The BRITISH UNIVERSITY IN EGYPT

Layered Graph Drawing

Web Topologies
2021CNM601

Alaa.Hesham

Table of Contents

Abstract 3
Introduction 3
Background 4
Layered/ Hierarchic Layout 5
Implementation methodology 7
Steps of implementing Sugiyama Algorithm 9
Application areas that this layout style is suited for [8]: 10
Drawbacks and limitations of Sugiyama algorithm 10
Related Apps that uses Layered Graph drawing 11
Y-Files for workflow diagrams visualization 11
Advantages and disadvantages of Y-files 13
Hierarchical Clustering 14
Paris algorithm – Node pair sampling [12] 16
Advantages of Paris Algorithm 17
Conclusion 18
References 19
Appendix: 20

Abstract

Graph is a way of visualizing information (datasets). It clearly presents information that are embedded in tabular format of a database. Graphs comes also in several. However, to benefit the most of a given graph. choosing the suitable layout is highly important. This can be done based on following a set do criteria that describes a good layout drawing. One of which is layer/Hierarchical drawing. Hierarchical drawing. it aims to place nodes in horizontal layers and minimize cross edges. Layered layout is implemented using Sugiyama framework which is composed of 5 levels: cycle breaking, leveling, crossing minimization, vertex positioning and edge drawing. Two different application domains used Layered layout were highlighted. The first is Y-files, it used hierarchical layout for visualizing workflows. It is a smooth tools with tons of features, yet it does not work on dynamic graphs. the second application is Hieratical clustering. The presented model is based on node sampling. This algorithm is parameters free yet faces difficulties handling large dataset.

Introduction

Graphs are a type of data structures. It stores information in form of nodes (vertices) and edges(arcs). These graphs present information in a visualized format that helps highlighting problems and extracting information. Constructing graphs and manipulating them is a science called graph theory. It studies the relationships between nodes [1]. This study provides analysis that take a crucial part in decision making, describing dynamic systems, understanding social networks as well as solving real world problems like scheduling problems or resources allocation. It can be used for DNA sequencing, Computer network security and many numerous domains.[2]. Therefore, it is necessary to understand the types and properties of graphs to guarantee optimal analysis of any

given graphs. The common types of graphs are: directed graphs, weighted graphs, undirected graphs, bipartite graphs …etc. However, graphs can be complicated and unclear and may requires adjusted to be in a readable format. Adjustments can be like transforming graphs form non-planar graphs (graphs with cross edges) to planar graphs, other instance can be removing cycles from a graph to prevent duplicates or potential overlapping etc. Other concepts that ensure a better visualization is graph layout. The presentation(layout) of a graph plays a significant role in manipulating any given graphs.

Background

Graph layout is using the appropriate presentation for drawing a set of connected nodes and their associated edges. Essentially, it involves determining the location of the nodes and the shape of the links [3]. Furthermore, graph layout comes in different types: Circular layout(see appendix), it gathers nodes in star topologies. Radial Layout (see appendix), concerns with grouping nodes into several groups each has a center in an overall circular shape. Tree Layout, it is suitable for tree collections converted into a graph. Layered or hierarchical layout (see appendix) which presents directed graphs in a form of horizontal layers. And may other several layouts, yet this paper will discuss the layered layout in detail. However, determining the suitable layout comes in a common criteria that needs to be followed to be able to choose the most fit layout. The criteria is:

- Minimizing the number of link crossings
- Minimizing the total **area** of the drawing
- Minimizing the number of **bends** (in orthogonal drawings)

- Maximizing the smallest **angle** formed by consecutive incident links
- Maximizing the display of **symmetries**

To meet this criteria, there are several factors like the domain of the problems and others. Moreover, methods that apply layout drawing are: Automatic, Semi-automatic, Layout from scratch and Incremental layout [3].

Layered/ Hierarchic Layout

Hierarchical/layered graphs are not a type of a graph but rather a type of a layout or a drawing. It separates nodes in a horizontal layers. It aims to highlight the flow of events the direction of nodes. It works with directed graphs in order to emphasize the flow[6]. This layout can be constructed with a top down or a bottom-up approach. It facilities reading directed diagrams or flowcharts by minimizing number of cross over edges that complicates relationships on the diagram. Additionally, it aims to remove cycles for less complexity and clear role of nodes, by applying these concepts and some others this layout applies the ideal criteria of a suitable layout design.

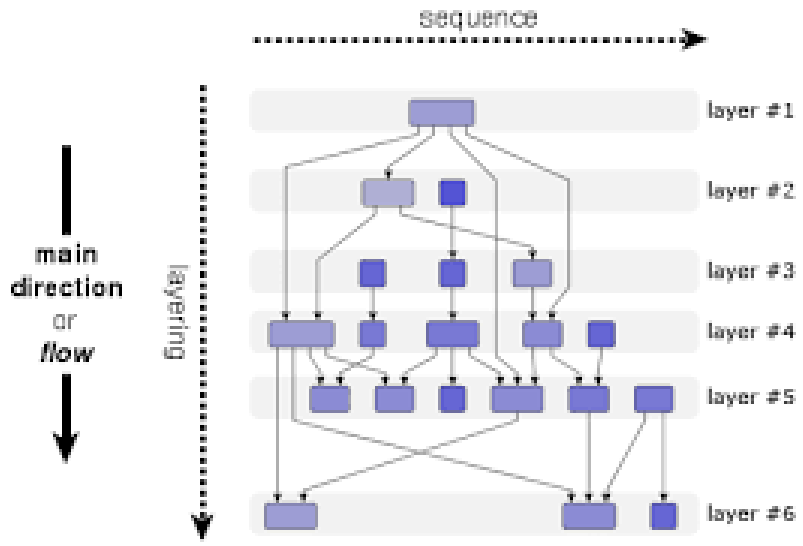


Figure 1:Layered Layout Overview

Figure 1 above illustrate how layered layout looks like. Layered layout works on DAG (directed acyclic graphs). However, some input directed graphs comes with cycles as most graphs are usually in large scales and the probability of cycles is high. As a result, on of the essentials steps in this layout is to break detected cycle. The layout also uses greedy heuristic approaches within the process. It aims to draw edges upwards to sever the concept of visualizing an organized flow. Also, it places nodes in several appropriate levels in the leveling stages. Additionally, the layout minimizes possible or repeated edges that can be crossing over. These steps above transform the shape of the graph as a result vertices get repositions evenly in the layers in an organized way and after the alterations based on the modifies edges. Then edges are redrawn after the modifications in number of edges and repositioning of vertices. The algorithm/framework used for implementing layered graph drawing is called Sugiyama named after developer Kozo Sugiyama who

developed this application. [7] Apart from that, Layered Layout comes in different styles [8]. (Polyline Edge Routing -Orthogonal Edge Routing -Curved Edge Routing and Grouped Graph) see appendix.

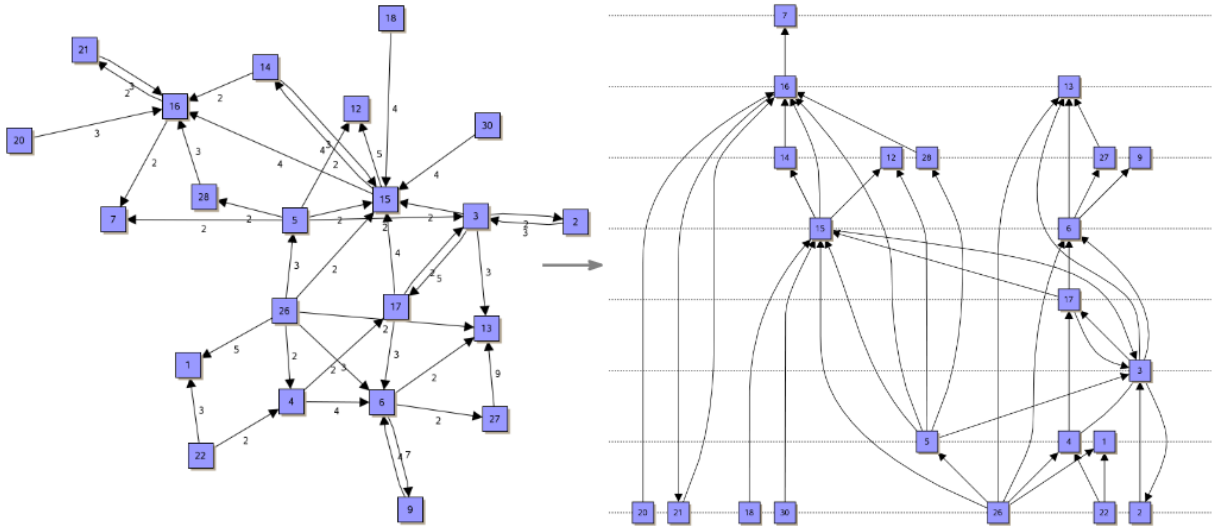


Figure 2: A illustration of A given graph (on the left hand side) then it is transformed to a hierarchal graph in the right hand side[9]

Implementation methodology

The algorithm of Sugiyama relies on implementing 3 main concepts. First concept is: layer assignment. It determines which node to be assigned to which layer. Keeping in mind that that nodes should be spread evenly and purposefully according to their weights and corresponding edges (this can be challenging). Second Concept is: Crossing reduction. It calculates the sequence of nodes for each as well as the possible minimum crossed

edges between nodes this can be done with median heuristics or greedy switch heuristic . Third concept is Coordinates assignments. It uses e Down-Up procedure with the barycenter algorithm of a vertex to assign coordinates to nodes based on their position in layers and their calculated sequence.

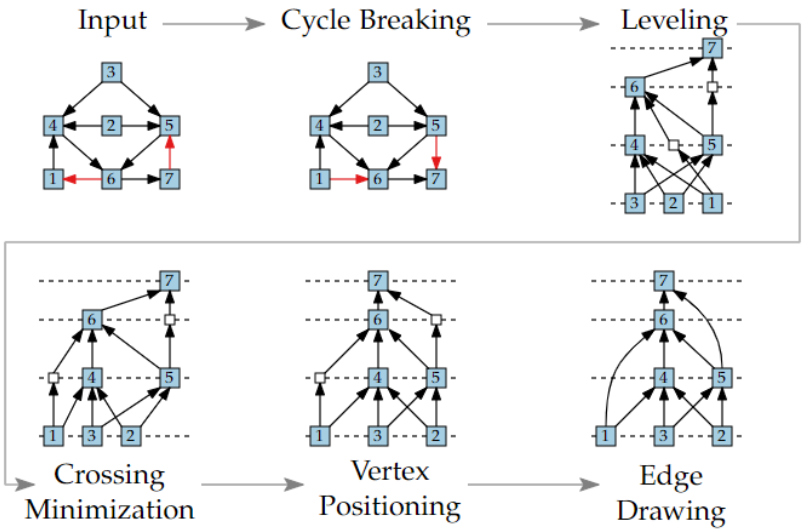


Figure 3:phases of Sugiyama algorithm [9]

The algorithm is applied based on a set of requirements below [7]:

a) Vertices are drawn on horizontal line representing layers, in which all edges are pointing downwards. *(easy)*

b) Short span edges between adjacent levels are drawn with straight lines. *(easy)*

c) Long span edges between nonadjacent levels are drawn as straight as possible. *(challenging)*

d) Number of edges crossing should be minimized as possible

e) Vertices connected to each other should be linked as close as possible.

f) Edges should be distributed evenly around the common target (in which in-degree/outdegree are balanced) *(challenging)*

Steps of implementing Sugiyama Algorithm [7]

Step1: Receiving a directed graph as an input to be processed into layered drawing.

> ***Step 1.1***: Transforming the directed graph into DAG by getting rid of edges causing cycles (reverse direction).
>
> ***Step 1.2:*** Place vertices in a format of several horizontal layers.
>
> ***Step 1.3***: Transform the graph into hierarchical based by adding dummy vertices to support long span edges.

Step 2: Specify the sequence of vertices on each level and minimize the total number of edges crossing.

Step 3: Determine the coordinates of vertices *(c,e,f)* while preserving a linear order of levels

Step 4: Draw the graph as a layered graph in a final version after modifications and removing dummy vertices.

The complexity of this algorithm is O(|V ||E| log |E|) requiring O(|V ||E|) memory which indicates that it is unstable for visualizing large graphs because of the many variables listed below[10].

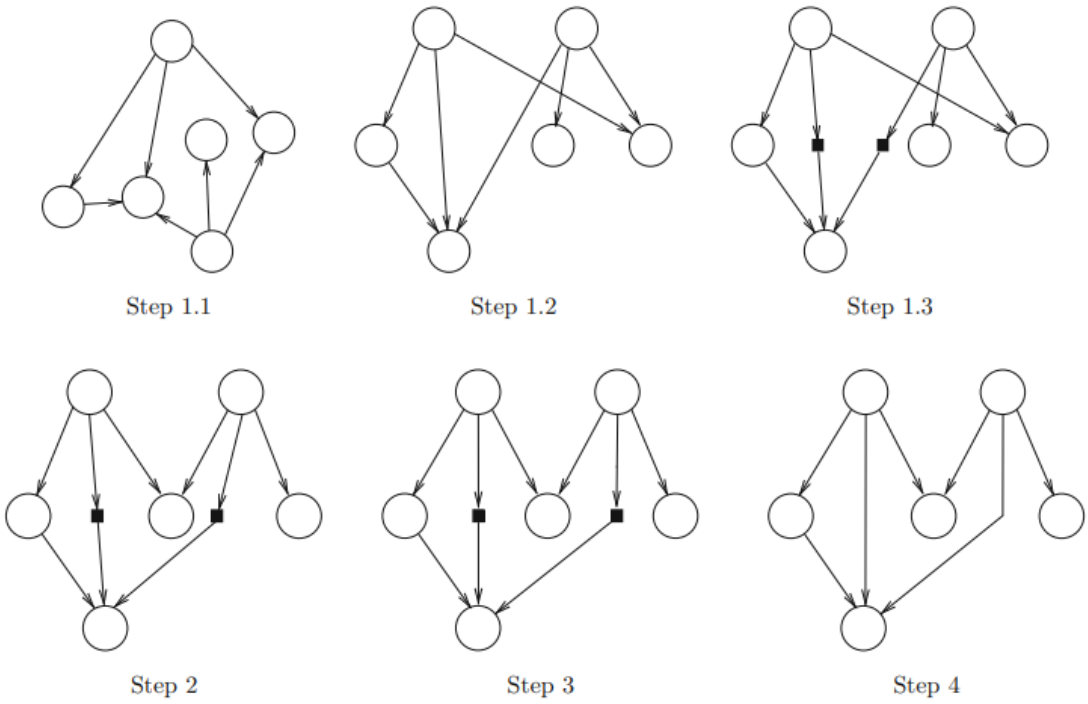


Figure 4:illustration of steps of implementing Sugiyama algorithm [7]

Application areas that this layout style is suited for [8]:

- Workflow visualization
- Software engineering (e.g., state diagram or activity diagrams)
- Process modeling
- Database modeling (e.g., Entity-Relationship diagrams)
- Bi-partite graphs (right hand side in a layer and left-hand side in second layer)
- Planner graphs
- Bioinformatics (e.g., biochemical pathways)
- Network management
- Decision diagrams
- Drawing critical paths

Drawbacks and limitations of Sugiyama algorithm

This algorithm uses decent computational problems effort as shown the complexity mentioned above. This remains in maintaining the requirements of building the framework which are obtained from the criteria of building a suitable layout. Precisely, when removing cycles and portioning nodes in several layers. It is classified to be a difficult combinatorial optimization problem [7].when reversing edges to remove cycles it required minimum feedback arc set problem of type NP-hard (see appendix). Moreover, in portioning nodes, imposing number of nodes and corresponding layers is called layering problem of type NP-complete.

Related Apps that uses Layered Graph drawing

Y-Files for workflow diagrams visualization

Yfiles is infamous software company. It was stablished since 2000.it is dedicated for visualizing and analyzing graphs. It used graph visualization SDK and algorithms for transforming graphs into several layout in 5 different platform[11]. This section y-files uses its own APS for implementing layered graph drawing in an outstanding performance. It includes 5 different modes for visualizing Layered graphs. (Incremental layout mode, Nested Graph Structure, Swimlanes or Grid-like Table Structures, Bus-Like Edge Style and Custom Port Assignment) see appendix.

Yfiles draws a graph to be layered on 4 phases:

First phase is called basic operations:

It works like an initial set up. In which is selects one of the available layout methods(From scratch , incremental , orientation sample). it uses an API for each. Additionally, it initiates3 APIs. Node Layout descriptor APi , This process nodes and their adjacent as well as reading minimum distance between them. Edge Layout Descriptor Api , it is used for drawing different edges layout styles. And Incremental Hints factory API which Is initiated in case the graph is incremental only.

Second phase is called Layering:

It uses 2 APIs , ILayerer and Ilayer constraint Factory. First api assigns nodes to layers based on given layer assignment options (Hierarchical topmost, Hierarchal tight tree, BFS

,User defined).layerer constraint factory is used to specify constrains , it is used in cases like from scratch and user defined modes.

Third phase is Sequence:

It uses 2 APIs (Default layer sequencer, Isequance constraint factory). They determine the orders of nodes within a layer. It can be determined also given node order options(constraints).

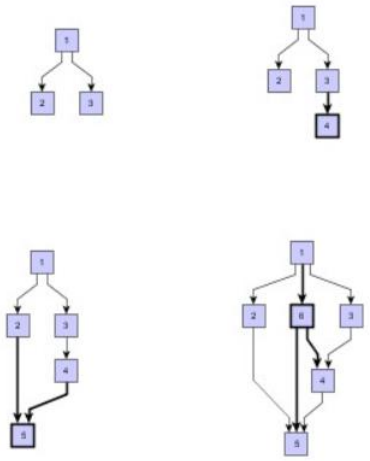
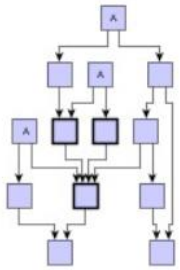
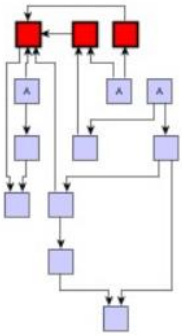


Figure 5:ordering nodes in sequence in phase 3 using default layer sequencer



Usual hierarchical layout (i.e. without taking constraints into account)



Resulting hierarchical layout where the constrained nodes are placed in the topmost layer.

Figure 6:ordering nodes in sequence in phase 3 using constraint factory sequencer

Forth phase is Drawing:

In this phase the coordinates of the nodes are placed. This is achieved through 2 apis (Simplex node placer API and Default drawing distance Calculator) they place nodes with respect to their sequence. it uses barycenter mode. Straighten edges for supporting the optimal path as well as a compaction strategy API to keep them in horizontal shape and eventually labeling competition API to determine the adjacent edges labels. The second API calculating the distance between ach element given the output of the first API.

Advantages and disadvantages of Y-files

- It is significant for visualizing diagrams in SE domain. It supports reading different diagrams.
- Supports different modes of the layered drawing. Using nested groups layered mode is a great choices for visualizing state machine diagrams in SE.
- It allows building customized graphs with customized constraints.
- Words across platforms. It generates layered graph in 5 different extensions like HTML, JavaFX, Java (Swing), WinForms, and WPF.

Yet it had some drawbacks

- It is complex. It is built on top of numerous APIs that are subjected to frequent change, also built with very sophisticated algorithms.
- It is not an open source, but more of a tool with unrealized code for their APIs
- It is not suitable for dynamic graphs.

Hierarchical Clustering

As per mentioned in sections above, layered/hierarchal layout is a type of drawing, it can be applied as previous application in visualizing diagrams. In this section it is used to visualize clusters in a hierarchical format. The core concept of hierarchical clustering relies on collecting several clusters with common characteristics. Building the hierarchy can be done via 2 approaches: top-down approach (not covered in report as it is rarely used) and agglomerative (bottom up more common).

The bottom-up approach takes a graph of nodes or datasets as an input and generate a dendrogram (hierarchical structure of clusters). This is implemented in 4 stages: first, it treats each node as a separate cluster. Second stage, measure the distance between nodes (clusters) and find the closest clusters. Third stage, it merges those clusters into one cluster (as if they are placed in one layer). For more details observe the figure below:

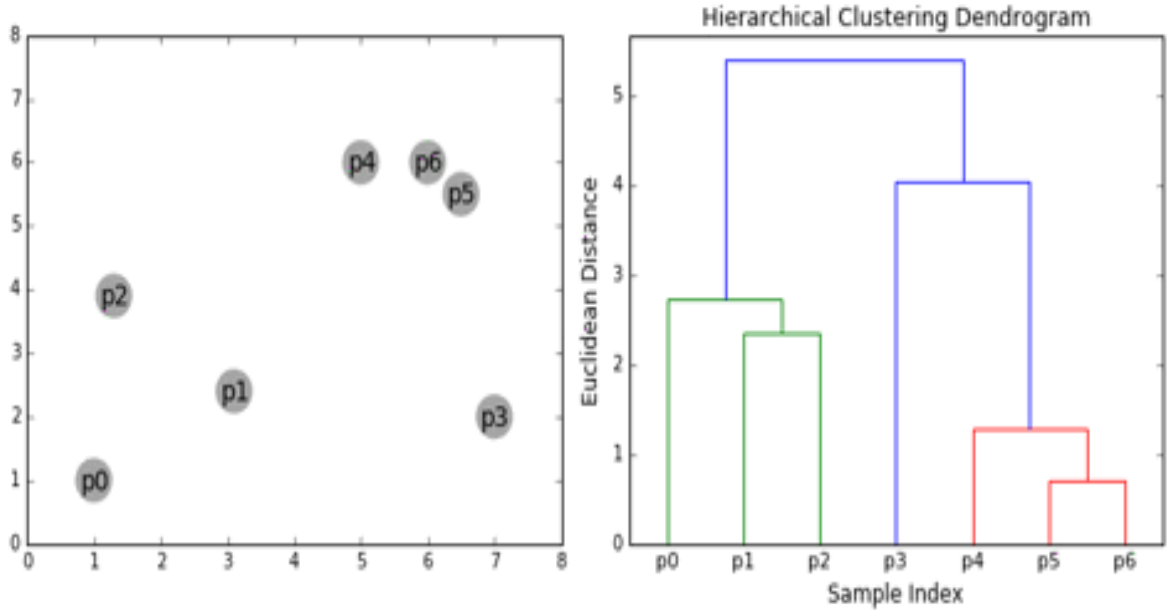


Figure 7: Dynamic demonstration of how agglomerative hierarchical clustering works

the significance in algorithms that applies hierarchical layout for clustering is shown in how the distances between nodes are measured for layering assumption. This reports presents 2 agglomerative approaches for preforming hierarchal clustering. Which are Louvain and Node pair sampling (Paris) as well as a brief comparison between both.

Paris algorithm – Node pair sampling [12]

It is developed in 2018 , inspired by modularity- based clustering for detecting communities. The significance in this algorithm is that it measures the distance between nodes based on joint distribution in conditional probability as the following the distance between two distinct nodes i, j as the node pair sampling ratio

$$d(i,j) = \frac{p(i)p(j)}{p(i,j)},$$

Based on which, nodes I,j are considered far (disconnected) if p(I,j) = 0 . As a result, Nodes i, j are close with respect to this distance if the pair i, j is sampled much more frequently through the joint distribution p(i, j) than through the product distribution p(i)p(j). Therefore, the distance will be calculated as the following:

$$d(i,j) = \frac{p(i)}{p(i|j)} = \frac{p(j)}{p(j|i)}.$$

Similarly , if the probability of distribution for clusters is given as :

$$\sum_{b \in C} p(a,b).$$

The distance for sampling 2 clusters will be as following:

$$d(a,b) = \frac{p(a)}{p(a|b)} = \frac{p(b)}{p(b|a)}.$$

Furthermore, the final format for merging a cluster of 2 nodes with a third node to build the bottom up approach will be:

$$d(a \cup b, c) = \left(\frac{p(a)}{p(a \cup b)} \frac{1}{d(a,c)} + \frac{p(b)}{p(a \cup b)} \frac{1}{d(b,c)} \right)^{-1}.$$

Eventually, the formula for placing the positioning (reducibility for any distinct cluster will be):By the reducibility property, merging clusters a and b cannot decrease their minimum distance to any other cluster c.

$$d(a \cup b, c) \geq \min(d(a, c), d(b, c)).$$

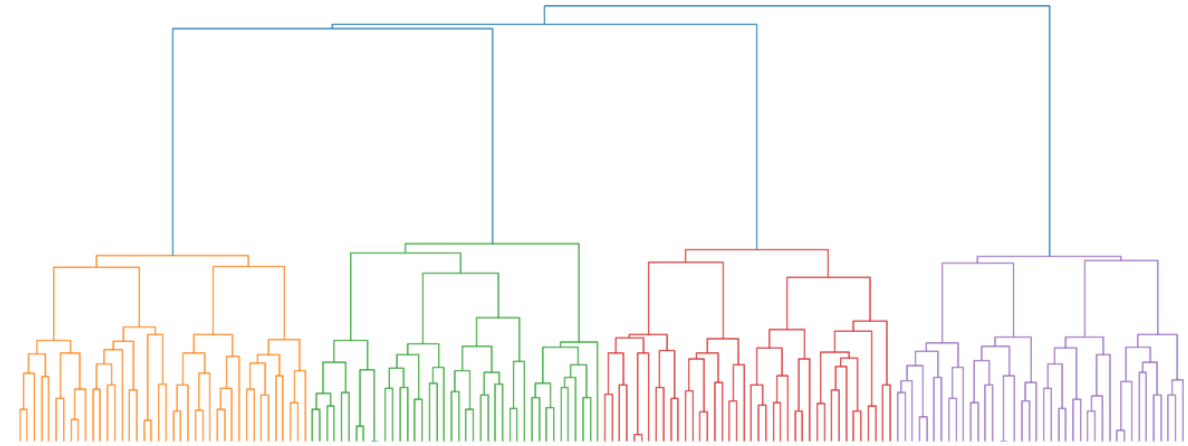


Figure 8:this is how the output dendrogram looks like using Node Pair sampling algorithm for Hierarchical clustering

Advantages of Paris Algorithm

Advantages

- Efficient and most of all parameters free.
- Fast implementation of dendrogram because it uses nearest neighbor chain scheme.
- It does not run several times for each resolution's parameter, instead it run once and the output is shown on the dendrogram.
- Captures multi-scale natures of a dataset.

Disadvantages

- Sensitivity to noise and outliers.
- Faces Difficulty when handling with different sizes of clusters.
- Might fail when computing large datasets.

Conclusion

Graph is a way of visualizing information (datasets). it clearly presents information that are embedded in tabular format of a database. graphs can represent and solve many domains like social networks, scheduling problems decision making ..etc. graphs comes also in several types like weighted graphs , directed graphs ...etc. However, to benefit the most of a given graph. choosing the suitable layout is highly important. this can be done based on following a set do criteria that describes a good layout drawing. On of which is layer/Hierarchical drawing. Hierarchical drawing comes in different tyles. it aims to place nodes in horizontal layers and minimize cross edges. Layered layout is implemented using Sugiyama framework which is composed of 5 levels: cycle breaking, leveling, crossing minimization, vertex positioning and edge drawing. Applications that uses Hierarchical layout vary, like workflow visualization and diagrams, process modeling, drawing critical paths ..etc. 2 different application domains used Layered layout. the first is Y-files, it used hierarchical layout for visualizing SE diagrams. it is a smooth tools with tons of features. Yet it does not work on dynamic graphs. the second application is Hierarchical clustering. it builds clusters in a Hierarchical format. the presented model is based on node sampling. this algorithm is parameters free yet faces difficulties handling large dataset. Generally, layered drawing is easy to understand as provides an organized vision with a clear role of each node which is helpful in decision making. Yet, it lacks flexibility and promotes solid thinking. as a node cannot have different roles. Additionally, power is divided in a Hierarchical format in which low levels have less powered and it is bidirectional top down.

References

[1] Gupta, V., 2015. *Importance of Graph Theory*. [online] Compmath-journal.org. Available at: <http://compmath-journal.org/dnload/Vandana-Gupta/CMJV06I06P0306.pdf> June 2015.

[2] Flovik, V., 2020. *What is Graph Theory, and why should you care*. [online] Towards Data Science. Available at: <https://towardsdatascience.com/what-is-graph-theory-and-why-should-you-care.> 20 August 2021.

[3] Ibm, 2021. *The Concept of graph layout*. [online] Available at: <https://www.ibm.com/docs/en/was/8.5.5?topic=SSEQTP_8.5.5/com.ibm.websphere.web2mobile.ilog.dojo.diagrammer.help/Content/Visualization/Documentation/Dojo/Dojo_Diagrammer/_pubskel/ps_dojo_diagrammer_ic98.html> 27 February 2021.

[4] [2] yWorks diagramming company the, "Circular Graph Layout," *yWorks, the diagramming experts*. https://www.yworks.com/pages/circular-graph-layout.

[5] A. Pavlo, C. Homan, and J. Schull, " A parent-centered radial layout algorithm for interactive graph visualization and animation | Semantic Scholar," *A parent-centered radial layout algorithm for interactive graph visualization and animation | Semantic Scholar*, Jan. 01, 2020. https://www.semanticscholar.org/paper/A-parent-centered-radial-layout-algorithm-for-graph-Pavlo-Homan/fbfe5e4fa35c31e58e316762cc7ea7d025f003d4

[6] P. Pati and G. Jaum, "Hierarchical graph representations in digital pathology - ScienceDirect," *Hierarchical graph representations in digital pathology - ELSEVIER*, Oct. 27, 2021. https://www.sciencedirect.com/science/article/pii/S1361841521003091

[7] N. S. Nikolov, "Sugiyama Algorithm - Encyclopedia of Algorithms," *Springer Science+Business Media New York* , Jan. 0, 2016. https://www.researchgate.net/publication/303226437_Sugiyama_Algorithm (accessed Jul. 16, 2022).

[8] yFiles D. Documentation, "Hierarchical Layout | Automatic Graph Layout | yFiles for HTML Documentation," *Hierarchical Layout | Automatic Graph Layout | yFiles for HTML Documentation*. https://docs.yworks.com/yfiles-html/dguide/layout/hierarchical_layout.html (accessed Jul. 16, 2022).

[9] P. Kindermann, "Classical Approach – Sugiyama Framework - Hierarchical drawings," *the Universities of Rhineland-Palatinate*, Jun. 02, 2021. https://seafile.rlp.net/f/51c1f201d4424113be3c/ (accessed Jul. 16, 2022).

[10][7]M. Eiglsperger and M. Siebenhaller, *An Efficient Implementation of Sugiyama's Algorithm for Layered Graph Drawing*, S http://jgaa.info/ vol. 9, no. 3, pp. 305–325 (2005)., vol. 3, 9 vols. Z¨uhlke Engineering AG 8952 Schlieren, Switzerland: Journal of Graph Algorithms and Applications, 2005.

[11] yWorks diagramming company the, “Layered Graph Layout,” *yWorks, the diagramming experts*. https://www.yworks.com/pages/layered-graph-layout#partitioning-into-swimlanes-or-grid-like-table-structures (accessed Jul. 16, 2022).

[12]T. Bonald, B. Charpentier, A. Galland, and A. Hollocou, “Hierarchical Graph Clustering using Node Pair Sampling - Archive ouverte HAL,” *Hierarchical Graph Clustering using Node Pair Sampling - Archive ouverte HAL*, Oct. 04, 2018. https://hal.archives-ouvertes.fr/hal-01887669 (accessed Jul. 16, 2022).

Appendix:

Circular layout:

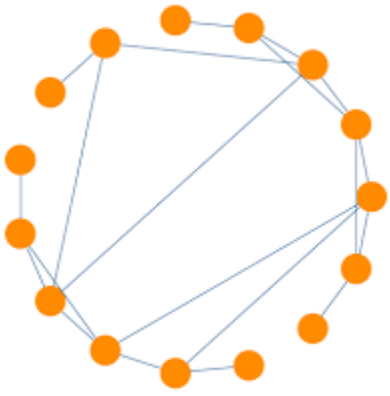


Figure 9:circular Layout graph [4]

Radial layout:

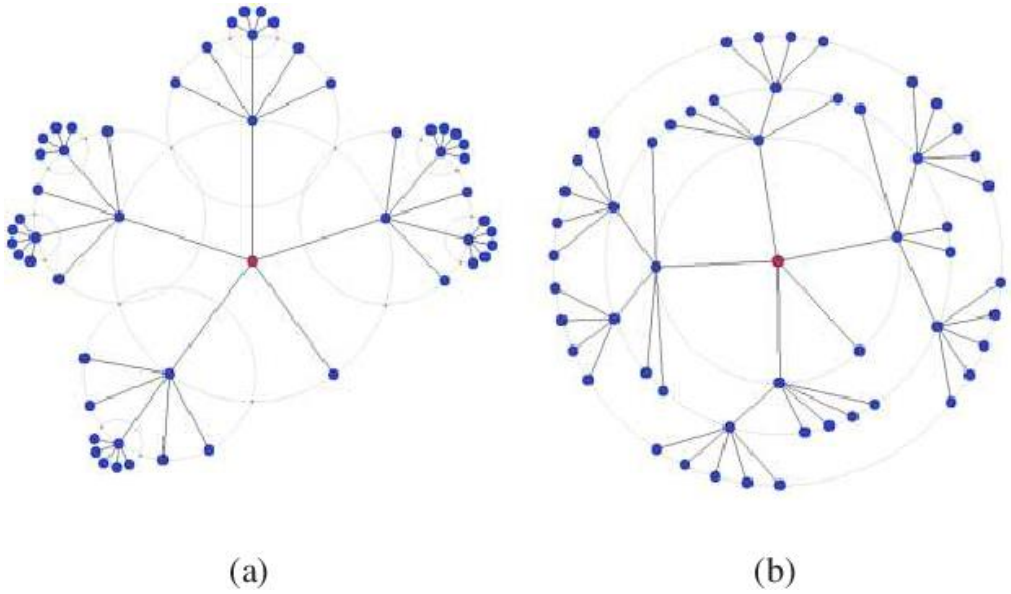


Figure 10:Radial Layout graph [5]

Layered/Hierarchical Layout

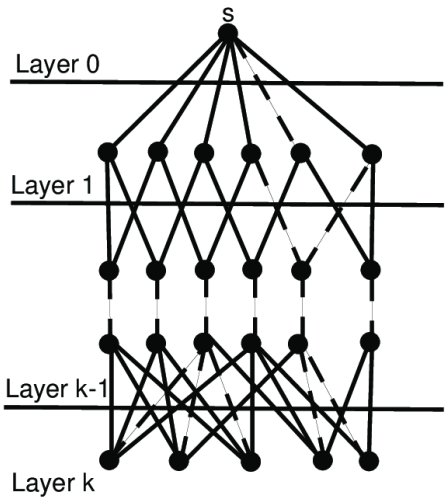


Figure 11: Layered Layout graph

Layered layout styles

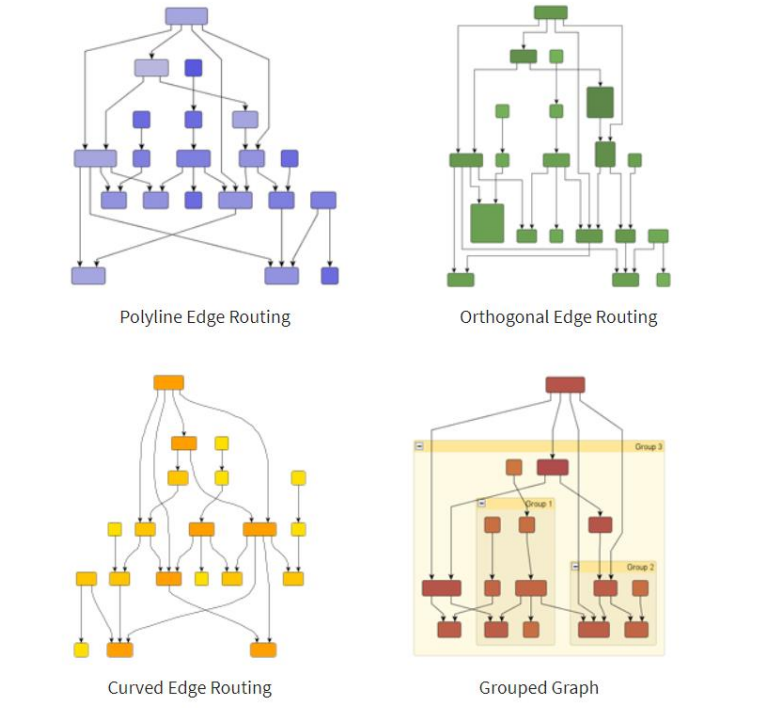


Figure 12: Hierarchical layout different styles[7]

Hierarchical Nested Layout mode

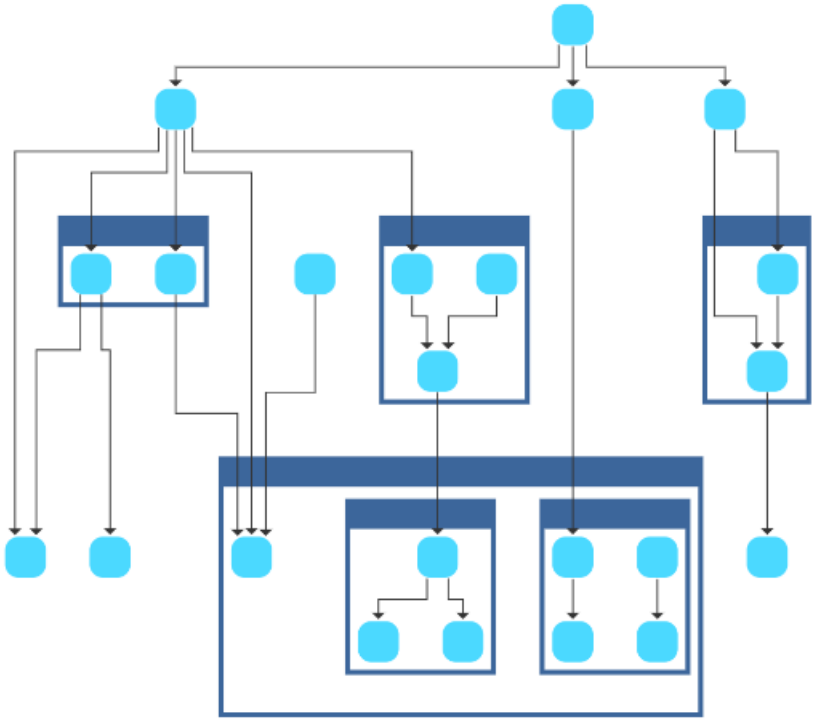


Figure 13:Hierarchic Nested example of Layered graph supported by Yfiles

Swimlanes or Grid-like Table Structures

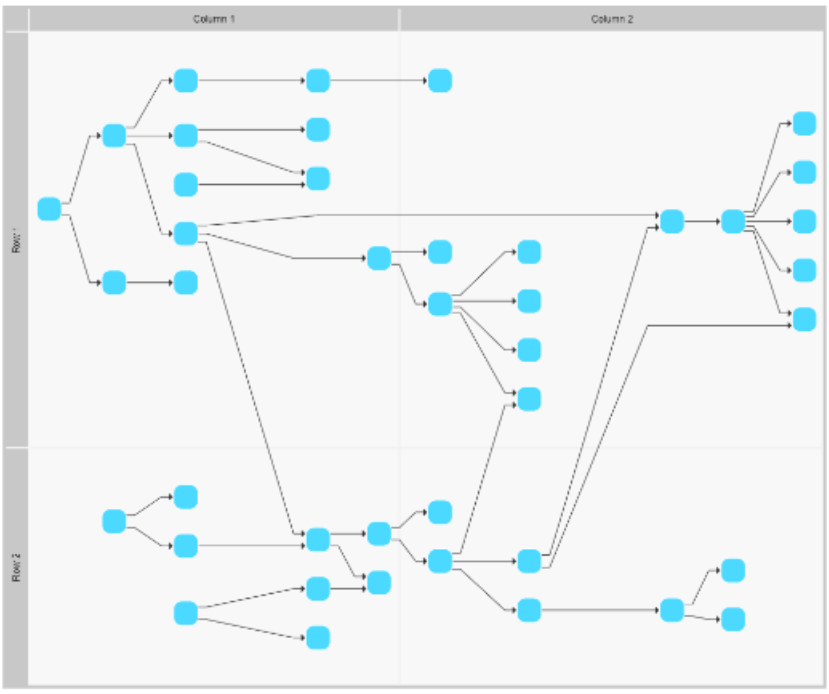


Figure 14: Swimlanes or Grid-like Table Structures layered mode supported by y-Files

Bus-like Edge Style

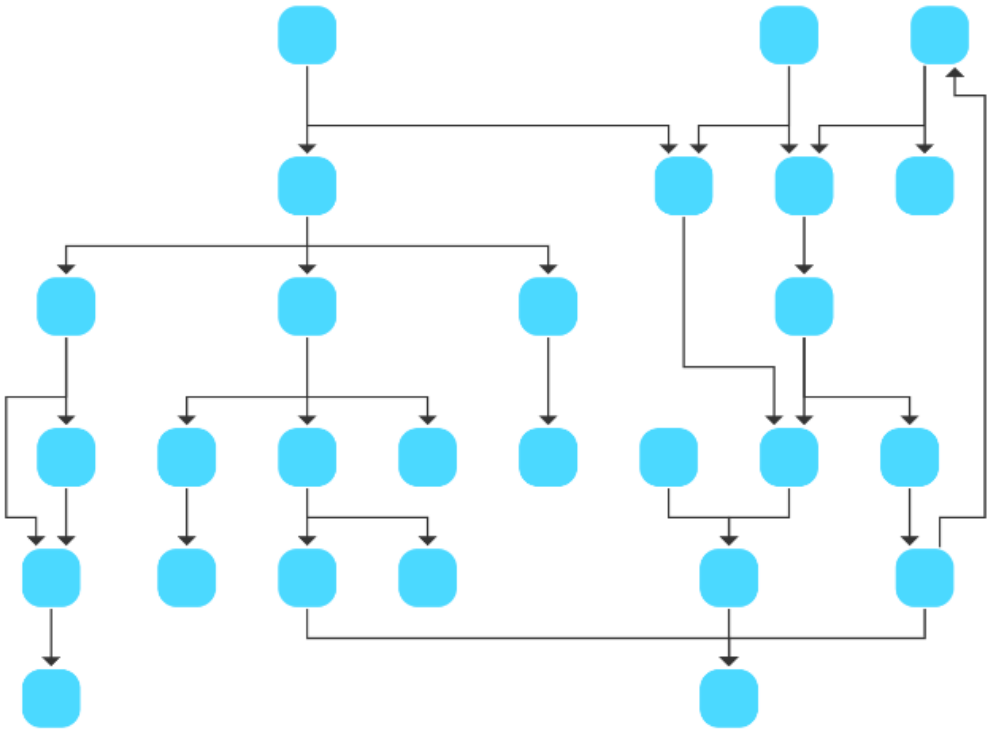


Figure 15: Bus-like Edge Style layered mode supported by Yfiles

Node pair sampling Algorithm for Hierarchical clustering (Paris)

```
11
12  def paris(G, copy_graph = True):
13      n = G.number_of_nodes()
14      if copy_graph:
15          F = G.copy()
16      else:
17          F = G
18
19      # index nodes from 0 to n - 1
20      if set(F.nodes()) != set(range(n)):
21          F = nx.convert_node_labels_to_integers(F)
22
23      # node weights
24      w = {u: 0 for u in range(n)}
25      wtot = 0
26      for (u,v) in F.edges():
27          if 'weight' not in F[u][v]:
28              F[u][v]['weight'] = 1
29          weight = F[u][v]['weight']
30          w[u] += weight
31          w[v] += weight
32          wtot += weight
33          if u != v:
34              wtot += weight
35
36      # cluster sizes
37      s = {u: 1 for u in range(n)}
38
39      # connected components
40      cc = []
41
42      # dendrogram as list of merges
43      D = []
44
45      # cluster index
46      u = n
47      while n > 0:
48          # nearest-neighbor chain
49          chain = [list(F.nodes())[0]]
50          while chain != []:
51              a = chain.pop()
52              # nearest neighbor
53              dmin = float("inf")
```

```
dmin = float("inf")
b = -1
for v in F.neighbors(a):
    if v != a:
        d = w[v] * w[a] / float(F[a][v]['weight']) / float(wtot)
        if d < dmin:
            b = v
            dmin = d
        elif d == dmin:
            b = min(b,v)
d = dmin
if chain != []:
    c = chain.pop()
    if b == c:
        # merge a,b
        D.append([a,b,d,s[a] + s[b]])
        # update graph
```

```
                    if d < dmin:
                        b = v
                        dmin = d
                    elif d == dmin:
                        b = min(b,v)
            d = dmin
            if chain != []:
                c = chain.pop()
                if b == c:
                    # merge a,b
                    D.append([a,b,d,s[a] + s[b]])
                    # update graph
                    F.add_node(u)
                    neighbors_a = list(F.neighbors(a))
                    neighbors_b = list(F.neighbors(b))
                    for v in neighbors_a:
                        F.add_edge(u,v,weight = F[a][v]['weight'])
                    for v in neighbors_b:
                        if F.has_edge(u,v):
                            F[u][v]['weight'] += F[b][v]['weight']
                        else:
                            F.add_edge(u,v,weight = F[b][v]['weight'])
                    F.remove_node(a)
                    F.remove_node(b)
                    n -= 1
                    # update weight and size
                    w[u] = w.pop(a) + w.pop(b)
                    s[u] = s.pop(a) + s.pop(b)
                    # change cluster index
                    u += 1
                else:
                    chain.append(c)
                    chain.append(a)
                    chain.append(b)
            elif b >= 0:
                chain.append(a)
                chain.append(b)
            else:
                # remove the connected component
                cc.append((a,s[a]))
                F.remove_node(a)
                w.pop(a)
                s.pop(a)
                n -= 1

    # add connected components to the dendrogram
    a,s = cc.pop()
    for b,t in cc:
        s += t
        D.append([a,b,float("inf"),s])
        a = u
        u += 1

    return reorder_dendrogram(np.array(D))
```

NP: Nondeterministic polynomial time

A problem is called NP (nondeterministic polynomial) if its solution can be guessed and verified in polynomial time; in which nondeterministic means that no particular rule is followed to make the guess

NP-Complete:

an NP problem can be reduced to the given problem) then the problem is NP complete.

NP-hard:

A decision problem H is NP-hard when for every problem L in NP, there is a polynomial-time many-one reduction from L to H