International Journal of Engineering and Applied Sciences (IJEAS)
ISSN: 2394-3661, Volume-2, Issue-7, July 2015

Analysis of Various Graph Layout Approaches Used in GUESS Software

Himanshu Sharma, Vishal Srivastava

Abstract— **One challenge in node-link diagrams is how to efficiently provide a node placement or layout that will yield a meaningful graph visualization. For simple structures, the system needs only a set of aesthetic choices to provide a useful graph—sometimes even a hand-drawn visualization could suffice. But for large, complex structures, effective layouts are harder to create, which motivates continual interest in graph layout algorithms as an integral part of visualizing complex networks. Although most traditional work involves developing more efficient layout methods for static graphs, more recent efforts have also focused on finding effective ways to generate dynamic graphs of time-varying networks.**

This paper discusses the various aesthetic criteria's which improve the readability of graphs and helps in how to choose proper layout algorithm for specific data to make the visualization better. This paper also discusses the different graph layout approaches which are used as the basis for developing many other new and improved graph layout algorithms helping in better visualization of graphs. Some easy-to-program network layout approaches are discussed here, with details given for implementing each one. This paper is mainly focused on the basic graph layout approaches which are used in "GUESS" the graph visualization and exploration software. This paper is also intended to beginners who are interested in programming their own network visualizations, or for those curious about some of the basic mechanics of graph visualization.

Index Terms— **Graph Visualization, Graph Layouts, Layout algorithms, GUESS.**

I. Introduction

Graph layout is concerned with placing a set of vertices and edges in the drawing in such a way that the reader of a graph can easily identify and understand the contents of the graph. When trying to place the vertices and edges of the graph, current graph layout methods aim to place the parts of the graph according to various aesthetic criteria to improve readability. A layout is a "good" layout if the layout achieves a majority of the aesthetic criteria. These aesthetics have been determined by research into the cognitive efforts of the human mind. Work by Ware et al. in [1] have tried to collate some important aesthetic criteria when laying out a graph. These include: 1) Minimal edge crossings, 2) Minimal edge bends, 3) Preservation of symmetry, 4) Nodes and edges should be evenly distributed in the drawing area, 5) Long edges should be avoided and deviation in edge lengths should be small, 6) Connected vertices should be close to each other, but the nodes shouldn't overlap each other or with edges in the layout, 7) The area used for drawing should be as small as possible, 8) The aspect ratio of drawing area is also be important, 9) Conform to the frame.

Himanshu Sharma, Computer Science & Engineering Department, Arya College of Engineering & Information Technology, Jaipur, Rajasthan, India, 9460867762, himanshu1182@gmail.com
Vishal Srivastava, Computer Science & Engineering Department, Arya College of Engineering & Information Technology, Jaipur, Rajasthan, India, 9214052387, vishal500371@yahoo.co.in

The aesthetic criteria mentioned are applied to a layout based upon various characteristics of a graph such as symmetry, hierarchical placement, etc. However there has not yet been one algorithm that has determined a set of aesthetics for graphs in all application domains. For example, a user may want a layout that preserves the symmetrical nature of a graph as shown in Fig. 1(a), however an algorithm that is more concerned with minimizing the number of edge crossings may layout the graph like that in Fig. 1(b) . The graph in Fig. 1(b) does not make the symmetry obvious to the reader, contrary to the wishes of the user. This is where the non-interactivity of graph layout algorithms can seriously detract from the quality and presentation of a final layout, and ultimately from what the users wants.

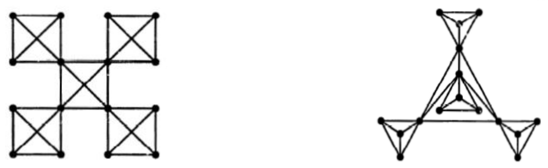
Fig. 1(a) Fig. 1(b)

Figure 1. A graph that is symmetrical (Fig. 1(a)), and the same graph with minimized edge crossings (Fig.1(b)). Available at [2].

Trying to achieve all these aesthetic criteria is considered to be difficult. Moreover, if you want to meet several criteria at the same time, an optimal solution simply may not exist with respect to each individual criteria because many of the criteria are mutually contradictory. Time-consuming trade-offs may be necessary. In addition, it is not a trivial task to assign weights to each criteria. Multicriteria optimization is, in most cases, too complex to implement and much too time-consuming. For these reasons, layout algorithms are often based on heuristics and may provide less than optimal solutions with respect to one or more of the criteria. Fortunately, in practical terms, the layout algorithms will still often provide reasonably readable drawings.

Various algorithms have been devised, and integrated into static layout tools, to calculate the best position of the vertices, and the routing of the edges in order to produce a readable graph, with attempts to achieve the aforementioned aesthetic goals. Some of the popular approaches for the graph layouts present in the GUESS [3]software are discussed in this paper. GUESS is a novel system for graph exploration that combines an interpreted language with a graphical front end that allows researchers to rapidly prototype and deploy new visualizations. GUESS also contains a novel, interactive interpreter that connects the language and interface in a way that facilities exploratory visualization tasks. GUESS use the Gython language for this purpose. The GUESS system is an attempt to combine analysis and visualization into one package that supports Exploratory Data Analysis (EDA) for graphs. It thus distinguishes itself from solutions that require one system to perform analysis, such as partitioning (e.g. using Analytic Technologies' UCINET), followed by a different program for rendering (e.g. Pajek [4] or GraphViz [5]). Interactive exploration is difficult when a user is forced to go back and forth between the analysis and the visualization packages. Moreover , GUESS software is easily available for download on internet and can be used free of cost.

II. Related Work on Graph Layout and Layout Algorithms

A lot of work has been done in the area of graph drawing, graph layouts and graph visualizations which includes the topics like aesthetics of graph drawing, various types of graph layout models for displaying of different types of graphs, generation of graphs, transition of graphs to a new state, how to visualize the graphs better, static and dynamic visualization of graphs etc. Some of the researchers who works on the above topics and related to the graph layout approaches discussed in this paper are as given.

Peter Eades with Giuseppe Di Battista, Roberto Tamassia, and Ioannis G. Tollis [6],[7] worked on Graph drawing and developed various algorithms for drawing graphs and algorithms for the visualization of graphs. In their work they discussed lots of the concept about the better graph drawing and visualization approaches. They also work on 3D drawings of graphs. P. Eades [11] has also worked on spring algorithms, maintenance of the "mental map" in dynamically changing drawings, heuristics for reducing the number of edge crossings in layered graph drawings, and visual display of clustering information in graphs. He is the first who use the concept of the combination of attractive forces on adjacent vertices, and repulsive forces on all vertices in spring algorithms. This concept still remains the base of various spring embedded algorithms with further continuous improvements. Kamada and Kawai[2], proposed a variation to Eades' Spring-algorithm. Their algorithm uses in its computations the desirable "geometric" (Euclidean) distance between two vertices in the drawing as the "graph theoretic" distance between them in the corresponding graph. T. M. Fruchterman and E. M. Reingold [21], also worked on graph drawing by force-directed placement by using the concept of temperature and cooling. They do not have the explicit concept of energy, and hence could not detect a stopping condition, they simply used 50 iterations every time. Going back to 1963, the graph drawing algorithm of Tutte [24], is one of the first force-directed graph drawing methods based on barycentric representation. R. davidson and D. Harel [20], worked on drawing nice looking undirected straight line graphs using simulated annealing. Arne Frick, Andreas Ludwig, Heiko Mehldau [26], proposed a fast adaptive layout algorithm for undirected graphs named as GEM. This algorithm include the concept of a local temperature,the attraction of vertices towards their barycenter and the detection of oscillations and rotations.

J. M. Six and I. G. Tollis [19], worked on the framework for circular drawings of networks and circular drawings of biconnected Graphs, and also developed a visualization tool named Vistool. Peter Eades [6],[7] discussed the concept and algorithms of radial layout for better space utilization. Ka-Ping Yee, Danyel Fisher, Rachna Dhamija, and Marti A. Hearst [14], worked on animated exploration of dynamic graphs with radial layouts. Classical MDS, was first introduced by W.S. Torgerson [53]. Further works are carried out by J. B. Kruskal [30],[36] who discussed the concept of classical, metric and nonmetric multidimensional scaling. R.N. Shepard [37],[38],[51] analyses the proximities in multidimensional scaling with an unknown distance function. I. Borg and P. Groenen [34], worked on the theory and applications of modern multidimensional scaling. B. Bollobas [55], worked on concept of random graphs. The systematic study of random graphs was started by P. Erdos and A. Renyi [56]. They discussed on the evolution of random graphs [57]. S. Janson, T. Luczak, A. Rucinski [58], also worked on random graphs. They give various algorithms, theorems, functions and other concepts about random graphs.

III. Present Work: Analysis of Different Graph Layout Approaches Present in GUESS.

Graph or network-like structures are among the most commonly used types of visual representations. Automatic layout of graphs plays an important role in many applications like, Visual Programming, Software Engineering (e.g. Flow-Charts, UML, Dependency visualization, Repository Structures), Engineering (e.g. Circuit Diagrams, Molecular Structures, Chemical Formulas), Sciences (e.g. Genome Diagrams) and currently a particularly hot-topic, Web-Visualization. Almost always when relational data that has been obtained as the result of an automated operation such as a database or repository computation, a web-crawl or any other kind of computation, it has to be visualized, we are facing the problem of automatic graph layout.

In this paper, some of the basic graph layout approaches, their features, advantages and drawbacks are discussed. The research in this paper gives an overall analysis of the present visualization techniques, which is vital to the future development of related algorithms and technologies.

IV. Different Graph Layout Approaches in Guess

The approaches discussed here are basis of graph drawing and is used in many graph visualization tools. This paper is mainly focused on the basic graph layout approaches which are used in "GUESS" the graph visualization and exploration software**.**

A. Circular Layout Approach (Drawing on a single embedding circle)

A circular drawing of a graph (see Fig. 2 for an example) is a visualization of a graph with the following characteristics:

1) The graph is partitioned into clusters,
2) The nodes of each cluster are placed onto the circumference of an embedding circle,
3) Each edge is drawn as a straight line, and
4) A node cannot be occluded by another node or by an edge.

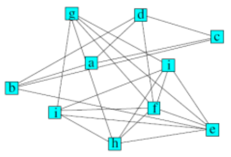

Fig. 2(a)

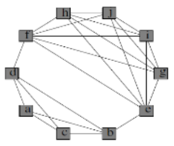

Fig. 2(b)

Figure 2. A graph with arbitrary coordinates for the nodes (Fig. 2(a)) and a possible circular drawing of the same graph (Fig. 2(b)). Available at [19].

Simple circular layout is based upon the various parameters to be considered at the time of layouting. The parameters like ordering of the nodes decides the crossing of edges in the circle, the size of every node decides the radius and circumference of a circle. The distance between the adjacent nodes should be symmetric. These all parameters are handled by only tweaking the following calculations for every node as:-

LaidOutX = CenterX + Radius * cosine(Angle).
LaidOutY = CenterY + Radius * sine(Angle).

To shift center of any cluster you need to modify the center coordinates in the above models. Adjusting of the spreading between the nodes will be added by incrementing the radius in the above models. Next node will be placed at a deviation of angle used in the above models. In this way tweaking is performed in the circular layout by modifying the above mathematical models.

The circular layout is mainly used for small-medium sized graphs data analysis and in the applications where that application requires the visualization of information having more emphasis on the aesthetics of drawing. In general, these layouts can provide a compact presentation, focusing on individual nodes and edges. Additionally, well-designed circular layouts sometimes reveal global properties of the graph such as symmetries and patterns of collective behavior. Other advantage of a circular layout which can be seen in some of these applications, such as bioinformatics or social network visualization, is its neutrality by placing all vertices at equal distances from each other and from the center of the drawing, none is given a privileged position, countering the tendency of viewers to perceive more centrally located nodes as being more important

These circular layouts have some limitation and problems. An inherent problem with circular layouts is that the rigid constraint on node placement often gives rise to long edges and an overall dense drawing. Circular layouts shows symmetric property, this strong regularity can also obscure other information as these drawings can be very dense, and following paths on them can be difficult. In these layouts, as the nodes need to arranged on the circumference, it restricts the area for the nodes to spread. This may incur in crossing between the edges and hence we cannot minimize the crossings beyond a certain extent. The ordering of the nodes is also a critical dynamic problem which needs to solve in the circular layout. Hence to resolve this problem, the topological ordering with respect degree of a nodes can be used . This ordering will select the highest degree node first and will place all adjacent node according to the next immediate node and applies the circle to all nodes in the ordering list. Calculation of radius for the circle is also a difficult problem considering the size parameters of the node which is variable for every node entity in the graph. Boundaries of clusters need to be calculated considering the size parameters of the graph entities.

The use of circular layouts are a good fit for communications network topologies such as star or ring networks, and for the cyclic parts of metabolic networks.

B. *Improved Circular Layout Approach*

An inherent issue with circular layouts is that the rigid restriction on node placement often gives rise to long edges and an overall dense drawing. In improved circular layouts, three independent, complementary techniques are used for lowering the density and improving the readability of circular layouts.

1) First technique, Places the nodes on the circle such that edge lengths are reduced. This is accomplished with a new algorithm.

2) Second technique, Enhances the circular drawing style by allowing some of the edges to be routed around the exterior of the circle. This is accomplished with an algorithm for optimally selecting such a set of externally routed edges. External routing can be very effective in reducing edge crossings. Since exterior routing of an edge is inherently longer than interior routing, utilize the exterior routing carefully, and make sure that edges routed externally are readable. Therefore, if possible do not allow any edge crossing within the external face. Two edges cross in the external face if and only if they cross internally.

3) Third technique reduces density by coupling groups of edges as bundled splines that share part of their route. This frees up drawing area without compromising structural clarity. Considering non-straight line edges opens up even more possibilities for better clarity.

Together, these techniques are able to reduce clutter, density and crossings compared with existing methods. This method also reduces the edge lengths. Figure 3 shows implementation of these techniques one by one and the final Fig. 3(e) shows combining effect of all these techniques on layout in Fig. 3(a). The final Fig. 3(e) has much better clarity and visualization.

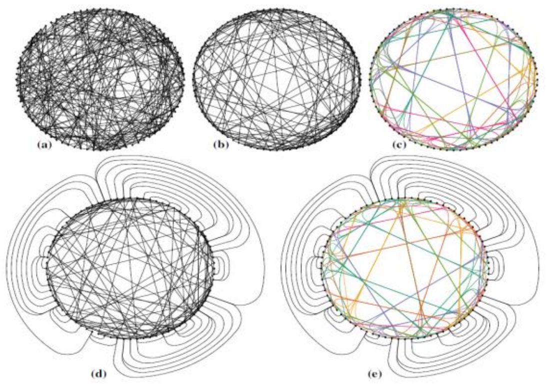

Figure 3. Variations on circular layouts of a random graph (|V | = 80,|E| = 241)
Fig. 3(a) random order;
Fig. 3(b) edge-length minimizing order;
Fig. 3(c) bundling edges to save ink and to improve area utilization (colors used to enhance readability);
Fig. 3(d) exterior routing lessens crossings and alleviates density;
Fig. 3(e) combining exterior routing with edge bundling.

This drawing convention is mainly used for small and medium sized networks. It can be used for the layout of networks and systems management diagrams, where it naturally captures the essence of ring and star topologies. It can be also used for other kinds of graphs, such as social networks, metabolic networks and WWW graphs. The usually unvisualized characteristics of self-organization, emergent structures, knowledge exchange, and network dynamics can be seen in the these drawings. Resource bottlenecks, unexpected work flows, and gaps within the organization are clearly shown in these circular drawings.

This drawing approach is a variant of circular drawing, so many disadvantages are same as of circular layouts like as the nodes need to arranged on the circumference, it restricts the area for the nodes to spread, it has long edges and exterior routes, overall dense drawing, cannot minimize the crossings beyond a certain extent, computation of radius of the circle etc.

C. Radial Layout Approach

Radial layout graph drawing algorithms are a well known method for creating drawings of rooted trees [6, 7, 8, 9]. In radial layout approach, the root vertex of a tree is positioned at the center of the drawing with descendant vertices situated on concentric circles emanating from it as shown in Fig. 4.

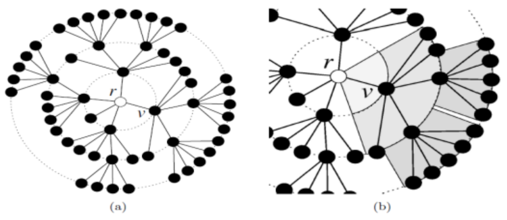

Figure 4. A radial layout graph drawing of a tree rooted at r Fig. 4(a), and the division levels of *v*'s annulus wedge Fig. 4(b).

For a tree T with a height *k*, the layers of concentric circles are labeled as C_1,C_2......, C_k and each vertex is placed on circle C_i, where *i* is the depth of the vertex in the rooted tree [20]. Using a defined heuristic, a radial layout drawing algorithm allocates each vertex a space in the drawing, known as its annulus wedge. This wedge confines the layout of a vertex's subtree to particular area in the drawing. A vertex's annulus wedge is divided among its descendants at subsequent levels in the subtree (Fig. 4(b)).

For a tree T rooted at a vertex *r*, a new radial layout graph drawing T' is obtained as follows. The algorithm first places the root vertex at the center of the viewing plane and allocates it an annulus wedge of the entire drawing (360^0). This space is divided among the root's descendants: the annulus wedge for a child vertex *c* of *v* is based on the number of leaf vertices in the subtree rooted at *c* proportional to the number of leaf vertices in the subtree rooted at *v*. Each vertex is placed at the center of its annulus wedge on a concentric circle corresponding to its depth in the tree. The algorithm continues down each subtree until positions for all of the graph's vertices in the new drawing T' are calculated. A new view of the network is constructed every time the user selects a new focal point vertex.

The advantage of the radial layout approach are: First one is the root node is taken as the center of graph so it helps to get focus on a particular node and its relations. Second, since the length of each orbit increases with the radius, there tends to be more room for the nodes. A radial tree will spread the larger number of nodes over a larger area as the levels increase. Third, the radial tree graph also solves the problem of drawing a tree so that nodes are evenly distributed. Radial layouts are used to display large hierarchies networks.

However, like with any graph drawing algorithm, the drawings generated by radial layout algorithms do have some drawbacks. The number of nodes increases exponentially with the distance from the first node, whereas the circumference of each orbit increases linearly, so by the outer orbits, the nodes tend to be packed together. Although the use of concentric circles does make it easier for users to ascertain the depth of a vertex in a tree, these circles confine vertices to positions that may not be optimal and can make it difficult for users to visually distinguish siblings from their parent. This is because sibling vertices may be spread out widely on their corresponding circle, and thus the lengths of the edges to their parent are dramatically different.

For example, the drawings in Figure 5 depict the same tree from Figure 4 but based on a different root vertex. In Figure 5(b), the edges marked by arrows illustrate edges for sibling vertices that are different in length. Another problem is that radial layout drawings can still allow edge crossings, even if the graph is planar (Figure 5(c)). These flaws can degrade the readability of a graph.

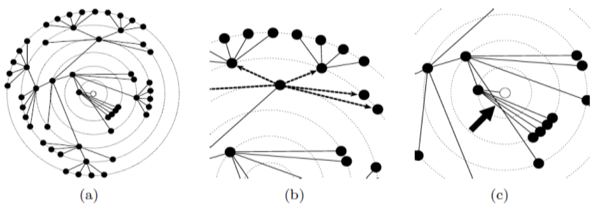

Figure 5. A radial layout graph drawing of the same tree from Fig. 4(a), but with a different root vertex as the focal point (Fig. 5(a)). Fig. 5(b) illustrates how sibling vertices have variable edge lengths to their parent, and Figure 5(c) highlights edge crossings in the drawing.

The radial layout is suitable for large and medium-sized data analysis like to identify the center with complex and large number of acts in a large database, typically telephone message declarations, e-mail between records and financial transactions, etc. Radial tree graph drawing algorithms are used for representing large hierarchies, business relation between families, marital ties between families, directory structure of websites etc.

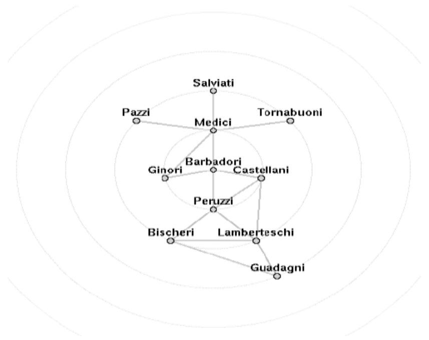

Figure 6. Showing business relations between families.

D. Context-Free Radial Layout Approach

Context-Free Radial Layout drawings of graph adhere to three aesthetic goals: (1) minimize the number of edge crossings (edges are drawn as straight- line segments), (2) minimize the total angular difference between the root's children's positions from the initial drawing to the new drawing, and (3) maximize the angular resolution of parent-child edges. These drawings conform to two constraints: (1) the root vertex is placed at the center of the drawing, and (2) vertices are equidistant to their parent vertex in the tree. This approach is designed to prevent two specific types of edge crossings: (1) edge crossings between sibling vertices, and (2) crossings between the edge of a vertex to its parent with one of its edges to its children.

This graph visualization system allows users to explore the structure and properties of a graph via multiple spanning -tree-based drawings. Given a graph G, a user-selected vertex *r* ϵ*V*(G), and an initial graph drawing T' for G, this graph drawing algorithm generates a new drawing of G based on a spanning tree rooted at *r* extracted from G. This algorithm is called as "context free" because the placement of children, relative to the frame of reference of the parent, only depends on the parent's position.

This visualization system creates a new drawing for a graph G with an initial drawing T' in three stages: First it extracts a spanning tree T rooted at *r* from G using breadth- first search. Second, by using this tree T, the system calculates the graph's vertices relative polar coordinates from their positions in T'. Based on these initial positions, the system then calculates the vertices relative polar coordinates in the new drawing. Third, Instead of using concentric circles where one circle is used for positioning all the vertices for a given depth in the tree, this algorithm creates drawings using a series of overlapping circles that we call *containment* circles. Each non-leaf vertex *v* ϵ *V* (T) is given its own containment circle centered at *v* and only *v's* children are positioned on this circle. Figure 7(a) to 7(f) shows how a radial layout is generated from a normal graph. Figure 8(a) to 8(f) shows the generation of a new radial layout from Fig. 7(f) by selecting a new vertex as root. Figure 9(a) to 9(h) shows the generation of a graph with Context-Free Radial Layout Approach in detail.

This approach enables the drawing algorithm to position sibling vertices close together and emphasize the parent-child relationships in the tree much better than other layouts. It also helps to see the graph with single user perspective by making it root. These layouts has high-level layout space utilization in the same proportion, with clearer hierarchy structure and less edge crossings, and the clarity of clustering performance is better. Also edge lengths decrease monotonically with distance from root, and all siblings within a family are arrayed along visually salient arcs equidistant from their common parent. With regard to animating transitions, this approach ensures that sibling edges never cross when a new focal node is selected, and whenever the graph to be drawn is itself a tree.

However, like with any graph drawing algorithm, the drawings generated by context free radial layout algorithms have the drawback that with every next level of *containment* circles the radius of circle becomes half of the previous level circle, so remote descendants of the root can become vanishingly small on the viewing plane. Also the edge crossings can occur when long subtrees encroach on neighboring containment circles.

Context free radial graph drawing algorithms are used for representing large hierarchies, business relation between families, marital ties between families, directory structure of websites etc.

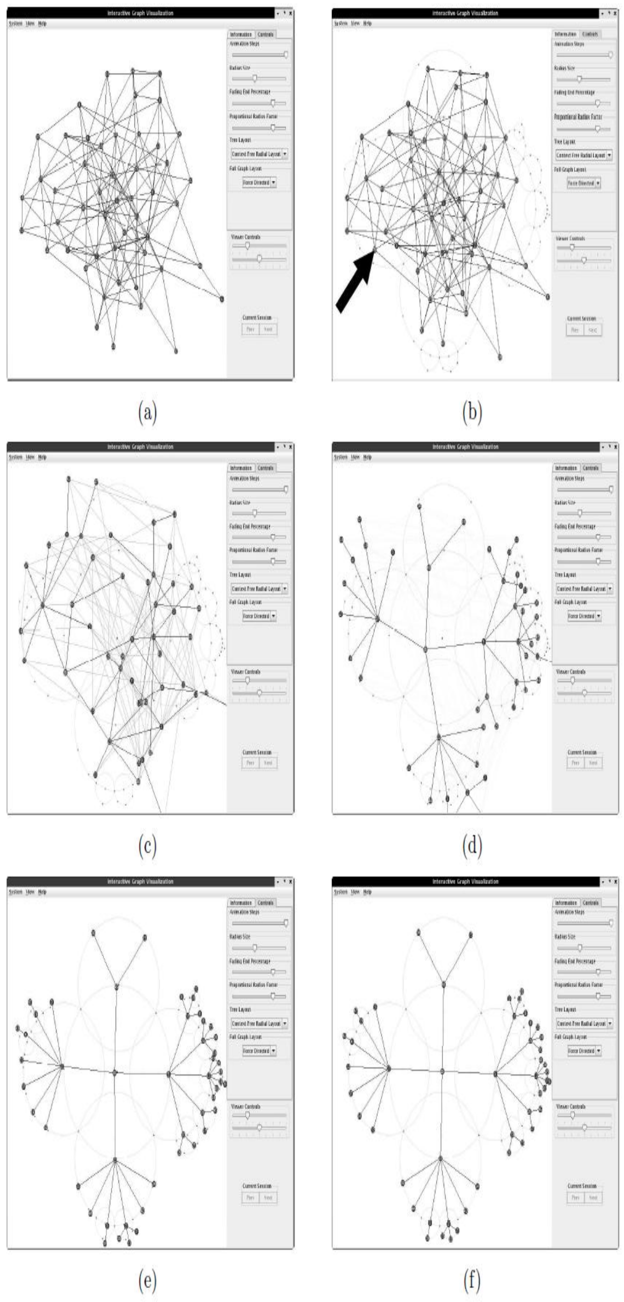

Figure 7. This graph visualization system first generates a force-directed layout drawing of a graph with 50 vertices (Fig. 7(a)). A user then selects a vertex (indicated by the arrow) to become the root of a new spanning-tree based drawing for the graph (Fig. 7(b)). The movement of the graph's vertices and edges is animated as the visualization system transitions from the original drawing to the new drawing (Fig. 7(c) to Fig. 7(e)). The animation sequence is complete when the vertices reach their final positions in the new layout (Fig. 7(f)).

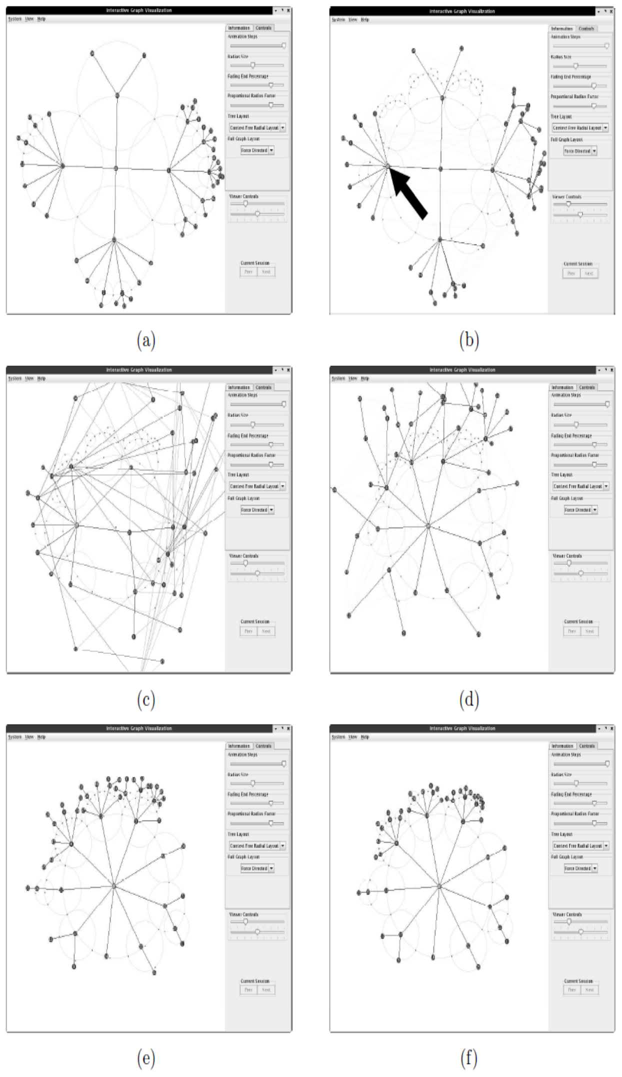

Figure 8. Using the same spanning tree drawing from Fig. 7(f), a user selects a different vertex in the graph indicated by the arrow to become the root of a new spanning-tree-based drawing (Fig. 8(b)). The movement of the graph's vertices and edges is animated as the visualization system transitions from the original drawing to the new drawing (Fig. 8(c) to Fig. 8(e)). The animation sequence is complete when the vertices reach their final positions in the new layout (Fig. 8(f)).

Working of a context free radial graph drawing approach

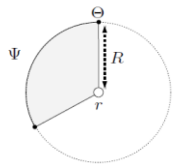

Fig. 9(a)

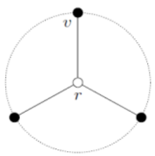

Fig. 9(b)

In Figure 9(a), the root is first placed at the center of the drawing along with its containment circle with a radius of R. The root's annulus wedge is divided into three equal portions of size Ψ and its first child is positioned at Θ.

In Figure 9(b), the root's children are positioned on its containment circle.

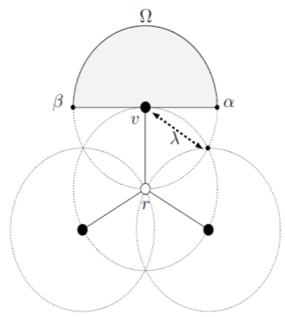

Fig. 9(c)

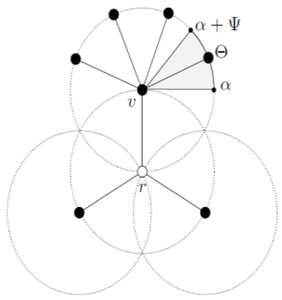

Fig. 9(d)

Figure 9(c) each of the root's children is allocated a separate containment circle with a radius of λ. The algorithm then allocates space in the drawing to position *v*'s children. *v*'s annulus wedge of size **Ω** is centered on the arc of *v*'s circle that is outside of the root's circle. The angles α and β are relative to *v*'s position

Figure 9(d), *v*'s annulus wedge is divided into four equal parts of size Ψ. Each child of *v* is positioned in the center of one *v*'s annulus wedge subdivisions starting at Θ.

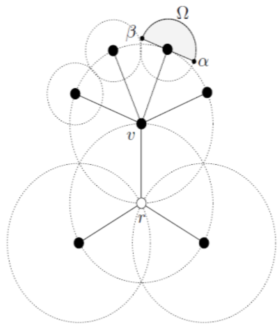

Fig. 9(e)

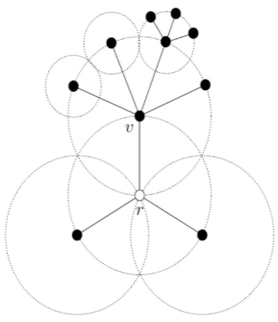

Fig. 9(f)

In Figure 9(e) and Figure 9(f), the drawing algorithm continues down the subtree rooted at *v* allocating containment circles and annulus wedges for descendant vertices.

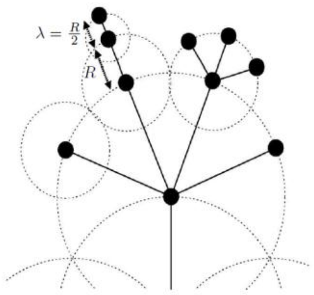

Fig. 9(g)

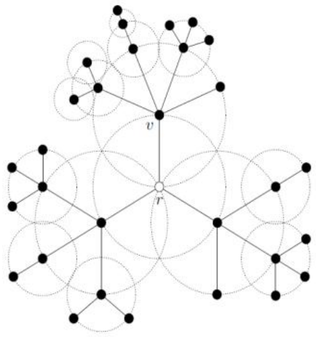

Fig. 9(h)

Figure 9(g), the radius λ is half the size of that vertex's parent's containment circle.

Figure 9(h),The algorithm positions the rest of the graph's vertices, resulting in the final drawing of graph.

Figure 9. The above diagram illustrates how context free radial graph drawing approach works and constructs a new drawing for a tree T rooted at r.

E. Spring layout

In the drawing of undirected graphs there is a problem of "too much freedom". The spring algorithm (Eades [7], 1984) is a successful layout creation algorithm for drawing an undirected graph. The basic idea of the spring algorithm is to treat a graph as a mechanical system, in which nodes are replaced by steel rings and edges by springs connected to the rings. All the springs have the same *natural length k*, and each spring has a *current length d*. Given a pair of rings connected by a spring, if *k*>*d* (spring is stretched), then the spring attracts the rings; if *k*<*d* (spring is compressed), then the spring repulses the rings; if *k*=*d* (Zero force & Zero energy), then the rings are stable. The spring forces will attract or repulse the rings until the system reaches the *minimum energy*. This is called the *balanced state*. The strength of the forces is determined by *k* and *d*. Eades[7] calculates the forces using the functions:

$$f_a = C_1 * log(k/C_2)$$

$$f_r = C_3/sqr(d)$$

where f_a is the attractive force, f_r is the repulsive force, and C_1, C_2, C_3, C_4 are coefficients. The Simple Algorithm given by Eades [7] is

Algorithm SPRING(G:graph);

```
Place vertices of G in random locations;
Repeat M times
Calculate the force on each vertex;
Move the vertex $C_4$ *(force on vertex)
Draw graph on CRT or plotter.
```

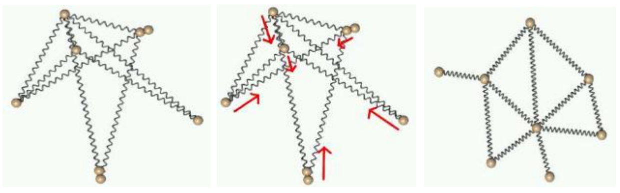
Fig. 10(a) Fig. 10(b) Fig. 10(c)

Figure 10. Illustration of a generic spring embedder: starting from random positions, treat the graph as spring system and look for a stable configuration (Fig. 10(a) to Fig. 10(c)).

The input of a spring algorithm is an undirected graph, and the output of the algorithm is a drawing of the embedded graph. The running time of a spring algorithm is $O(n^2)$, where *n* is the number of the nodes. The layout generated by a spring algorithm has two features: display of symmetry and uniform edge length. A spring system is not guaranteed to generate a good drawing for all graphs. The forces exerted on some rings might be too strong or too weak, and the spring system cannot arrive at the *balanced state*. One solution is to adjust the coefficients; another is to adopt different force models. Some implementations, however, also set a fixed number of iterations, in order to end the spring forces in a predictable time. There are some variants of the spring algorithm that adopt different force models. All of these algorithms are broadly called force-directed algorithms in most literature.

F. *Spring layout (Variant having spring system and electrical charges)*

Force-directed methods use analogies from physics to compute the positions of a graph's vertices. Consider an undirected graph G = (V, E) as a system of bodies (V), influenced by forces (E). The layout algorithm's goal is to find a configuration of those bodies where the sum of all forces is minimized, i.e. it assigns every vertex to a position
such that the sum of forces which influence this vertex is 0.

The model in the spring layout uses a combination of springs and electrical forces. The edges correspond to springs. The vertices correspond to equally charged bodies which experience mutually repulsive forces in between them. In the following, we denote vertices by *u*, *v* ϵ *V* , and by $\vec{u}, \vec{v} \in$ R their position vectors in the two-dimensional space. The length of the difference vector $\vec{v} - \vec{u}$ is denoted by

$$||\vec{v} - \vec{u}|| = \sqrt{(x_v - x_u)^2 + (y_v - y_u)^2},$$

i.e. the euclidian distance of the points (x_u, y_u) and (x_v, y_v). Moreover, $\vec{e}_{uv}$ denotes the unit vector

$$\vec{e}_{uv} = \frac{\vec{v} - \vec{u}}{||\vec{v} - \vec{u}||}$$

which points from $\vec{u}$ to $\vec{v}$.
The repulsive force that *v* experiences from *u* obeys the Coulomb's law equation

$$\vec{F}_0(u, v) = \frac{c_0}{||\vec{v} - \vec{u}||^2} \cdot \vec{e}_{uv},$$

where c_0 ϵ R is a constant which determines the strength of the repulsive force at either one of the vertices *u* or *v*. The spring force experienced by *v* via the spring from *u* can be computed by the Hooke's law equation.

$$\vec{F}_1(u, v) = -c_1(||\vec{v} - \vec{u}|| - l) \cdot \vec{e}_{uv},$$

where the constant c_1 ϵ R denotes the strength of the spring and *l* ϵ R denotes its natural length. The global force that is acting upon a vertex *v* ϵ *V* , can be computed as follows:

$$\vec{F}(v) = \sum_{u \in V} \vec{F}_0(u, v) + \sum_{(u,v) \in E} \vec{F}_1(u, v).$$

Working of this algorithm is as follows:

Vertices, which are not yet at equilibrium of forces, experience a force. To relax the system, vertices are iteratively moved towards the direction of their influencing force. At time *t*, a vertex *v* experiences the force $\vec{F}_t(v)$. After these forces are computed for all vertices at time *t*, each vertex *v* is moved by $\delta \cdot \vec{F}_t(v)$ in its respective direction.
The constant $\delta,\ 0 < \delta < 1,$ avoids that the vertices are moved too far. The simplest option to terminate the algorithm is to stop after a certain number of iterations. Another option is to compute the global force of the system at a time *t*. This can be done as follows:

$$F_{\text{global}} = \sum_{v \in V} |\vec{F}_t(v)|.$$

If F_{global} reaches/falls below a certain value (ideally, F_{global} = 0), then the computation can be stopped.

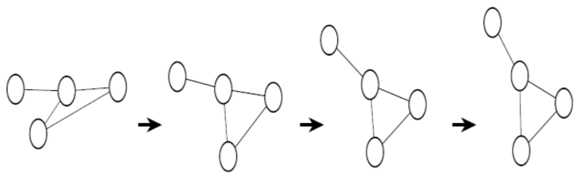

Figure 11: Process showing how nodes of a graph in the spring embedder algorithm reaches to equilibrium position (Fig. 11(a) to Fig. 11(d)).

In order to compute the force $\vec{F}(v)$ of a vertex $v \in V$, the sum $\sum_{u \in V} \vec{F_0}(v,u)$ is computed. This computation is done $n(n-1) = O(n^2)$ times per iteration. Vertices, which are far apart, only experience a very weak repulsive force from each other. Because of this, one can neglect these distant vertices in the computation of the repulsive force. The force $\vec{F_0}(v,u)$ is computed only for vertices *u* whose distance to *v* is $||\vec{v} - \vec{u}||^2 < d^2$ (for a proper constant d ϵ R).

The advantage of spring approach is that it gives relatively good results in a simple and intuitive manner. The spring system is a fast in speed and usually converges after 100 iterations or so. It avoids node overlapping and edge crossings. It gives the symmetric layouts as output.

The disadvantage is that the given algorithm only works for connected graphs and for using it on disconnected graphs additional exercise is to be done. The other drawbacks of the approach are its running time complexity, possibility of convergence to a poor local minima and do not have a good termination condition.

Spring models are generally not used for large networks because there is a good chances that system is likely to settle in a local minimum. They are used to visualize small size networks. Spring models can be used to visualize small and medium social networks (Fig. 12). Moreover with some modifications can be used in bioinformatics including phylogenetic trees, protein-protein interaction networks, and metabolic pathways.

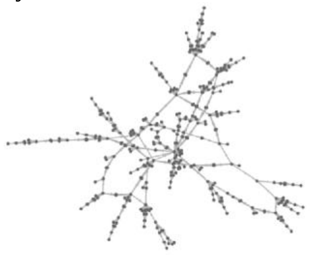

Figure 12. A Small Sized Social Network

G. *Fruchterman and Reingold Layout (Based on Electrostatic forces & Temperature concept)*

Fruchterman and Reingold [21] replace the spring embedder's repelling and attracting forces between pairs of vertices by repelling forces

$$f_{\mathsf{rep}}(x_u, x_v) = \frac{l^2}{d(x_u, x_v)}$$

between every pair of vertices *u* and *v*, and additional attracting forces

$$f_{\mathsf{attr}}(x_u, x_v) = \frac{d(x_u, x_v)^2}{l}$$

only between adjacent vertices. Parameter *l* describes the optimum length of a single edge, and is set to *c. Sqrt* (c_A / n), where c_A is the desired layout area, and *c* is an experimentally chosen constant. These functions are cheaper to evaluate and steeper, thus resulting in faster convergence. The algorithm to find a stable configuration features an adaptive parameter controlling the maximum displacement allowed.

We have only two principles for graph drawing in this Layout:

1) Vertices connected by an edge should be drawn near each other.
2) Vertices should not be drawn *too* close to each other. How close vertices should be placed depends on how many there are and how much space is available. For placing nodes we calculate the optimal distance *k* between vertices. Some graphs are too complicated to draw attractively at all.

There are three steps to each iteration:

1. Calculate the effect of attractive forces on each vertex,
2. Then calculate the effect of repulsive forces, and
3. Finally limit the total displacement by the temperature and cooling concept.

Temperature indicate the maximum distance a vertex can travel when being updated. The idea is that the displacement of a vertex is limited to some maximum value, and this maximum value decreases over time (cooling); so, as the layout becomes better, the amount of adjustment becomes smaller and smaller.

A special case occurs when vertices are in the same position: our implementation acts as though the two vertices are a small distance apart in a randomly chosen orientation: this leads to a violent repulsive effect separating them.

We calculate *k*, the optimal distance between vertices as

$$k = C\sqrt{\left(\frac{\text{area}}{\text{number of vertices}}\right)}$$

where the constant *C* is found experimentally. We would like the vertices to be uniformly distributed in the frame, and *k* is the radius of the empty area around a vertex. If *fa* and *fr* are the attractive and repulsive forces, respectively, with *d* the distance between the two vertices, then

$$fa(d) = d^2/k$$

$$fr(d) = -k^2/d$$

Figure 13. illustrates these forces and their sum versus distance. The point where the sum of the attractive and repulsive force crosses the *x* -axis is where the two forces would exactly cancel each other out, and this is at *k*, the ideal distance between vertices.

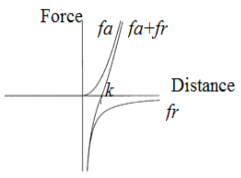

Figure 13. Forces versus distance

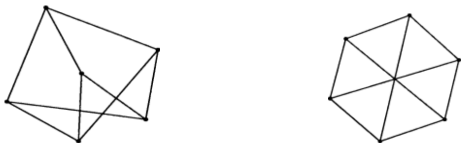

Figure 14. Fig. 14(a) Showing a graph. Fig. 14(b) Shows same graph of Fig. 14(a) after applying Fruchterman and Reingold algorithm. It is more symmetrical and balanced than previous one. Available at [21].

This approach is simple, intuitive, and interactive. This approach can produce good quality results for small and medium sized graphs. The results are good in terms of uniform edge length, uniform vertex distribution and showing symmetry. This approach can be easily adapted and extended and have flexibility due to which can be used in 3D graph drawing, cluster graph drawing, constrained graph drawing and dynamic graph drawing. The primary advantage of this approach is speed.

The disadvantage of this approach is poor local minima. Large graphs have many local minima's so there is good chances that system is likely to settle in a local minimum. Second, because this approach had no explicit concept of energy, and hence could not detect a stopping condition, so simply 50 iterations every time are used; this was excessive on the simpler graphs. Third is for a given vertex, repulsive force from all other vertices needs to be calculated, this makes the per iteration cost of the algorithm (O$|V^2|+|E|$) with |V| the number of vertices and |E| number of edges in the graph for basic approach. This approach becomes slow as the networks grow.

This approach can be used to visualize small and medium social networks like organizational network, communication networks, collaboration networks. It can be used to visualize information networks like WWW hyperlinks, blog networks, citation networks. It can be used to visualize technological networks like power grid, airline, road, river, telephone networks. It can also be used to visualize biological networks like metabolic networks, food web, neural networks etc.
It can also be used in visualize the tree structures and 3D drawings.

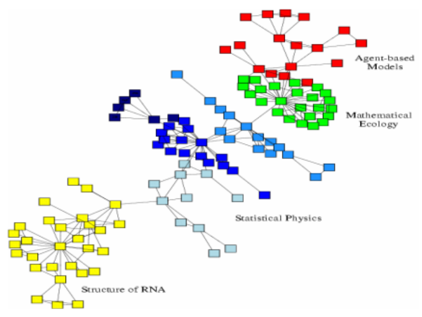

Figure 15. Showing a collaboration network

H. GEM Layout

GEM is short form for *graph embedder*. GEM layout algorithm contains the concept of a local temperature, the attraction of vertices towards their barycenter and the detection of oscillations and rotations. The major design goal was that interactive speed should be achieved even for medium-sized graphs. We consider a drawing to be interactive if it takes less than 2s to compute. Randomization plays an important role in several places of the algorithm.

The discussion of the GEM algorithm starts with the observation that cooling schedules appear to give better results than methods relying solely on a gradient descent, but their running time is unsatisfactory. Temperatures as used in GEM indicate the maximum distance a vertex can travel when being updated. The temperature scale has a direct influence on a suitable choice of other parameters, i.e. the constants used in the formulae for the attractive and repulsive forces.

Rather, the algorithm adapts to the data locally and does not require global cooling as assumed by a schedule. For each vertex, a *local temperature* is defined that depends on its old temperature and the likelihood that the vertex oscillates or is part of a rotating subgraph. Local temperatures raise if the algorithm determines that a vertex is probably not close to its final destination. The *global temperature* is defined as the average of the local temperatures over all vertices. Thus, it indicates how stable the drawing of the graph is. This algorithm consists of two stages, an initialization stage and an iteration stage. The initialization consists of the assignment of an initial position, impulse and temperature to each vertex. The main loop updates vertex positions until the global temperature is lower than a desired minimal temperature or the time allowance has expired.

It is found in practice that most graphs would easily cool down to T_{min}, we cannot exclude the possibility of a graph moving chaotically between rounds. This can be solved by choosing T_{min} larger. Vertices are moved sequentially according to a *choice function*. Assuming that vertex *v* was chosen to be updated, the attractive and repulsive forces acting on *v* are computed. In addition, a *gravitational force* pulling the vertex towards the barycenter of the vertex cluster is assumed. The use of gravitation accelerates the convergence of GEM. In addition, it helps to keep disconnected graphs and loosely connected components together.

The resulting force is scaled with *v*'s current temperature to form the *impulse* of *v* such as to reflect the algorithm's "knowledge" of the state of computation. A low temperature indicates either that the layout is almost stable (at least locally) or that there exist oscillations or rotations. In each case, movements should be short. GEM has the ability to leave wells containing local energy minima, since local temperature increases change the global energy distribution. Relative to the old distribution, uphill moves become possible. Unfortunately, this feature makes proofs of convergence hard, if not impossible.

The advantages of this layout are as follows: This algorithm achieves drawings of high quality on a wide range of graphs with standard settings. The algorithm is fast (is significantly faster than Kamada and Kawai [2] or Fruchterman and Reingold Layout [21] algorithms), being thus applicable on general undirected graphs of substantially larger size and complexity. The GEM layout approach is optimal to layout cyclic as well as acyclic graphs. Aesthetically pleasing solutions are found in most cases. Also, the additional attractive force has two important effects: unconnected and loosely connected components are not separated too far, and it may lead to a 30% increase in convergence speed.

The one drawback is that the output of the algorithm is often depends on the original layout of the graph. It will be more jumbled if the underlying graph is very jumbled. It may at times help to first provide one of the other algorithms, which could put the vertices in a slightly better order, and then apply the GEM layout algorithm. The other disadvantage is of oscillating nodes. In some cases, a node may not find its final position even after more than 100 iterations, and will alternate between several positions indefinitely. The main reason for this phenomenon is that the movement calculation is not continuous, but discrete.

The GEM approach can be used for somewhat larger graphs compared to Kamada and Kawai [2] approach and Fruchterman and Reingold [21] approach. It can be used on different types of somewhat complex graphs and trees. This approach can be used to visualize medium and large sized undirected social networks, information networks, technological networks, biological networks etc. It can also be used in visualize the tree structures and 3D drawings.

I. Kamada Kawai Layout (Spring model based on Euclidean distance)

The 1989 algorithm of Kamada and Kawai [2], introduced a different way of thinking about "good" graph layouts. Whereas the algorithms of Eades [7] and Fruchterman-Reingold [21] aim to keep adjacent vertices close to each other while ensuring that vertices are not too close to each other, Kamada and Kawai [2] take graph theoretic approach.

In this model, the "perfect" drawing of a graph would be one in which the pair-wise geometric (Euclidean) distances between the drawn vertices match the graph theoretic pair-wise distances, as computed by an All-Pairs-Shortest -Path computation. As this goal cannot always be achieved for arbitrary graphs in 2D or 3D Euclidean spaces, the approach relies on setting up a spring system in such a way that minimizing the energy of the system corresponds to minimizing the difference between the geometric and graph distances. In this model there are no separate attractive and repulsive forces between pairs of vertices, but instead if a pair of vertices is (geometrically) closer/farther than their corresponding graph distance the vertices repel/attract each other.

In this approach a dynamic system is considered in which n (= $|V|$) particles are mutually connected by springs. Let p_1, p_2, ... , p_n be the particles in a plane corresponding to the vertices $v_1, v_2, \dots, v_n \in V$ respectively. This approach relate the balanced layout of vertices to the dynamically balanced spring system. As a result, the degree of imbalance can be formulated as the total energy of springs, i.e.,

$$E=\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\tfrac{1}{2}k_{ij}(\,|\,p_i-p_j\,|-l_{ij})^2. \tag{1}$$

Pleasing layouts can be obtained by decreasing E, then the best layout is the state with minimum E in this model. The original length l_{ij} of the spring between p_i and p_j corresponds to the desirable length between them in the drawing, and is determined as follows. The distance d_{ij} between two vertices v_i and v_j in a graph is defined as the length of the shortest paths between v_i and v_j. Then the length l_{ij} is defined as

$$l_{ij} = \mathrm{L} \times \mathrm{d}_{ij}$$

where L is the desirable length of a single edge in the display plane. When the display space is restricted, it is a good way to determine L depending on the diameter (i.e., the distance between the farthest pair) of a given graph. That is,

$$L = L_0 / \max_{i<j} \mathrm{d}_{ij}$$

where L_0 is the length of a side of display square area. The parameter k_{ij} is the strength of the spring between p_i and p_j , and is determined as follows. The expression (1) can be regarded as the square summation of the differences between desirable distances and real ones for all pairs of particles. From this point of view, the differences per unit length is better to be used in (1). Then, k_{ij} is defined as-

$$k_{ij} = K/d_{ij}^2$$

where K is a constant. The parameters l_{ij} and k_{ij} are symmetric, i.e., $l_{ij} = l_{ji}$ and $k_{ij} = k_{ji}$ $(i \neq j)$. In this spring model, the density of particles does not become large, because every two nodes are forced to keep certain distance by the tension of a spring. Note that symmetric graphs correspond to symmetric spring systems, which result in symmetric layouts by minimizing E.

Assume, the position of a particle in a plane is expressed by x and y coordinate values. Let (x_1, y_1), (x_2, y_2),... (x_n, y_n). be the coordinate variables of particles $p_1, p_2, \dots, p_n$ respectively. Then, the energy E defined as (1) is rewritten by using these $2n$ variables as follows.

$$E=\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\tfrac{1}{2}k_{ij}\left\{(x_i-x_j)^2+(y_i-y_j)^2+l_{ij}^2-2l_{ij}\sqrt{(x_i-x_j)^2+(y_i-y_j)^2}\right\}. \tag{2}$$

The purpose is to compute the values of these variables which minimize $E(x_1,x_2,\cdots,x_n,\ y_1,y_2,\dots.y_n)$. (Here after the parameters of the function E are omitted). It is, however, quite difficult to compute the minimum, so instead of it a local minimum is computed. A method is used for computing a local minimum of E from a certain initial state based on the Newton-Raphson method. The necessary condition of local minimum is as follows.

$$\frac{\partial E}{\partial x_m}=\frac{\partial E}{\partial y_m}=0 \quad \text{for } 1 \leqslant m \leqslant n. \tag{3}$$

The state satisfying above equation (3) corresponds to the dynamic state in which the forces of all springs are balanced. The partial derivatives of (2) by x_m and y_m are calculated. It gives $2n$ simultaneous non-linear equations of (3). But they cannot be directly solved by using a $2n$-dimensional Newton-Raphson method, because they are not independent of one another. A approach is adopted here, in which only one particle $p_m(x_m\ ,\ y_m)$ is moved to its stable point at a time, freezing the other particles. That is, viewing E as a function of only x_m and y_m , they compute a local minimum of E by using a two-dimensional Newton-Raphson method. They obtained a local minimum which satisfies equation (3) iterating this step. In each step, they choose the particle that has the largest value of Δ_m which is defined as

$$\Delta_m=\sqrt{\left\{\frac{\partial E}{\partial x_m}\right\}^2+\left\{\frac{\partial E}{\partial y_m}\right\}^2}.$$

Starting from $(x_m^{(0)},\ y_m^{(0)})$ which is equal to the current position (x_m, y_m), the following step is iterated.

$$x_m^{(t+1)}=x_m^{(t)}+\delta x,\qquad y_m^{(t+1)}=y_m^{(t)}+\delta y \qquad \text{for } t=0,1,2,\dots$$

The unknowns δx and δy will be computed in further steps . The iterations terminates when the value of Δ_m at $(x_m^{(t)},\ y_m^{(t)})$ becomes small enough.

Figure 16. illustrates the process of minimizing E in the case of a 6-vertex graph. First the particle B is moved (Fig. 16(b)) and next the particle F is moved (Fig. 16(c)). The final state (Fig. 16(d)) is obtained after 21 moving steps.

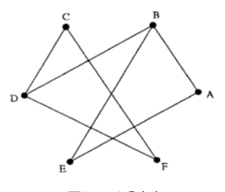

Fig. 16(a)

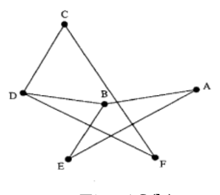

Fig. 16(b)

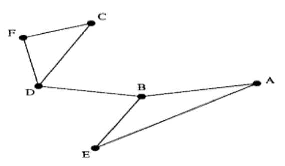

Fig. 16(c)

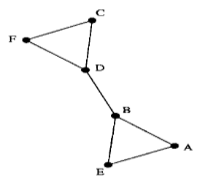

Fig. 16(d)

Figure 16. The energy minimization process. Available at [2].

The advantages of this approach are it is used for drawing general undirected graphs for human understanding. It can be widely used in the systems which deal with network structures. Here, the graph theoretic distance between vertices in a graph is related to the geometric distance between them in the drawing. This spring algorithm has many good properties like; symmetric drawings, a relatively small number of edge crossings, and almost congruent drawings of isomorphic graphs. This approach can be easily adapted and extended and have flexibility due to which can be used in 3D graph drawing, cluster graph drawing, constrained graph drawing and dynamic graph drawing. The important thing is it can be applied on weighted graphs.

The disadvantage is this algorithm does not guarantee to compute the true minimum of *E*. In order to prevent the energy from converging to a large local minimum, a simple test is to be added to the algorithm.

The algorithm of Kamada and Kawai [2] is computationally expensive, requiring an All-Pair-Shortest-Path computation which can be done in O($|V|^3$) time using the Floyd-Warshall algorithm or in O($|V|^2$ log |V |+|E||V|) using Johnson's algorithm. It has lot of iterative computations. It can be slow as the network grows.

This approach can be used to visualize small and medium sized undirected social networks, information networks, technological networks, biological networks etc. It can also be used in visualize the tree structures and 3D drawings. Most important is it can be used on weighted networks.

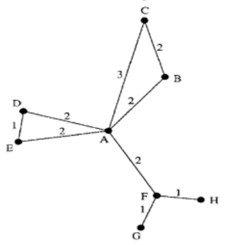

Figure 17. A weighted network. Available at [2].

I. Random Graphs Layout

The notion of a random graph was first introduced by Erdos [56]. The theory of random graphs was founded by Erdos and Renyi [56], [57]. A random graph is a graph generated by some random procedure. The two standard models for random graphs are considered here. There are many (non-equivalent) ways to define random graphs. The simplest, denoted by $G_{n,m}$ (or one of several common similar notations), where *n* and *m* are two integers with $0 \leq m \leq \binom{n}{2}$, is obtained by taking a set of *n* elements as the set of vertices, for definiteness we may take the integers 1,........, *n*, and then randomly selecting *m* (by drawing without replacement) of the $\binom{n}{2}$ possible edges.

A closely related model, denoted by for example $G_{n,p}$, where $0 \leq p \leq 1$, is obtained by taking the same vertex set but now selecting every possible edge with probability *p*, independently of all other edges (In particular, *p*=1/2 gives the uniform distribution over all (labelled) graphs on *n* vertices). The main interest is of anyone is in the case when *n*, the number of vertices, is very large, and especially in asymptotic results when $n\rightarrow\infty$ and *m* or *p* is a given function of *n*.

Example for drawing random graphs is given below in Figure. 18:

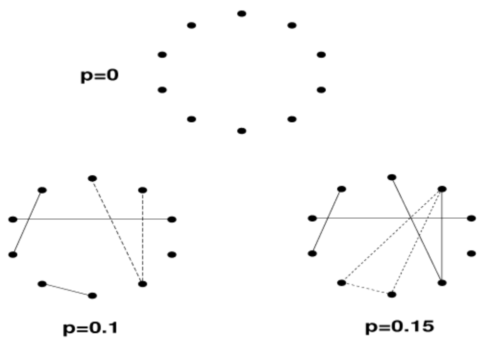

Figure 18. Evolution of the graph (p grows) $G = G_{n,p}$

The advantage of the Erdos-Renyi model [56], is the independence of choices for the edges (i.e., each pair of vertices has its own dice for determining being chosen as an edge). The computations are easy as the probability of two independent events is the product of probabilities of two events. The beauty of random graphs lies in being able to use relatively few parameters in the model to capture the behavior of almost all graphs of interest. And finally it is a very fast approach to draw graphs.

To model real graphs, there are some obvious difficulties. For example, the random graph *G(n,p)* has all degrees very close to *pn* if the graph is not so sparse, (i.e., $p\geq \log n/n$). The distribution of the degrees follows the same bell curve for every vertex. As we know, many real-world graphs satisfy the power law which is very different from the degree distribution of *G*(*n*,*p*). In order to model real-world networks, it is imperative to consider random graphs with general degree distribution and, in particular, the power law distribution. This algorithm is not recommended for automata with many high-degree vertices and for those with many vertices, as there is more potential for edge-intersection and vertex overlap respectively. Still, this algorithm can be useful by generating a radically new layout each time it is called, and has its uses for small automata. The other disadvantage of random approach is that it is not easy to interpret the graph.

The random graph approach can be used for small and somewhat medium size graphs. With some modification and addition of generating function it can be used for real world networks like companies director's network, actors network, networks from biomedicine field, collaboration networks, www linked web pages network, physically connected network of routers, email network, citation network, sexual networks etc.

Figure 19. shows a random graph and its visual complexity problem if number of nodes and edges are too many.

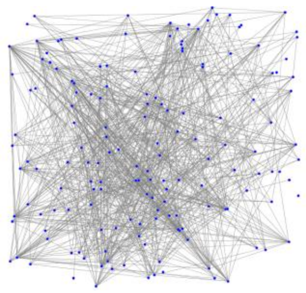

Figure 19. Showing a random graph & its visual complexity problem for graphs having large number of nodes and edges .

J. Multidimensional Scaling Layout

The goal of an MDS analysis is to find a spatial configuration of objects, when all that is known, is some measure of their general (dis)similarity. The spatial configuration should provide some insight into how the subject(s) evaluate the stimuli in terms of a (small) number of potentially unknown dimensions. Once the proximities are derived the data collection is concluded, and the MDS solution has to be determined using a computer program.

The basic approaches of MDS are Classical MDS and Nonmetric MDS. Classical MDS assumes that the data, the proximity matrix, say, display metric properties, like distances as measured from a map. Thus, the distances in a classical MDS space preserve the intervals and ratios between the proximities as good as possible. While, Nonmetric MDS assumes that the order of the proximities is meaningful. The order of the distances in a nonmetric MDS configuration reflects the order of the proximities as good as possible while interval and ratio information is of no relevance.

Classical MDS

Consider the following problem: looking at a map showing a number of cities, one is interested in the distances between them. These distances are easily obtained by measuring them using a ruler. Apart from that, a mathematical solution is available: knowing the coordinates x and y, the Euclidean distance between two cities a and b is defined by

$$d_{ab} = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}.$$

Now consider the inverse problem: having only the distances, is it possible to obtain the map?

Classical MDS, which was first introduced by Torgerson [53], addresses this problem. It assumes the distances to be Euclidean. Euclidean distances are usually the first choice for an MDS space. There exist, however, a number of non-Euclidean distance measures, which are limited to very specific research questions In many applications of MDS the data are not distances as measured from a map, but rather proximity data. When applying classical MDS to proximities it is assumed that the proximities behave like real measured distances. This might hold e. g. for data that are derived from correlation matrices, but rarely for direct dissimilarity ratings.

The advantage of classical MDS is that it provides an analytical solution, requiring no iterative procedures.

Steps of a Classical MDS algorithm

The classical MDS algorithm rests on the fact that the coordinate matrix **X** can be derived by eigenvalue decomposition from the scalar product matrix B = **XX'**.The problem of constructing **B** from the proximity matrix **P** is solved by multiplying the squared proximities with the matrix **J=I** - n^{-1} **11'**. This procedure is called double centering. The following steps summarize the algorithm of classical MDS:

1. Set up the matrix of squared proximities $P^{(2)} = [p^2]$.
2. Apply the double centering: $\mathbf{B} = -\frac{1}{2}\mathbf{J}\,\mathbf{P}^{(2)}\,\mathbf{J}$ using the matrix **J=I** - n^{-1} **11'**, where n is the number of objects.
3. Extract the m largest positive eigenvalues $\lambda_1 \ldots\ldots \lambda_m$ of B and the corresponding m eigenvectors $e_1 \ldots\ldots e_m$.
4. A m-dimensional spatial configuration of the n objects is derived from the coordinate matrix $\mathbf{X} = \mathbf{E}_m \mathbf{\Lambda}_m^{1/2}$, where E_m is the matrix of m eigenvectors and Δ_m is the diagonal matrix of m eigenvalues of B, respectively.

Take the example of cities in Denmark. Assume that we have measured the distances between Kobenhavn (cph), Arhus (aar), Odense (ode) and Aalborg (aal) on a map. Therefore, the proximity matrix (showing the distances in millimeters) might look like as Fig. 20:

	cph	aar	ode	aal
cph	0	93	82	133
aar	93	0	52	60 .
ode	82	52	0	111
aal	133	60	111	0

Figure 20. Proximity matrix

After applying the steps of algorithm following graph will be obtained as MDS solution.

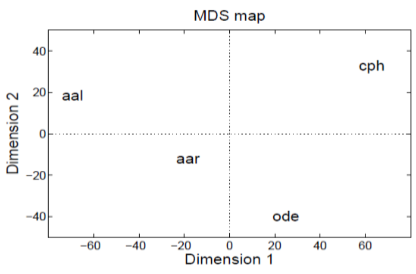

Figure 21. Shows a graphical representation of the MDS solution. Remember that this "map" is derived only from the distances between the points. Note that the dimensions cannot directly be identified with "North-South" and "East-West" without further rotation.

The advantage of classical MDS is that it provides an analytical solution, requiring no iterative procedures. It finds low-dimension projection that respects distances. Classical MDS is optimal for euclidean input data. It is still optimal, if matrix B has non-negative eigenvalues (position semi-definite) and it is very fast layout technique .

The disadvantage of classical MDS is slow, particularly for large data sets and the reasons for this will become apparent in its Computation process. Second, there is no clear guarantees for distances other than euclidean distances. It has no guarantees if matrix B has negative eigenvalues. There are two difficulties with increasing the number of dimensions. The first is that even 3 dimensions are difficult to display on paper and are significantly more difficult to comprehend.

Four or more dimensions render MDS virtually useless as a method of making complex data more accessible to the human mind.

MDS applications include scientific visualization and data mining in fields such as cognitive science, information science, psychophysics, psychometrics, marketing and ecology. In marketing, MDS is a statistical technique for taking the preferences and perceptions of respondents and representing them on a visual grid, called perceptual maps. By mapping multiple attributes and multiple brands at the same time, a greater understanding of the marketplace and of consumers' perceptions can be achieved, as compared with a basic two attribute perceptual map. MDS is becoming a popular method used in sequence clustering and visualization. In bioinformatics, MDS is used to reduce the dimensionality by giving the dissimilarity scores from each pair of sequences. These disimilarity scores are usually calculated using Sequence Alignment. By mapping each sequence from the high dimensional space to a visually acceptable space (such as 2D/3D space), the correlations between each sequence cluster can be observed easily.

Nonmetric MDS

The assumption that proximities behave like distances might be too restrictive, when it comes to employing MDS for exploring the perceptual space of human subjects. In order to overcome this problem, Shepard [37],[38] and Kruskal [23] developed a method known as *nonmetric multidimensional scaling.*

In nonmetric MDS, only the ordinal information in the proximities is used for constructing the spatial configuration. A monotonic transformation of the proximities is calculated, which yields scaled proximities. Optimally scaled proximities are sometimes referred to as *disparities* $\hat{d}$ = **f(p)**. The problem of nonmetric MDS is how to find a configuration of points that minimizes the squared differences between the optimally scaled proximities and the distances between the points. More formally, let **p** denote the vector of proximities (i. e. the upper or lower triangle of the proximity matrix), **f(p)** a monotonic transformation of **p**, and **d** the point distances; then coordinates have to be found, that minimize the so-called *stress*

$$\text{STRESS} = \sqrt{\frac{\sum (f(p) - d)^2}{\sum d^2}}.$$

MDS programs automatically minimize stress in order to obtain the MDS solution; there exist, however, many (slightly) different versions of stress.

Judging the goodness of fit

The amount of stress may also be used for judging the goodness of fit of an MDS solution: a small stress value indicates a good fitting solution, whereas a high value indicates a bad fit. Stress decreases as the number of dimensions increases. Thus, a two-dimensional solution always has more stress than a three-dimensional one. Since the absolute amount of stress gives only a vague indication of the goodness of fit, there are two additional techniques commonly used for judging the adequacy of an MDS solution: the Scree plot and the Shepard diagram

In a scree plot, the amount of stress is plotted against the number of dimensions. Since stress decreases monotonically with increasing dimensionality, one is looking for the lowest number of dimensions with acceptable stress. An "elbow" in the scree plot indicates, that more dimensions would yield only a minor improvement in terms of stress. Thus, the best fitting MDS model has as many dimensions as the number of dimensions at the elbow in the scree plot.

The Shepard diagram displays the relationship between the proximities and the distances of the point configuration. Less spread in this diagram implies a good fit. In nonmetric MDS, the ideal location for the points in a Shepard diagram is a monotonically increasing line describing the so-called disparities, the optimally scaled proximities. In an MDS solution that fits well the points in the scree plot are close to this monotonically increasing line.

Figure 22. Shows a paradigmatic scree plot and a Shepard diagram. The elbow in the scree plot suggests a three-dimensional MDS space, while the little amount of spread in the Shepard diagram indicates a rather good fit of the solution.

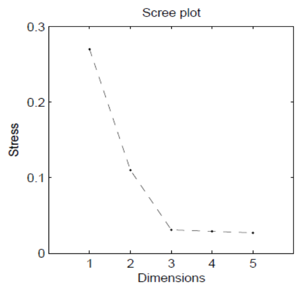

Fig. 22(a)

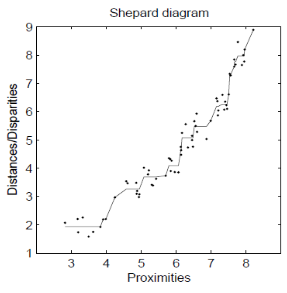

Fig. 22(b)

Figure 22. Fig. 22(a) Shows a Scree plot displaying an elbow at three dimensions.
Fig. 22(b) Shows a Shepard diagram with the optimally scaled proximities.

Steps of nonmetric MDS algorithm

The core of a nonmetric MDS algorithm is a twofold optimization process. First the optimal monotonic transformation of the proximities has to be found. Secondly, the points of a configuration have to be optimally arranged, so that their distances match the scaled proximities as closely as possible. The basic steps in a nonmetric MDS algorithm are:

1. Find a random configuration of points, e. g. by sampling from a normal distribution.
2. Calculate the distances **d** between the points.
3. Find the optimal monotonic transformation of the proximities, in order to obtain optimally scaled data **f(p).**
4. Minimize the stress between the optimally scaled data and the distances by finding a new configuration of points.
5. Compare the stress to some criterion. If the stress is small enough then exit the algorithm else return to step 2.

The advantage of nonmeteric MDS are, it fulfills a clear objective without many assumptions that is minimizing stress. Its results don't change with rescaling or monotonic variable transformation. It works even if you only have rank information.

The disadvantages of nonmetric MDS are,first, MDS is slow, particularly for large data sets and the reasons for this will become apparent in its iterative computational steps. Second, because MDS is a numerical optimization technique, it can fail to find the true best solution because it can become stuck on local minima, solutions that are not the best solution but that are better than all nearby solutions. Third, normally, MDS is used to provide a visual representation of a complex set of relationships that can be scanned at a glance. Since maps on paper are two dimensional objects, this translates technically to finding an optimal configuration of points in 2-dimensional space. However, the best possible configuration in two dimensions may be a very poor, highly distorted, representation of your data. If so, this will be reflected in a high stress value. Fourth, there are two difficulties with increasing the number of dimensions. The first is that even 3 dimensions are difficult to display on paper and are significantly more difficult to comprehend. Four or more dimensions render MDS virtually useless as a method of making complex data more accessible to the human mind. The second problem is that with increasing dimensions, you must estimate an increasing number of parameters to obtain a decreasing improvement in stress. The result is model of the data that is nearly as complex as the data itself. Fifth, in nonmetric MDS usually only local (not global) optimum found.

Non-metric MDS has been used extensively in the psychometrics and psychophysics communities to embed similarity and dissimilarity ratings derived from a variety of sources. MDS has been used extensively in geostatistics, for modeling the spatial variability of the patterns of an image and natural language processing, for modeling the semantic and affective relatedness of natural language concepts.MDS is also used in marketing, bioinformatics and ecology fields. MDS can be used for mapping computer usage data, mathematical graphs, social network graphs and reconstruction of molecules in nano-technology.

V. Conclusion And Future Scope

Graph Drawing is concerned with the geometric representation of graphs and networks and is motivated by those applications where it is crucial to visualize structural information as graphs. Since graph drawing methods form the algorithmic core of network visualization, that is why bridging the gap between theoretical advances and implemented solutions is an important aspect. This paper presents an overview of many possible types of graph visualization approaches and their implementation details. Its primal goal is, to help a user, with some relational data on his hands and a need to visualizing it, with the choice of what to use, and what is out there that can be used. However, it can be also viewed as historical overview of graph drawing algorithms, and its evolution and will be quite helpful in the future development of related algorithms and technologies.

The field of graph drawing and visualization has a broad scope like development of tools and systems for graph drawing, development of user interfaces for viewing graphs, interactive exploration of large graphs, presentation of dynamic graphs and animation of graphs, applications of graph drawing to areas such as software visualization, user interface design and database query formulation. The field of graph drawing has mainly focused on the structure of graphs, whereas practitioners of information visualization are more concerned with embedding information, often multivariate, into the nodes and the links. In future we would like to work on how can we merge the best practices of both fields?

Multivariate networks are large and complex and their complexity will increase in the future. Thus, not all problems can be solved in the short term. There already exist a number of technical approaches, algorithms, and methods to interactively visualize multivariate networks. In future we would also like to work on which (approaches) ones are suitable for solving specific tasks in our applications areas, what is their potential, and what are their limitations. By identifying the range of approaches that do exist, we see the potential for new, innovative visualization ideas.

References

[1] Colin Ware, Helen Purchase, Linda Colpoys, Matthew McGill, "Cognitive Measurements of Graph Aesthetics," *Information Visualization*, 1, Issue 2, 2002, pp. 103-110.

[2] Tomihisa Kamada, Satoru Kawai, "An Algorithm for Drawing General Undirected Graphs," *Information Processing Letters*, 31(1), 1989, pp. 7-15.

[3] E. Adar, "GUESS: A Language and Interface for Graph Exploration," Conference on Human Factors in Computing Systems. http://www.graphexploration.org

[4] V.Batagelj, A. Mrvar, "Pajek – Program for Large Analysis," *Connections*, 21, 1998, pp. 47-47.

[5] E.R. Gansner, S.C. North, "An open visualization system and its applications to software engineering," *Software – Practice and Experience*, 30(11), 2000, pp. 1203-1233.

[6] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis, "Graph Drawing: Algorithms for the Visualization of Graphs," Prentice Hall, Upper Saddle River, New Jersey, 1999.

[7] G. Di Battista, P. Eades, R. Tamassia and I. Tollis, "Algorithms for Drawing Graphs: An Annotated Bibliography," *Computational Geometry: Theory and Applications*, 4(5), 1994, pp. 235-282.

[8] G. Di Battista, A. Garg, G. Liotta, R. Tamassia, E. Tassinari, F. Vargiu and L. Vismara, "An Experimental Comparison of Four Graph Drawing Algorithms," *Computational Geometry: Theory and Applications*, 7(5-6), 1997, pp.303-26.

[9] R.F. Cohen, G. Di Battista, R. Tamassia, I. G. Tollis, P. Bertolazzi, "A framework for dynamic graph drawing," In *Proceedings of the 8th Annual Symposium on Computational Geometry (SCG '92)*, 1992, pp. 261–270.

[10] Peter Eades, "Drawing free trees," Bulletin of the Institute of Combinatorics and its Applications, 5, 1992, pp. 10-36.

[11] P. Eades, "A heuristic for graph drawing," *Congressus Numerantium*, 42, 1984, pp. 149–160.

[12] Guy Melancon , Ivan Herman, "Circular drawings of rooted trees," Technical report, Amsterdam, The Netherlands, 1998.

[13] Soon Tee Teoh, Kwan-Liu Ma, "Rings: A technique for visualizing large hierarchies," In GD '02: Revised Papers from the 10th International Symposium on Graph Drawing, London, UK, Springer-Verlag. ISBN 3-540-00158-1, 2002, pp. 268-275.

[14] Ka-Ping Yee, Danyel Fisher, Rachna Dhamija, and Marti A. Hearst. "Animated exploration of dynamic graphs with radial layout," In *Proceedings of the IEEE Symposium on Information Visualization*, 2001, pp. 43-50.

[15] U. Dogrusoz, B. Madden, P. Madden, "Circular layout in the graph layout toolkit," In Proc. GD '96, LNCS 1190, 1997, pp. 92–100,.

[16] M. Kaufmann, R. Wiese, "Maintaining the Mental Map for Circular Drawings," In Proc. GD 2002, LNCS 2528, 2002, pp. 12–22.

[17] E. Makinen, "On Circular Layouts," In Intl. Jrnl of Computer Mathematics, 24,1988, pp. 29–37.

[18] F. P. Preparata, M. I. Shamos, "Computational Geometry: An Intro-duction," Springer-Verlag, New York, 1985.

[19] J. M. Six, I. G. Tollis, "A framework and algorithms for circular drawings of graphs," Jrnl. of Discrete Algorithms, 4(1), 2006, pp. 25-50.

[20] Ron Davidson, David Harel, "Drawing graphs nicely using simulated annealing," *ACM Transactions on Graphics*, 15(4), 1996, pp. 301–331.

[21] T. M. Fruchterman, E. M. Reingold, "Graph drawing by force-directed placement," Software-Practice and Experience, 21(11), 1991, pp. 1129- 1164.
[22] Benjamin Finkel, Roberto Tamassia, "Curvilinear Graph Drawing Using the Force-Directed Method," In *Proc. 12th Int. Symp. on Graph Drawing (GD 2004)*, 2005, pp. 448–453.
[23] J. B. Kruskal, "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis," *Psychometrika*, 29, 1964, pp. 1–27.
[24] William T. Tutte, " How to draw a graph," *Proc. London Math. Society*, 13(52), 1963, pp. 743–768.
[25] T. Kamada, S. Kawai,"'Automatic display of network structures for human understanding," *Technical Report 88-007*, Department of Information Science, Tokyo University, February, 1988.
[26] A. Frick, A. Ludwig, H. Mehldau, "A fast adaptive layout algorithm for undirected graphs," In R. Tamassia and I. G. Tollis, editors, Graph Drawing (Proc. GD '94), volume 894 of Lecture Notes in Computer Science, Springer-Verlag, 1995, pp. 388--403.
[27] W. S. Cleveland, R. McGill, "Graphical perception: Theory, experimentation, and application to the development of graphical methods," Journal of the American Statistical Association, 79(387) 1984, pp. 531-554.
[28] S.G. Eick, " Aspects of network visualization," IEEE Computer Graphics and Applications, 16(2), 1996, pp. 69-72.
[29] E.R. Gansner, S.C. North, "Improved force-directed layouts," *Proceedings of Graph Drawing '98*, LNCS 1547, Springer-Verlag, 1998, pp. 364-373.
[30] J.B. Kruskal, M. Wish, "Multidimensional Scaling," volume 07-011 of Sage University Paper series on Quantitative Applications in the Social Sciences. Sage Publications, 1978.
[31] X. Mendonca, P. Eades, "Learning aesthetics for visualization," In Anais do XX Seminario Integrado de Software e Hardware, Florianopolis, Brazil, 1993, pp. 76-88.
[32] H.C. Purchase, R.F. Cohen, M. James,"An experimental study of the basis for graph drawing algorithms," ACM Journal of Experimental Algorithmics, 2(4), 1997.
[33] W.T. Tutte, "How to draw a graph," Proceedings of the London Mathematical Society, Third Series, 13, 1963, pp. 743-768.
[34] I. Borg, P. Groenen, " Modern Multidimensional Scaling: Theory and Applications," 2nd edition, Springer Verlag, 2005.
[35] T.F. Cox, M.A. Cox, "Multidimensional Scaling," Chapman & Hall/ CRC, 2000.
[36] J. B. Kruskal, "Nonmetric multidimensional scaling: A numerical method," Psychometrika, 29, 1964, pp. 115–129.
[37] R.N. Shepard, " The analysis of proximities: Multidimensional scaling with an unknown distance function. I," Psychometrika, 27(2), 1962, pp. 125–140.
[38] R.N. Shepard, "The analysis of proximities: Multidimensional scaling with an unknown distance function. II, " Psychometrika, 27, 1962, pp. 219–246.
[39] F.J. Brandenburg, M. Himsolt, C. Rohrer, "An Experimental Comparison of Force-Directed and Randomized Graph Drawing Algorithms, " *Proceedings of Graph Drawing '95*, LNCS 1027, Springer Verlag, 1995, pp. 76–87.
[40] R. Hadany, D. Harel, "A Multi-Scale Method for Drawing Graphs Nicely, " Discrete Applied Mathematics **113,** 2001, pp. 3-21.
[41] D. Harel, Y. Koren, "A Fast Multi-Scale Method for Drawing Large Graphs," Journal of Graph Algorithms and Applications **6**, 200), pp. 179–202.
[42] D. Harel and Y. Koren, "Graph Drawing by High-Dimensional Embedding," Proceedings of Graph Drawing 2002, LNCS 2528, Springer-Verlag, 2002, pp. 207–219.
[43] M. Kaufmann, D. Wagner (Eds.), "Drawing Graphs: Methods and Models," LNCS 2025, Springer-Verlag, 2001.
[44] C. Walshaw, "A Multilevel Algorithm for Force-Directed Graph Drawing," Proceedings 8th Graph Drawing (GD'00), LNCS 1984, Springer-Verlag, 2000, pp. 171–182.
[45] J.D. Carroll, P. Arabie, "Multidimensional scaling," M. R. Rosenzweig and L.W. Porter, eds. *Annual Review of Psychology*, 31, 1980, pp. 607–649.
[46] M.L. Davison, "Multidimensional scaling." New York, John Wiley & Sons, 1983
[47] P.E. Green, F.J. Carmone, "Multidimensional scaling and related techniques," Boston, Allyn and Bacon, 1970,
[48] P.E. Green, V.R. Rao, "Applied multidimensional scaling," New York, Holt, Rinehart and Winston, 1972.
[49] J.C. Lingoes, E.E. Roskam, "A mathematical and empirical study of two multidimensional scaling algorithms," *Psychometrika Monograph Supplement*, 19, 1973.
[50] S.S. Schiffman, M.L. Reynolds, F.W. Young, "Introduction to multidimensional scaling: Theory, methods, and applications," New York: Academic Press, 1981
[51] R.N. Shepard, A.K. Romney, S. Nerlove, "Multidimensional scaling: Theory and application in the behavioral sciences," New York, Academic Press, 1972.
[52] W.S. Torgerson, "Theory and methods of scaling," New York, John Wiley & Sons, 1958.
[53] W.S. Torgerson, "Multidimensional scaling: I- Theory and method. Psychometrika, 17,1952, pp. 401-419.
[54] E. R. Gansner, Y. Koren, S. North, "Graph drawing by stress majorization," In Proceedings of the 11th International Symposium in Graph Drawing (GD'03), volume 2912 , Springer LNCS, 2004, pp. 239-250.
[55] B. Bollobas, "Random Graphs," Academic Press, New York, 1985.
[56] P. Erdos, A. Renyi, "On random graphs," Publicationes Mathematicae 6, 1959, pp. 290-297.
[57] P. Erdos, A. Renyi, "On the evolution of random graphs," Publications of the Mathematical Institute of the Hungarian Academy of Sciences 5, 1960, pp. 17-61.
[58] S. Janson, T. Luczak, A. Rucinski, "Random Graphs," John Wiley, New York, 1999.
[59] Ulrik Brandes, "Drawing on Physical Analogies-Drawing Graphs," LNCS 2025, 2001, pp. 71-86.
[60] J.S. Tilford, "Tree Drawing Algorithms, " Technical Report UIUCDCS-R-81-1055, Department of Computer Science, University of Illinois at Urbana-Champaign, IL, 1981.

Himanshu Sharma recieved his B.E degree in computer science from University of Rajasthan in year 2006. He is pursuing M.Tech presently. His main research interests includes data mining, network analysis and knowledge engineering.

Vishal Srivastava recieved his B.Tech & M.Tech in degree computer science from RGPV college in 2004 and 2007 respectively and pursuing his PhD presently. He is working as professor in Arya College of Engineering and Information Technology, Jaipur, Rajasthan. He is member of CSI, IAE, UACEE, SCIEI, IJCTT, IJATER. He is also editorial board member and reviewer in IJACR, IJRET, IJCST, IJCT, IJAERD, IJATEE etc in area of CS and IT. His area of interest includes Networking and Data Mining. He has published 77 International papers and 24 National papers in international journals, conferences and national conferences. He also get grants from DST, Government of Rajasthan for his R&D and projects like SAMADHAN, Vaccination schedule alert system on mobile, Petition signing an android application.