International Journal of Computer Applications (0975 – 8887)
Volume 120 – No.21, June 2015

A Force Directed Layout Algorithm for Biological Networks

Pritish Dubey
M.Tech Student
Vishwakarma Institute of Technology, Pune

Ashwini Shingare
Assistant Professor
Vishwakarma Institute of Technology, Pune

Vrushali Inamdar
Team Lead
Persistent Labs, Pune

ABSTRACT

In this paper, we present a layout algorithm for clustered graphs which is a modified force directed algorithm. We have used Spring Embedder algorithm by Eades as base for our algorithm and modified it to suit the constraints of general biological graphs. Our main contribution is adopting spring embedder algorithm to maintain clustered structure of original graph with inherent depth of nesting and handling different node sizes. Results show that our layout algorithm draws graphs with acceptable quality with respect to aesthetic criteria for graph drawing. The algorithm has been integrated with systems biology visualization suit called "eSkin", which allows lay outing and analyze biological graphs.

General Terms

Graph Visualization, Layout Algorithm.

Keywords

Clustered Graph, Biological Pathways, Force Directed Layout

1. INTRODUCTION

A graph is a collection of points and lines connecting some (possibly empty) subset of them. The points of a graph are called graph vertices or nodes and the lines connecting the vertices of a graph are known as graph edges or lines [1]. For a long time graphs have been used to visualize complex data.

A drawing of graph is the representation of nodes and edges on a surface. A graph can have many various drawings as the position of nodes on the surface does not matter. The process of drawing graphs using some algorithm is called automatic graph drawing. These algorithms are called as lay outing algorithms. Over the years, a set of assumed 'graph drawing aesthetics' has emerged, defining the criteria by which the 'goodness' of the graph drawing produced by a layout algorithm can be assessed [2]. Such aesthetics include, for example, a minimum number of edge crossings, as few edge bends as possible, a display of symmetric sub-structures, and large angles between edges incident at a node. Graph layout algorithms thus are rated to the extent they conform to these criteria. Graphs drawn from such algorithms can alternately be valued upon the level of human cognition or comprehension of underlying data being represented [3].

1.1 Biological Networks

Networks have been used for a long time to represent important biological processes. Networks are now routinely used to show relationships between biologically relevant molecules. Analysis of these networks is important for understanding such relationships and formulate hypotheses about biological properties [4]. A biological pathway is a series of actions among molecules in a cell that leads to a certain product or a change in the cell [5].

These pathways are represented as graphs with complex structures inside them representing complex molecules. These molecules are modelled as clusters in the graph. In addition there are constraints that child cluster or molecules of one cluster should not move outside the parent cluster, as well as should not enter other clusters. These constraints make drawing of biological pathways with traditional lay out algorithms not possible. In order to layout such graphs we have to make appropriate changes and consider various constraints for layout [6].

CONVENTIONS

Formally, a graph is a pair of sets (V, E), where V is the set of vertices and E is the set of edges, formed by pairs of vertices. A cluster C is set of vertices and may or may not be the part of other clusters. A clustered graph G = (V, E, T) consist of E set of edges, V set of vertices and a rooted tree T such that children of T can be vertices of G or other clusters..

A cluster C is part of the graph G which contains other clusters or nodes. Cluster is interchangeably called a 'sub graph' from here onwards. An "inter graph edge" is an edge which is connecting two vertices that are part of different sub graphs.

2. PROPOSED FORCE MODEL

We propose a physical model which is the extension of traditional physical model proposed by Eades in his algorithm of Spring Embedder [7]. This physical model is modelled to adhere to constrains imposed by biological networks which are essentially clustered graphs in nature. Traditional force directed algorithms assign charge to each nodes in a graph such that every nodes repel every other node and adjacent nodes attract each other. The following modifications were made to the traditional force model.

a) Nodes repel other nodes only if they are part of same cluster or sub graph.
b) Nodes attract other adjacent nodes only if they are part of same cluster or sub graph.
c) Edges going from one cluster to other clusters are split into smaller parts or springs from the point where it intersects the clusters. These parts then acts as individual springs for nodes they are connected to.
d) Sub graphs or clusters repel other nodes present in the same cluster or sub graph.
e) Sub graphs or clusters repel other clusters present in the same cluster or sub graph.
f) Point of repulsion between two clusters and vertices of different size is calculated by joining the centers of two clusters and finding clipping points with the cluster.

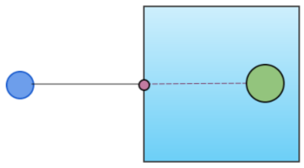

Figure 1: An edge between node of different sub graphs and its clipping point with sub graph.

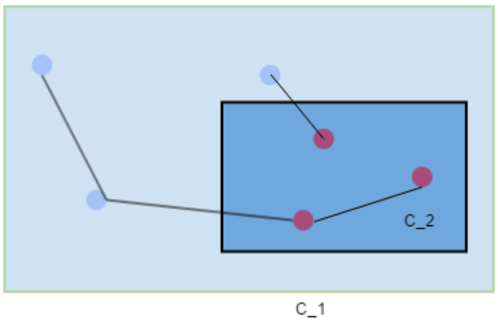

Figure 2: Structure of clustered graph

Fig. 1 shows an inter graph edge that connects two nodes of different sub graphs or clusters. Figure also shows the clipping point of edge with the boundary of graph. This point is used to divide the spring into two parts, each one attracting a node towards the clipping point. We calculate clipping point using Cohan and Sutherland algorithm [8].

Fig. 2 shows the structure of a typical clustered graph. Graph has two clusters with cluster C_1 containing three nodes and a cluster C_2. According to our force model all the nodes belonging to C_1 only (marked by blue color) will repel each other. Only immediate nodes of C_1 which are adjacent to each other will be subjected to spring forces. Immediate nodes of cluster C_1 will be repelled by the cluster boundaries of C_2. Inter graph edges between nodes of C_1 and C_2 are split up at points where they intersect with cluster boundary of C_1 and are called as clipping points. Edge is broken up into two springs at clipping points and attracts the nodes towards the clipping point.

3. PROPOSED LAY OUT ALGORITHM

The algorithm comprises of following seven phases for each level of nesting depth in the graph. These five phases are repeated until convergence is achieved or some default numbers of iterations are done.

a) **Relax Edges:** Edges which are part of present sub graph or nesting depth are relaxed.
b) **Relax Inter graph edge**s**:** All the graph edges which are connecting nodes of different graphs are relaxed.
c) **Repel Nodes:** Repel nodes which are part of current nesting depth.
d) **Repel Clusters:** Repulsion forces are applied to clusters and nodes at each nesting depth.
e) **Move Clusters and Nodes:** Move the clusters and nodes at a nesting depth depending upon their displacement.

Every node and cluster is associated with a 'displacement' value which denotes the amount by which that node or cluster will be displaced. It is assumed that each sub graph has a unique Id and it is same as that of immediate nodes of that sub graph. It helps us check whether a node is the immediate children of a given sub graph.

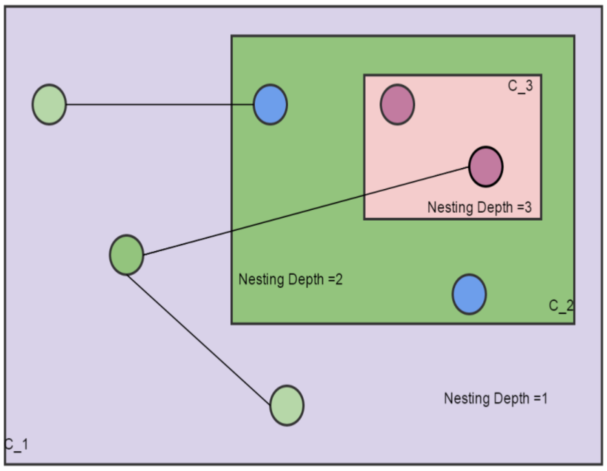

Figure 3: Flow of Algorithm

We start applying the algorithm from the lowest nesting depth applies layout and then moves higher nesting.

```
Clustered Graph Layout
Input: Main graph G
For each nesting level 'L' in the graph
    For each sub graph G' in current level 'L'
        Relax Edges (G')
        Relax Inter graph edges (G')
        Repel Nodes (G')
        Repel Clusters (G')
        Move Clusters and Nodes (G')
    End
End
```

The time complexity of our algorithm is found to be $NS\left(O(M_i^2) + O(E_i) + O(E_{inter\ graph})\right)$ where 'N' is number of iterations; 'S' is the number of total sub graphs. 'M_i' is the total number of nodes and cluster at a nesting depth 'i', 'E_i' is the immediate edges of a sub graph at a nesting depth 'i'. $(E_{inter\ graph})$ is the total number of inter graph edges in the graph.

As shown in figure 3, our algorithm will start layout with nesting depth one, perform lay outing with three nodes and a cluster, after that we move the inner nesting level (two) and apply the above steps will be applied until all nesting levels are applied.

3.1 Applying Spring Forces

We apply standard spring forces suggested by Eades which were modelled on Hook's Law. Force acting on the vertices of an edge will become zero, when reaches a certain "desired length". Constant α is the attraction multiplier, which is similar to elasticity constant of springs and is used to determine the rate at which edges contract or expand.

Algorithm 1: Relax Edges

```
Input: Sub graph G
For each immediate edge ‘E’ of G connecting any two U and V
  $Distance\ D = LineSegment(U,V)$
  $Fs = (\alpha * (desiredEdgeLength - D))/D$
  $Fs = Fs * pow(degreeFactor, Degree(u+v))$
  $Displacement(U) = Fs$
  $Displacement\ (V) = -Fs$
  End
End
```

Relaxing inter graph edges are similar to relaxing edges of the same graph except that we need to calculate clipping points of that edge with sub graph. Depending upon the level of nesting of that inter graph edge the calculation of clipping points may vary and can be costly.

Algorithm 2: Relax Inter Graph Edges

```
Input: Main graph G’
For each inter graph edge ‘E’ in graph G’ connecting any two U and V
  Get Clipping points Cu and Cv for U and V
  Distance Du = LineSegment(Cu,V)
  Distance Dv = LineSegment(Cu,V)
  $Fs = (\alpha * (desiredEdgeLength - Du))/Du$
  $Fs = Fs * pow(degreeFactor, Degree(u+v))$
  $Ft = (\alpha * (desiredEdgeLength - Du))/Du$
  $Ft = Ft * pow(degreeFactor, Degree(u+v))$
  $Displacement(U) = Fs$
  $Displacement\ (V) = Ft$
  End
End
```

Here α is the inter edge attraction multiplier. Both the above algorithms require $o(E)$ time, where is ‘E’ number of edges in the graph.

3.2 Applying Repulsive Forces

We apply repulsive forces among nodes and clusters by moving a node across the centers of all the other nodes and clusters and adding to its displacement by some constant distance. We calculate clipping points between nodes and clusters for a line passing through their centers to accommodate the difference of their sizes.

Algorithm 3: Repel Nodes

```
Input: Sub graph G
  For each immediate node U of sub graph do
    For each immediate node V of sub graph do
      Distance D = LineSegment(U,V)∩ Width(U,V)
      Dx = U - V
      Fr = $\varphi$ * ( Dx/ D)
      Dispacement(U) += Fr/$||D||$
    End
  End
End
```

In the above algorithm we traverse all the immediate children nodes to apply repulsive force between them. φ is the repulsion constant and affects the amount by which nodes repel each other. We have taken default value 100. This step is costliest and requires $o(n^2)$ time. Similarly here also we have taken width of nodes to accommodate different vertex sizes.

Algorithm 4: Repel Clusters

```
Input: Sub graph G
For each Cluster “C” in current graph G
  For each Node “n” in current nesting depth
      Get Clipping points Cu between n and C
      Distance D = LineSegment(Cu,n)
      MovementVector = Cu – Position(n)
      Fr = $\omega$ * MovementVector/D
      Dispacement(n) = Fr/$||D||$
      Dispacement(U) += Fr/$||D||$
      Limit Displacement(U) by two units only
  End
  For each Cluster “J” in graph G
      Get Clipping points Cu and Cv between C and J
      Distance D = LineSegment(Cu,Cv)
      MovementVector = Cu – Cv
      Fr = $\omega$ * MovementVector/D
      Dispacement(J) += Fr/$||D||$
      Dispacement(U) += Fr/$||D||$
      Limit Displacement(C,J) by 15 units only
  End
End
```

While applying repulsive forces between clusters and nodes we traverse through the cluster at current nesting depth. For each cluster we repel nodes and clusters at that level and to their displacements the movement vectors. ω is the repulsion constant and affects the amount by which nodes repel each other. This step requires $o(nC^2)$ where C is the number of clusters and *n* is the number of nodes at current nesting depth.

3.3 Moving Nodes and Clusters

We move the nodes and cluster a nudge. We try to avoid huge changes to the positions of clusters so as to maintain user's mental map [9]. We also limit the movement of nodes if their displacement value is too high, similar to that of 'Simulated Anneling' method [10].

Algorithm 5: Move Cluster and Nodes

```
Input: Sub graph G
For each cluster 'C' in current graph G
    Limit Displacement(C) by some factor
    Move cluster by Displacement(C)
    End
For each node 'n' in current graph G
    Limit Displacement(n) by some factor
    Move node by Displacement(n)
    End
End
```

4. RESULTS AND ANALYSIS

We implemented the algorithm using C++ and Boost Graph Library and are currently available as a part of "eSkin" visualization suite. The results we got were satisfactory when considering general aesthetic criteria like edge crossings, bounding area of graph [11] and drawing time of the algorithm. We tested and validated our algorithm with standard systems biology pathways. Following are some of sample graphs drawn with our algorithm.

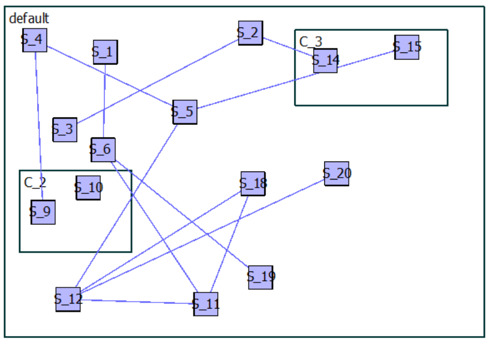

Figure 4: a) Clustered graph before applying Layout

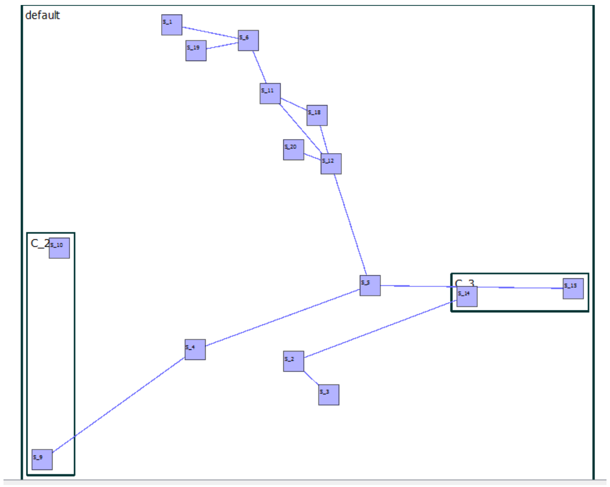

Figure 4 b) Graph after applying lay outing algorithm

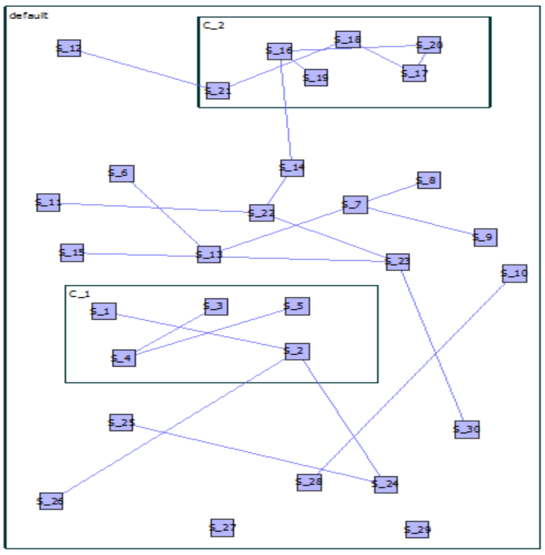

Figure 5 a) Clustered graph before applying Layout

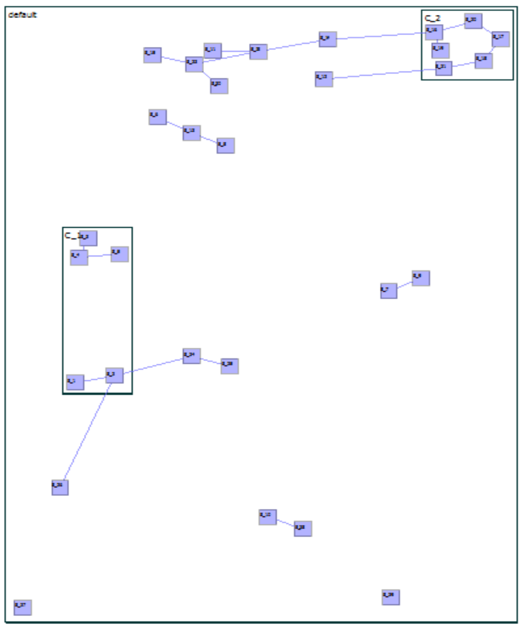

Figure 5 b) Graph after applying lay outing algorithm

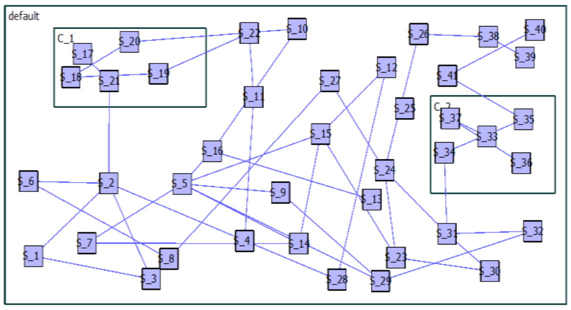

Figure 6 a) Clustered graph before applying Layout

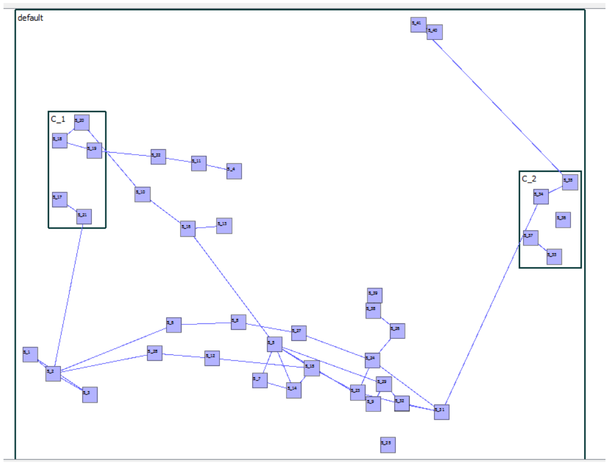

Figure 6 b) Clustered graphs after applying Layout

The above figure of graphs laid out with our algorithms confirms the following points:

a) Inherent structure of clustered graph is maintained and all the nodes remain in their respective clusters after layout.
b) Algorithm exposes the underlying symmetry of clustered graphs.
c) Edge crossings are reduced and in some cases eliminated completely. However just like other layout algorithms there is no guarantee.
d) Graphs turn out to be more aesthetically pleasing.
e) No violent changes are made to location of clusters to maintain user's mental map.

The algorithm was tested with the standard biological pathways for skin cells [12]. The pathway graphs were initially laid out randomly on a plane. On these initial random layouts, our modified Force Directed Layout algorithm was applied. Layout performance can be seen in Figure 7. The figure shows the number of edge crossings in 35 standard pathway graphs before and after applying the algorithm. We observed that the edge crossings were reduced by 75% on an average after applying the algorithm. Also the clusters in the graph were preserved.

It is profoundly hard to consummately eliminate node overlaps for non-uniform node dimensions that are engendered by a spring embedder [13], this is due to the reason that the constants associated with opposing magnetization and repulsion forces are arduous to fine-tune [14],[15]. The overlap amounts are virtually always inconsequential.

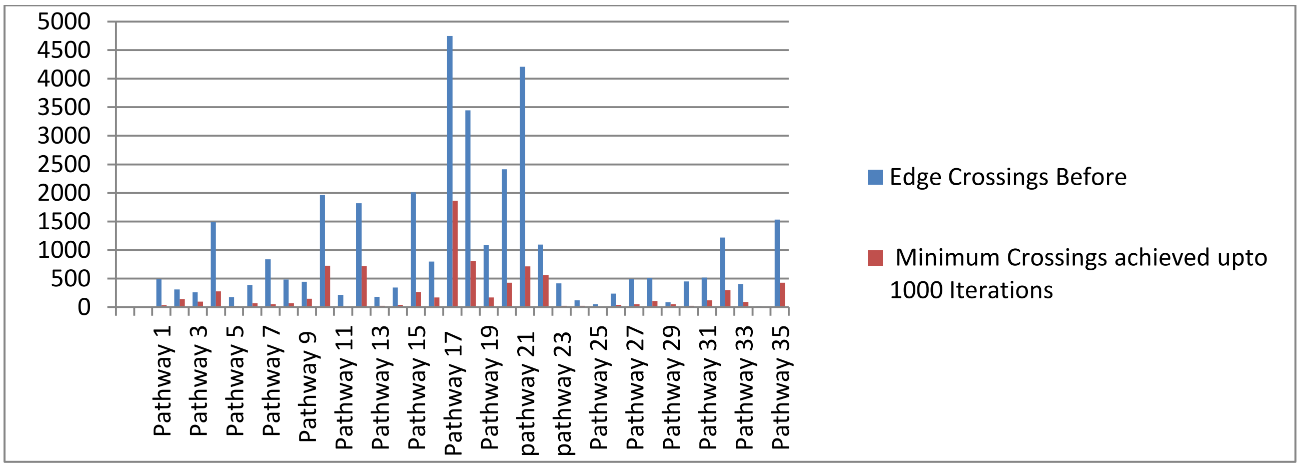

Figure 7: Edge Crossings of biological pathways before and after lay outing.

5. CONCLUSION

We present a novel algorithm for layout of complex biological networks which are inherently clustered in nature. We modelled spring embedder algorithm to handle additional constraints of clustered graphs and irregular node sizes. We considered variable node sizes and clusters of different sizes in our physical model. Our algorithm does not impose any limitation of nesting depth of the graph and produces satisfactory results for general metrics including computational efficacy. The results show 75% reduction in edge crossing number in the resulting layout.

As a future work, we will try to reduce the computational time of the algorithm. The main drawback of this algorithm is its large running time. As discussed in section 3, the running time depends on number of clusters, number of nodes at each level of clusters and number of iterations. As we are using recursive approach to layout a cluster, we can make use of parallel computing approaches. By this, we will be able to layout disjoint clusters in parallel. This is expected to reduce the computational time.

6. REFERENCES

[1] Eric Weisstein, "Graph Theory", May 13 2015 Available: http://mathworld.wolfram.com/Graph.html

[2] Coleman and Stott Parker, 12 Dec. 1996, "Aesthetics-Based Graph Layout for Human Consumption",

Journal of Software Practice & Experience, Volume 26, p 1415-1438

[3] Helen C. Purchase, Beryl Plimmer, Baker, Pilcher, 2010, “Graph Drawing Aesthetics in User-Sketched Graph Layouts”, AUIC '10 Proceedings of the Eleventh Australasian Conference on User Interface, Volume 106, p 80-88.

[4] John Morris, Allan Kuchinsky, 2014, “Analysis and Visualization of Biological Networks with Cytoscape”.

[5] “Biological Pathways”,National Human Genome Research Institute. April 5 2015,[Online] Available https://www.genome.gov/27530687 .

[6] Kathy Ryall, Joe Marks and Stuart Shieber, 1997, “An interactive constraint-based system for drawing graphs”, Proceedings of the 10th annual ACM symposium on User interface software and technology, p 97-104.

[7] Peter Eades, 1984, “A heuristic for graph drawing”. Congressus Numerantium, 42:149–160.

[8] P. Asokarathinam, Cohen – Sutherland Line Clipping Algorithm, November 27, 1996. [Online] http://www.cs.helsinki.fi/group/goa/viewing/leikkaus/lineClip.html

[9] Yi-Yi Lee, Chun-Cheng Lin, Hsu-Chun Yen, 2006 “Mental Map Preserving Graph Drawing Using Simulated Annealing” at Asia Pacific Symposium on Information Visualization (APVIS 2006).

[10] T. Fruchterman and E. Reingold, 1991, “Graph drawing by force-directed placement”, Software Pract. Exp., 21(11), p 1129–1164.

[11] H. C. Purchase, 2002, “Metrics for Graph Drawing Aesthetics”, at Journal of Visual Languages and Computing.

[12] KEGG Pathway Database April 27, 2015. [Online] http://www.genome.jp/kegg/pathway.html

[13] Josep Díaz, Jordi Petit and Maria Serna, September 2002, “A survey of graph layout problems”, Journal of ACM Computing Surveys (CSUR) Surveys Homepage archive Volume 34 Issue 3, Pages 313-356.

[14] Yifan Hu, 2005, “Efficient and high quality force-directed graph drawing”. The Mathematica Journal, 10:37–71.

[15] J. B. Kruskal, 1964, “Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis”, Psychometrika, 29:1–27.