness as two, and some of the graph which can be plotted with maximum count of edges 6N-19 where N resemble number of nodes (having geometrical thickness two).We have come up with a drawing where number of edges could be increase in the above scenario.

Contents

Chapter 1

Introduction

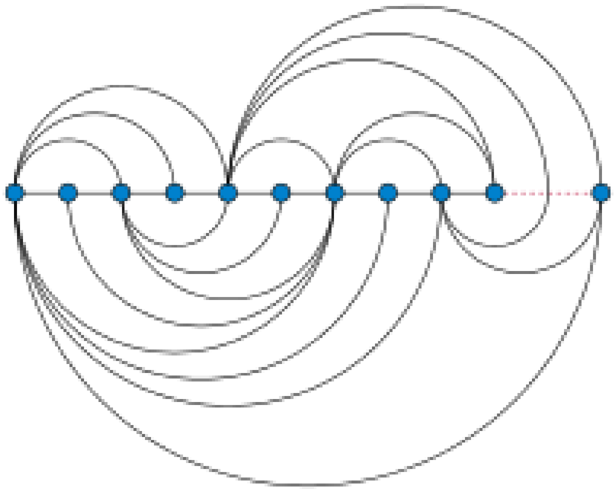
1.1 Graph Visualization

It is a part of visualization, the visualization of data can be performed on/for different purposes and one of them is visualization of graph. Visualization refers to technique for creating images, diagrams and graphs to deliver a message. Many of the data/information cannot be explained verbally it is clearly explainable when they are viewed in form of some images or diagrams. Visualization helps in analysing the data, interactive exploration, presentation of data, probing into the problem domain, better understanding, communicating with the peers, communicating with people. The ultimate goal of visualization is to allow person to easily understand and share the data.

Graph visualization[1] is a way of representing structural information as diagrams of abstract graphs and networks. Graph visualization is the most important phase now a days. Several applications are related to it. Wide range of relations, network configuration, and relation among various patterns are shown on basis of it.

Graphs are widely used as a visualization tool[2] in many different domains for example networking, bio-informatics, web computing, information systems, software engineering, database and web design, machine learning, and in visual interfaces for other technical domains[3]. Visualization of graph is like a pictorial representation of the vertices and edges of a graph. This drawing should not be confused with the graph itself.

Different layouts[4] can correspond to the same graph. The drawing of graph is not unique. The view of the graph that is presented should not be confused with the graph itself. Various layout of a single graph could be created, the main motive is that graph must be visible properly and the nodes which are connected to each other must maintain their same relation without losing any relation among them.



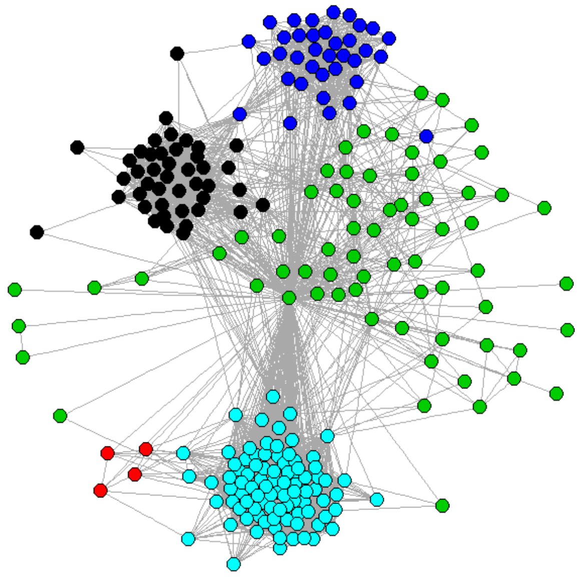


FIGURE 1.1: Different layout possible. source:wikipedia

There are different points that are to consider while visualization of graph as visualization of graph is not to perform only on some hundred nodes, but visualization is to be performed of millions of node in many cases. At different scenario different layouts are possible according to requirements and need. Visualization offers a method for seeing the unseen. It embellish the process of scientific discovery. Under the process of visualization data is transformed into some graphic form and rendered.

Graph drawing is concerned with the problem of rendering a given graph on the two-dimensional plane so that the resulting graph/drawing is as readable as possible. Not only this now a days three-dimensional view are possible by various technique as it provide better aspect of view to the graph. Various tools are available which directly accept the node-edge description of the given system of graph and provide us the view of graph, but the view obtained from the existing tools have various limitation in itself and work for specific conditions and scenarios only. Many algorithms are available which provide us the coordinates of the graph for given data set but they required pre-processed data for proper visualization.

Non-Planar graphs can also be view in the planar scenario by projecting the graph into more planes. Some of the edges of the graph can be view in some plane and other edges in the other planes, such that each plane of view of drawing will provide us the planar view of the non-planar graph.

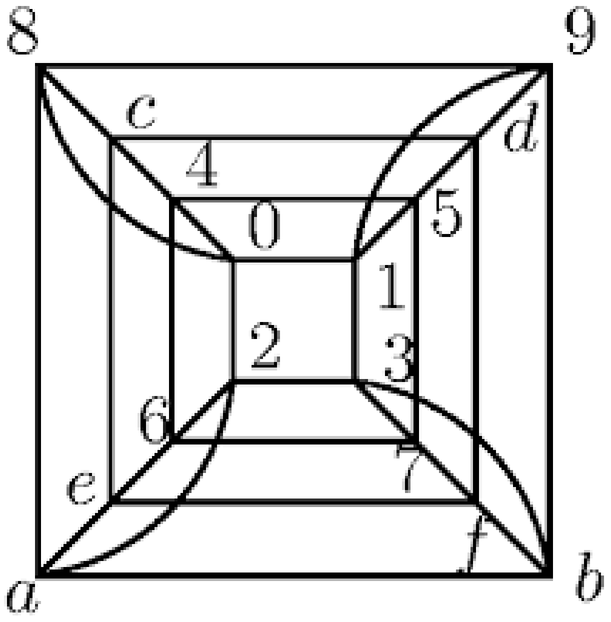
Thickness of graph[13] is an important parameter as thickness can help in determination of the minimum number of planes can be used for plotting the non-planar graph into number of planes such that each plane provide the planar view of the graph. Partition of edges into minimum number of planar sub-graphs. Some of the graph edges will be plotted in some plane and some of the other in some other plane such that each of the plane will give us planar view.

Thickness of graph can be measured as graphical thickness[15,17] or on the basis of geometrical thickness. According to graphical thickness it give the count of the minimum number of planar graphs into which a graph G can be decomposed also the location of each vertex in different plane can differ as well as arc can be drawn for edges. The geometric thickness[13,17] of a graph G,is to be the smallest value of k such that we can assign planar point locations to the vertices of G, represent each edge of G as a line segment, and assign each edge to one of k layers so that no two edges on the same layer cross, also in each plane the location of vertex should not be changed and all the edges connectivity must show with the straight line.

1.2 Problems in Visualization

When the number of nodes are less, the representation is easily understandable, but as the size increases the haziness appear in the graph. It depend upon the layout, that how the drawing is performed of the nodes as well as connecting edges, it could be possible that even a large size graph with thousands of nodes can be represented cleanly without any fuzzy nature and even some time because of the presentation a small graph could provide a vague view. It depend on the nature of graph as well as how the layout could be provided.

The problem arise even when the graph/structure provided is non-planar[5]. As when there is less number of node we can identify that which node/user is connected to which other node/user in most of cases but when the number of nodes are very large it is quite difficult in the non-planar graph to view and uniquely identify each and every connected edge in the system. As the edges of graph would be hazy, some of the edges may be overlapping, some of the edges would be so close to each other that we could not visualize that which edge is connecting to which node and many such problem would arise in the system.



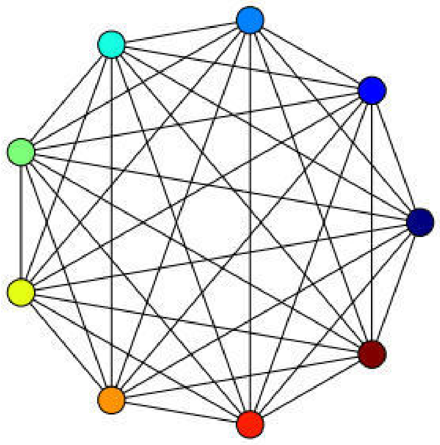


Figure 1.2: Visualization problems in graph. source:wikipedia

Similar condition also occurs in the cyclic graph, when the length of cycle will be small and also the number of nodes are less then there is not much issue but when the number of nodes as well as length of cycle increases the graph shows its fuzzy nature. The visualization under such scenario is not proper.

Above both cases can even occur in lesser number of nodes system also. As the edges would be in imbrication fashion and some nodes will be so close that we cannot differentiate between the ends of edges which are connecting to the node, so in that way we will not be able to identify the correct relation among them.

The readability of graph is dependent on its application, but our main aim here is to get the view of graph without edge crossing or with non-cyclic nature. Such a scenario is termed as planar viewing.

As every graph cannot be represented without edge crossing[5] or cyclic nature. So for the planar as well as non-cyclic view of graph we are implementing various algorithm so as to assign colors to node of the preprocessed graph and use the color set class of the nodes to visualize the graph in a planar view according to the requirement.

Another aspect of the visualization is to plot the non-planar graph into various planes but with certain class of graph certainty is present like in the K-number of plane only Y-number of edges can be plotted with occurrence of planar view in each plane, so as to resemble whole non-planar graph with planar visualization[16] in each plane is another problem.

Also till now for certain class of graph which is having geometrical thickness two maximum edges can be plot into two planes each plane having planar resemblance is 6N-19[17,18] , N refers to number of node/vertex($N >= 9$). So our aim is to plot more number of edges for above scenario of graph with certain modification in drawing under three-dimensional view.

1.3 Proposed Work

For the representation of non-planar and cyclic graphs we have proposed two methodology:

- To provide a planar view of non-planar/cyclic graph such that 2-color class node at a time can be viewed. (For each non-planar graph we assign color class such that for any two set of color class we will get a planar view of them).
- Next we work on visualization of non-planar graph (some complete and some bipartite graph) and represented them with a 3-Dimenesional view such that we

plotted them into two different planes, each plane is providing a planar view and each of them are projected into another plane.

Also we have work on class of graph which have geometrical thickness is 2. Maximum of $6N - 19$ ($N >= 9$) edges can be plotted into two plane such that each plane is having planar view for graph having thickness two. So we come with an approach of presenting the such graph into 3-Dimensional projection such that we could increase the number of edges while visualization.

Chapter 2

Shifting and Coloring Algorithms Implemented

For the better visualization of the graph different algorithms are available which help in providing proper sequence to the node of the graph so that they could be well visualized, hence with the help of algorithm proper visualization of graph is possible along with certain properties which also get satisfied that help in providing better visual aspect of the information.

So our idea is to visualize the graph such that we can obtained the graph as planar/non-cyclic with certain property holding. So we basically allot color to graph and visualize for two color class such that for that two set of color class the graph must necessarily planar/non-cyclic.i.e we will assign color to node and check for each set of two color class that it is perfectly non-cyclic/planar and when such assigned of color is obtained then we visualize the graph with specific 2-color class at a time.

Among various algorithms that we have gone through some of the algorithms are for providing the orthogonal representation, some of the algorithms used for the proper coloring sequence, some of them are used for determining whether given system is planar or not and some for verifying that given data set exist cyclic nature or not, some for verifying that given graph is tri-connected or not and how to modify it to create it into tri-connected. Some of them are specific to certain conditions of graph. Some of the algorithms which are implemented:

2.1 Planarity Checking

Kuratowski Algorithm: Provided a finite graph is planar if and only if it does not contain a subgraph that is homeomorphic to Pentatope Graph or Utility Graph[6]. A graph is planar if and only if it does not contain a sub-graph that is expansion of either Pentatope Graph or Utility Graph, both of the graphs are minimal examples of non-planarity within their class of graphs i.e. K_5 or $K_{3,3}$

For implementing it Boost Library of C++ is used, they provide the algorithm structure and when input is provided in specific format of list to it, it result in output as planar/non-planar.

For the given data it is verified that it's subgraph is a expansion of of either K_5 or $K_{3,3}$.

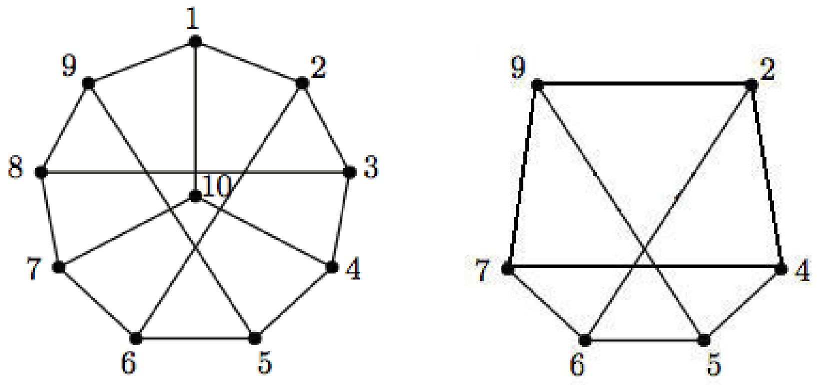


Figure 2.1: Finite graph verifying that graph is planar or not. source:wikipedia.

Above graph is non-planar as with the edge/node contradiction above scenario is formed that is of $K_{3,3}$.

2.2 Canonical Ordering and Shifting Algorithm

The next algorithm implemented is determination of canonical ordering[7],[8] with the visualization of the given graph on basis of this canonical ordering. When the given graph is finite and tri-connected with the help of taking each outer face and depending on the separation pair the ordering of each node is determine. As we get the required ordering of tri-connected graph then shift algorithm[9],[10] is applied which provide us the coordinates of the node on the basis of ordering and then that coordinates are used for drawing the line/edge between the nodes that are present in the graph structure.

Steps:

1. Determination of canonical ordering (boost library) of given edge list of graph.
2. From the obtained order shifting algorithm (canonical drawing and Manhattan distance) is applied and the coordinates of each node is store.
3. Again the given edge list is traversed, as edge is obtained we plot the edge/line on the set of node whose coordinates are already computed and stored.

For example: For following tri-connected system.

0 1; 0 2; 0 3; 1 2; 1 3; 2 3 (pair of number indicate that edge appear between the following nodes in the system).

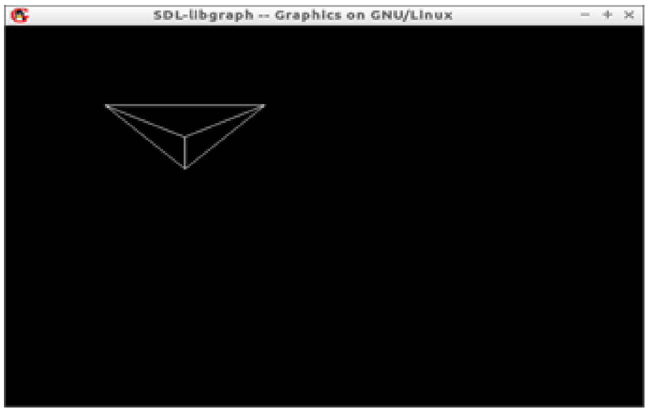


FIGURE 2.2: Output for the visualization of given data which is tri-connected.

2.3 Coloring Approach

2.3.1 Squaring the graph for coloring

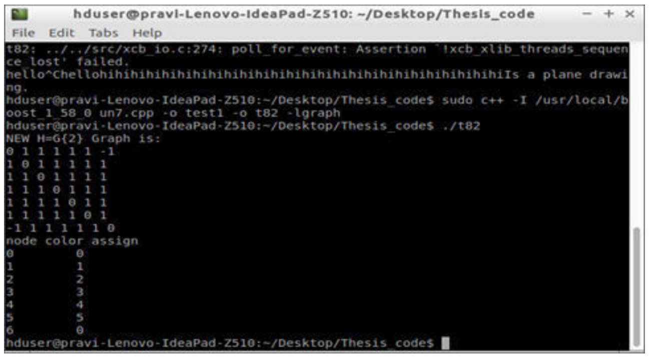
Under this implementation coloring is performed on the basis of color of neighbor. All the node are colored randomly i.e. all the nodes are assigned colored with randomness and stored and it is checked that whether the neighbor/connected node is having different color or not, if same color is found then again a new random color is allotted to node and then we further check for other node that each of the connected member is having different color or not, and this methodology is performed continuously unless all the neighbors are assigned with different colors among themselves. But this checking is done on the squaring of the graph[11] and color allotted is depend on the maximum degree of the new graph which is squared.

Steps:

1. For the pre-processing step first we will create a new graph with configuration like squaring of the graph. i.e. we will assume that the node which are at distance two from each node is also directly connected to the node, In this way new graph is created.
2. Further we will determine the degree of the preprocessed graph and we will determine the maximum degree of the graph.
3. Max allotment of color to the node will be (maximum degree+1), we will start traversing each node and check the color of its neighbor and assign unique color to the node with respect to its neighbor.

For example: For following system of graph

0 1; 1 2; 2 3; 3 4; 4 5; 5 6; 6 3; 0 4; 1 3; 3 5; 2 6; 1 4; 1 5



```
t82: ../../src/xcb_io.c:274: poll_for_event: Assertion `!xcb_xlib_threads_sequen
ce_lost' failed.
hello^ChellohihihihihihihihihihihihihihihihihihihihihihihihihihihihihiIs a plane drawi
ng.
hduser@pravi-Lenovo-IdeaPad-Z510:~/Desktop/Thesis_code$ sudo c++ -I /usr/local/b
oost_1_58_0 un7.cpp -o test1 -o t82 -lgraph
hduser@pravi-Lenovo-IdeaPad-Z510:~/Desktop/Thesis_code$ ./t82
NEW H=G{2} Graph is:
0 1 1 1 1 1 -1
1 0 1 1 1 1 1
1 1 0 1 1 1 1
1 1 1 0 1 1 1
1 1 1 1 0 1 1
1 1 1 1 1 0 1
-1 1 1 1 1 1 0
node color assign
0        0
1        1
2        2
3        3
4        4
5        5
6        0
hduser@pravi-Lenovo-IdeaPad-Z510:~/Desktop/Thesis_code$ 
```

FIGURE 2.3: Assign color to the node according to the algorithm.

2.3.2 Degeneracy approach for coloring

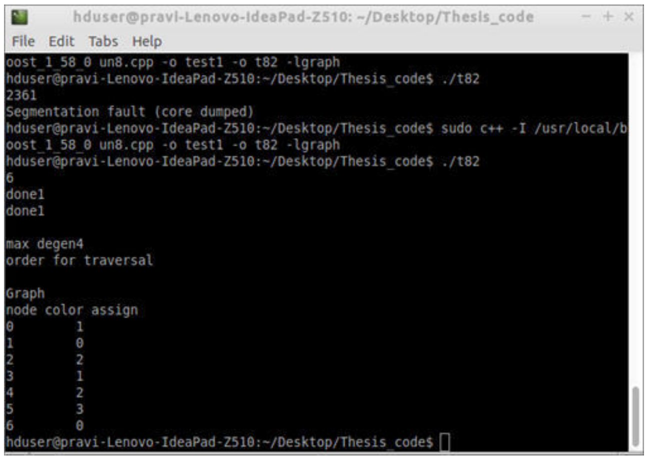
Implemented a scenario where we will determine the degeneracy order[12], according to which we will assign color to node and check that color assigning to this node is other than the assigned color to the neighbor.(color-allotment = degeneracy + 1).

Steps:

1. For Obtaining the degeneracy order we perform the following procedure:
    (a) Let G be original graph. At each step, we will maintain a current graph H, initially H=G.
    Initially d = -1/0.
    While H is non-empty do:
    (b) Let v be a vertex of minimum degree in current graph H. Store label (v) in end now the new graph is H minus v.
    Update d as d = max (d, degree (v)).
2. Now we will use the list of d+1 colors to colors the node.
3. According to stored label we will directly assign color to each node satisfying the same condition that neighbors does not get the same color.

For the above set of data i.e.

0 1; 1 2; 2 3; 3 4; 4 5; 5 6; 6 3; 0 4; 1 3; 3 5; 2 6; 1 4; 1 5



```
oost_1_58_0 un8.cpp -o test1 -o t82 -lgraph
hduser@pravi-Lenovo-IdeaPad-Z510:~/Desktop/Thesis_code$ ./t82
2361
Segmentation fault (core dumped)
hduser@pravi-Lenovo-IdeaPad-Z510:~/Desktop/Thesis_code$ sudo c++ -I /usr/local/b
oost_1_58_0 un8.cpp -o test1 -o t82 -lgraph
hduser@pravi-Lenovo-IdeaPad-Z510:~/Desktop/Thesis_code$ ./t82
6
done1
done1

max degen4
order for traversal

Graph
node color assign
0       1
1       0
2       2
3       1
4       2
5       3
6       0
hduser@pravi-Lenovo-IdeaPad-Z510:~/Desktop/Thesis_code$ 
```

Figure 2.3: Color assigned to the node according to degeneracy methodology.

2.3.3 Randomness methodology for coloring of the graph

Under this implementation randomness is used. All the node are colored randomly i.e. all the nodes are assigned colored with randomness and stored and it is checked that whether the neighbor/connected node is having different color or not, if same color is found then again a new random color is allotted to node and again we check from the first node that each of the connected member is having different color or not, and this methodology is performed continuously unless all the neighbors are assigned with different colors among themselves. But the number of colors depend upon the value equal to 8*max degree of node.

Steps:

1. We will determine the degree of the given graph/data.
2. We will allot total number of color = 8*max degree.
3. While each neighbor has unique color perform following procedure:
   (a) Assign a random color (between 0 to no. of color-1) to node/vertex.
   (b) Check that it does not match with the neighbor color.
   (c) If it matches with the neighbor color allot randomly new color to all the neighbors.
   (d) Apply same process for each node from the first vertex.
4. Finally we obtained the nodes with the color allotted to them.

2.4 Conversion of Graph to Required Two Color Class

2.4.1 For Non-planarity removal

Under this implementation: Initially color are assign to node (from degeneracy method it could be done or even by randomly allocation color to nodes), then for each two colors class we pass all the edge and observe that whether non-planarity exist or not, if non-planarity exist in the sub-graph of this two current color class set then according to condition either we will assign new color to nodes or reassign color to nodes randomly. When all two possible color combination are colored in such a fashion that none of them is forming non-planar scenario, then we can plot any two color class vertex and their corresponding edges for the visualization.

Steps:

1. Total number of colors to be allotted from 0 to n is set (according to degeneracy rule or max degree rule).

2. Then we will assign randomly colors to each of the node between 0 and n (both included).

3. For each two possible color class set we will identify whether this color class contain non-planarity or not, it is verify using Kuratowski algorithm. If given color class is having non-planarity (i.e. sub-graph exist non-planarity) then if the number of nodes in non-planar region is greater hen some specific number (threshold) then we will assign randomly a color to node, so that similar condition does not arise again, but if number of nodes are less than some specific threshold then the we will assign randomly new color apart from this two color to the node. This step is perform continuously unless all the 2-color class combination is forming a planar embedding.

4. Once step 3 is completed then we can ask the user about the 2-color class that he want to view, As two color class is provided by user we take all possible node of that two class and then we will initially converted that portion of graph to tri-connected component so that we can apply canonical ordering and shifting algorithm to determine the coordinates of all the nodes.

5. After determining the coordinates of the node we only plot the edges which are part of the original graph.

For above algorithm, we have used multiple data set consisting of thousands of nodes in it, some of the result of the approach are:

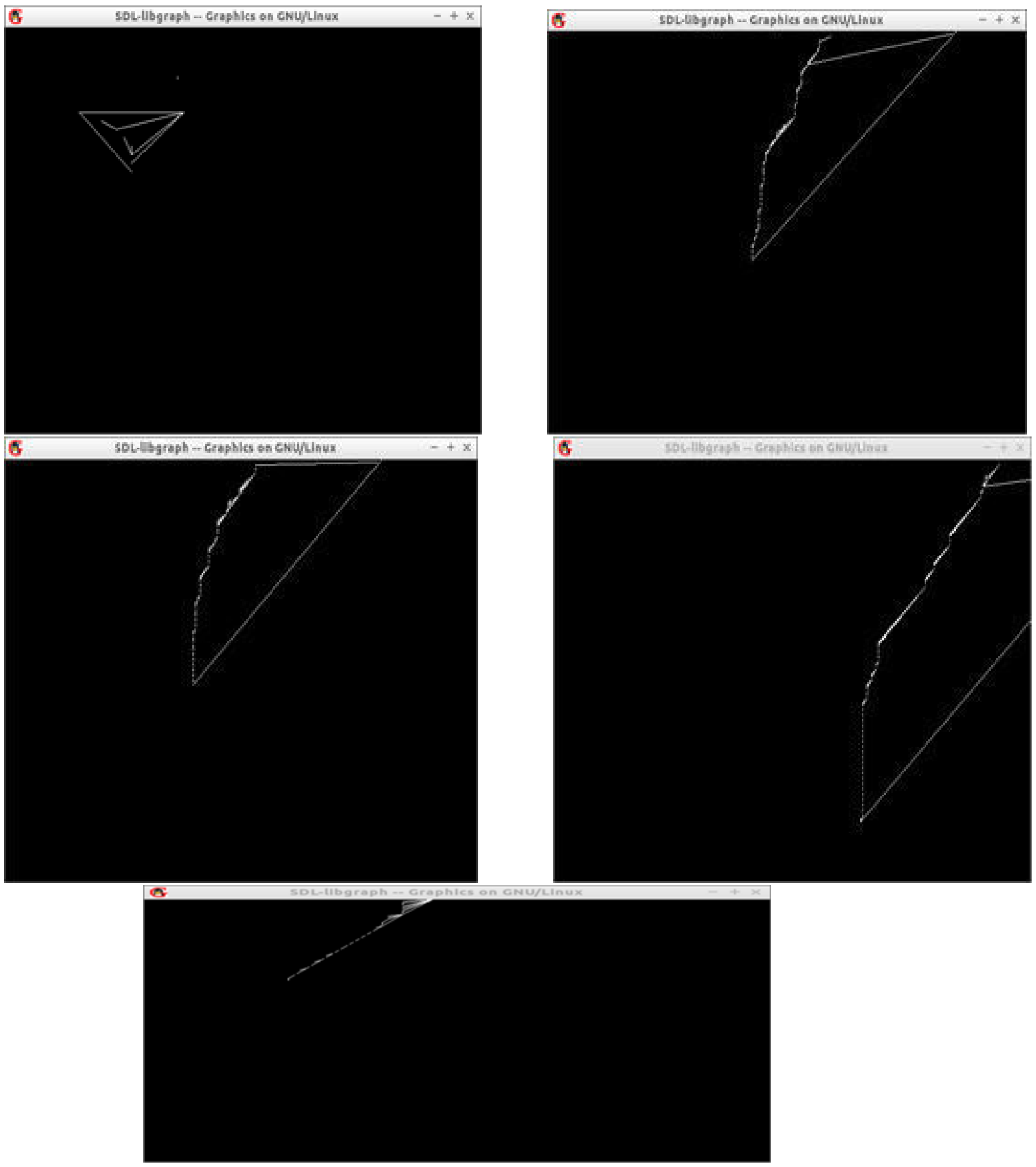


FIGURE 2.4: Various Visualization obtained representing two color class nodes.

2.4.2 For Cyclic nature absorption

Like for the visualization of non-planar graph we apply the procedure of the cycle detection and representation in two color class. Under this procedure we will detect the cycle presence for two color class and if cyclic nature is detected according to threshold value, we will decide that whether we have to recolor any node or assign new color to node and in this way we are going to visualize all possible two color class.

Steps:

1. With the help of degeneracy or randomness approach we will determine the number of color required for coloring the node.
2. For each possible two color we will perform following case unless all possible combination verify that there is no cyclic nature exist in them:
   (a) List of all neighbors is created of specified color.
   (b) Then stack is maintain, also visited list of all member, along with the member neighbor accesses count, and the accessed mode (0, 1, 2) of each node is also maintained.
   (c) For each member if we push any element in stack we check the accessed mode.

   If it is 0: we will add it in stack and make it visited and make it accessed mode as 1.

   If it is 1: we will check that at which point/member it is present in stack, we will pop element of stack and determine the number of member, it will give us the count of length of cycle and its parent as well as back edge member and again we will push back elements into the stack.

   If it is 2: nothing is performed.

   Similarly for each top member of stack we will send its neighbor member and check same condition and check for its accessed mode.

   (d) If length of cycle is less than the threshold value we will reassign randomly color to one of the nodes among them such that it must be different from current color class and if the length of cycle is greater than threshold value than we will check the color of both the nodes of back edge, if both are same then we will assign new color to any two nodes of the cycle and if back edge nodes is having different color than we will assign any one of the node with new color.
3. For the purpose of visualization of the colored node we will use the canonical ordering and shifting approach to get the coordinate of the node.

2.5 Data-Set Referred

For the visualization different graphs data-sets are referred:

DataSet	**Degree**	**Node**	**Edges**
Gr2	66	2361	7178
Yeast	112	2114	4482
Ownership	552	8497	6726

Chapter 3

3-D Visualization of Graph

So initially started with plotting the non-planar graph into 3-dimensional system by plotting some set of edges into one plane and other some set of edges into the another plane.Union of both the edges of both the planes will give us the original non-planar graph.

While plotting the figure(for some class of graphs) we plot some edges resembling a cycle in a plane then we plot remaining edges forming a path into the another plane and to resemble each of the node we used the projection connecting to each of the planar surface.

As the starting step the goal was to get some examples of the non-planar graph which have some planar projection[13,15].So to proceed on it we started by taking a planar graph and then adding a path through all the vertices. The edges of the planar graph are projected on one plane and the edges of the path are projected on another plane.

As to plot the above scenario the result of previous work that uses coloring approach was used. The goal is to achieve the 3-dimensional view of the graph under the mention procedure.

Steps to plot the 3-dimensional scenario for above mentioned procedure we are having the two- dimensional coordinates of two color class nodes and which are the edges present in it:

1. As we are having Y-Z coordinate and considering all X coordinate as 0 to plot the figure in Y-Z plane. Nodes are plotted with Blue color with circle indication the vertex and also each vertex is tagged with its name and coordinate values). Consider it as A1.

2. The projection between A1 and the coordinates A2(all vertex X axis coordinate are at distance of 10 coordinates from each other and y and z coordinates as we obtained in first step) with Red color projection lines.

3. Consider it as A3(taking all nodes Y-axis coordinates as 0, X as 10, 20 ..so on for each vertex in the step 2 order and Z coordinates as initial coordinate that we are plotting) projecting this with A2 with the color Yellow.

4. Created the path (merging A3 with its neighbours so that all the nodes could create a path in it). color used Black.

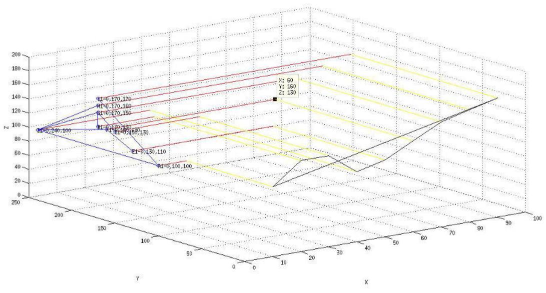
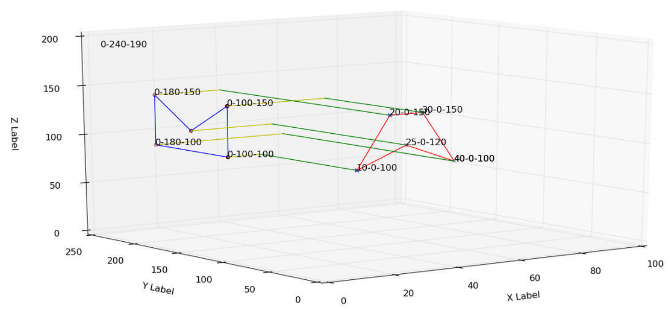


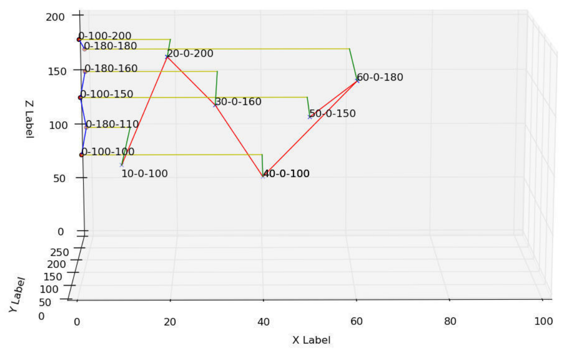
FIGURE 3.1: A 3-D view of a graph.

As the above described visualization in three dimension was achieved we started working on the scenario when the graph is already non-planar and then how to plot it. Like how to plot K_5 or $k_{3,3}$ or any other non-planar graph into two planar surface resembling the 3-dimensional figure with the help of projection of vertex into another plane. We work on both the complete graph as well as bipartite graph[14].

For this we thought of plotting a set of edges which are forming a cycle into one plane and then projecting the node into other plane where the remaining edges are plotted forming a path in that plane,to achieve the proper visualization we set the offset of each of the node projections according to proper view in other plane and we obtain the required drawing.

So the next approach that we applied is that: plotting one set of edges in the vertical plane. Now imagine that this plane is folded down onto the horizontal





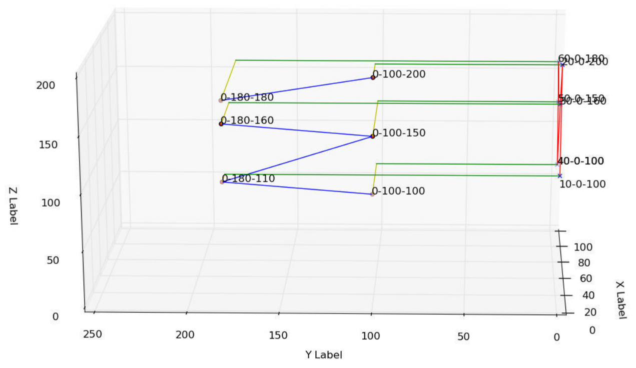


FIGURE 3.2: first figure represent K_5graph in three-dimensional figure. Other figure represent $k_{3,3}$ in three-dimensional figure with different viewing angel.

plane this gives co-ordinates for the vertices in the horizontal plane, where we can plot the other set of edges. In this way we can we can divide the non-planar graph edges into two set of edges which provide the planar view in each of the plane. The Plotting can be performed in the following way:

1. Initially we take some edges of the non-planar graph which are forming a maximum planar view with respect to the graph. We will plot them into one plane.(By setting the coordinate of one of the axis as 0 for each node Let say X axis).
2. Now we will plot all other nodes into the another plane let say X-Y plane plotting all the coordinates of vertex with Z coordinate as 0 and Y coordinates as previous step only and making the X coordinate value as the previous Z coordinate value so that each of them get common view scenario.
3. Then projection lines are plotted and its three dimension coordinates are obtained from both above two steps.

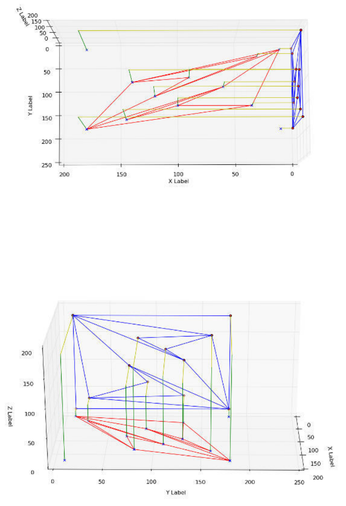


FIGURE 3.3: Representing $k_{6,6}$ in three-dimensional figure with different viewing angel.

Above projections we have performed for various non-planar graph (some complete graph and some bipartite graph).

We observed that all the possible drawing that we are doing in three-dimensional refers to the graph which are having thickness termed as 2. Thickness of a graph[14,15,16] G is the minimum number of planar graphs into which the edges of G can be partitioned. That is, if there exists a collection of k planar graphs, all having the same set of vertices, such that the union of these planar graphs is G, then the thickness of G is at most k.

Geometrical thickness[15] refers to the minimum number of planar planes are possible to plot a given graph such that each planes provide a planar view and all the edges plotted in each planes are straight lines and the vertex position in each of the planes are identical.

So from the geometrical thickness we came to a conclusion that all the previous plotting that we are doing in above mentioned non-planar graphs are having thickness of two as we are able to plot the non-planar graphs into two planar planes with each of the planar drawing are obtained on plotting straight line edges in it.

We came across a scenario that maximum of or say at most 6N-19[17,18] edges can be plotted having geometrical thickness as two where N resemble number of nodes and it is valid for $N >= 9$. So we started working under this scenario to plot more number of edges under above mention case, (by trying with many of the possible cases) we were able to plot more than 6N-19 edges for all $N >= 9$.On the basis of our drawing we were able to plot 6N-17 edges for all $N >= 9$.

The result that we obtain are:

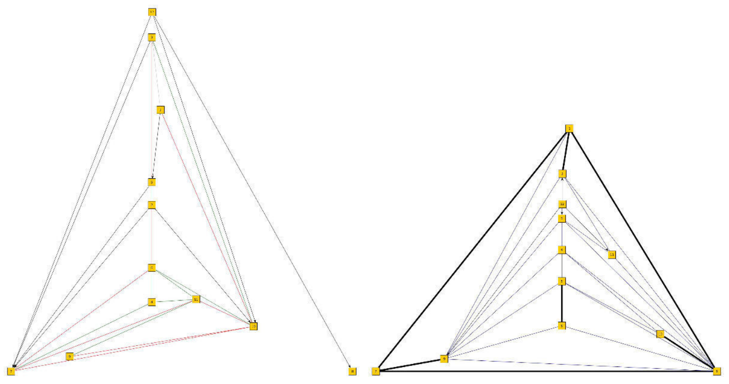
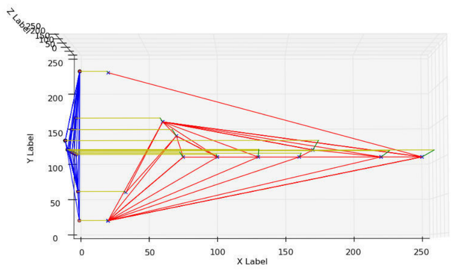
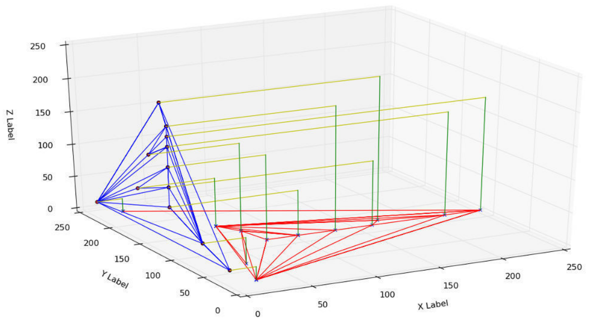


FIGURE 3.4: Representation of 12 nodes with 55 edges in it.

The three Dimension representation with projection lines of above example:





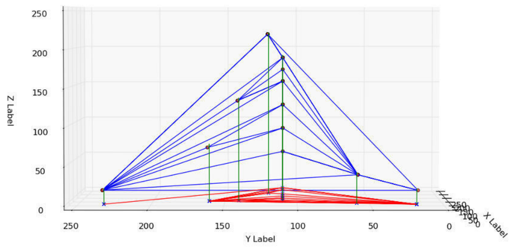


FIGURE 3.5: Representation of 3-D view of 12 nodes with 55 edges in it.

We have plotted for various number of nodes set 8,11,12,15..24 and so on and hence we came to conclusion that we can plot 6N-17 edges in two plane with above pattern of drawing having planar view as well as all the edges are of straight line.

Chapter 4

Conclusion

We have come up with an approach of visualization of non-planar graph in 2-Dimensional planar drawing by using the coloring class approach and with the help of shifting algorithms we can plot the two-color class nodes as a planar drawing at a same time.

We have also come up with an approach of drawing the non-planar graph into two planar graph in 3-dimensional space by plotting some of edges in one of the planes and then projection the other edges into different plane so as to get better visualization of non-planar graph.

Also the graph having geometrical thickness two could be plotted with 6N-19 edges for $N >= 9$, so according our drawing approach we can plot 6N-17 edges for $N >= 9$ nodes. Also same improvement can be performed for different class of graphs.

Chapter 5

Reference

1. Graph Visualization and Navigation in Information Visualization, Ivan Herman, Member, IEEE CS Society,2000.

2. Visualization Tools of Data Structures Algorithms A Survey,International Journal of Advanced Research in Computer Science and Software Engineering,2014.

3. MoireGraphs: Radial Focus+Context Visualization and Interaction for Graphs with Visual Nodes, IEEE Symposium on Information Visualization 2003.

4. Tim Dwyer, A Comparison of User-Generated and Automatic Graph Layouts, 2008.

5. Confluent Drawings: Visualizing Non-planar Diagrams in a Planar Way, Springer-Verlag Berlin Heidelberg 2005.

6. Dan Archdeacon Variations on a theme of Kuratowski, Volume 302,2005.

7. G.Kant, Drawing Planar Graphs Using the Canonical Ordering, Algorithmica, 1996.

8. Melanie Badent, Ulrik Brandes, Sabine Cornelsen, More Canonical Ordering,journal of graph algorithm and applications, vol.15,2011.

9. Goss Kant, Utrecht University, Drawing Planar Graphs Using the lmc-Ordering, report Utrecht University, 1992.

10. The Drawing Framework and Convex drawing, Handbook of Graph drawing and Visualization.

11. Jan van den Heuvel1, Coloring the Square of a Planar Graph, Department of Mathematics, UMEA , SWEDEN, 2002.

12. The Two-Coloring number and degenerate colorings of planar graphs, Society for Industrial and Applied Mathematics, 2009.

13. On Graph Thickness, Geometric Thickness, and Separator Theorems, CCCG, 2009.

14. Geometric Thickness of Complete Graph, University of California, Irvine, 2015.

15. The Drawing Thickness of Graph Drawings Alexandros Angelopoulos ,core-labs, 2014

16. Separating Thickness from Geometric Thickness, David Eppstein, "Towards a Theory of Geometric Graphs", Contemporary Math, 2003.

17. The Geometric Thickness of Low Degree Graphs, CHRISTIAN A. DUNCAN, DAVID EPPSTEIN, Arizona, 2003

18. Thickness and Colorability of Geometric Graphs, Elsevier, 2016