# Near-Optimal Concentric Circles Layout

Prabhakar V. Vemavarapu, Mehmet Engin Tozal, and Christoph W. Borst

School of Computing and Informatics
University of Louisiana at Lafayette, Lafayette LA 70504, USA
{C00255627, metozal, cxb9999}@louisiana.edu

**Abstract.** The majority of graph visualization algorithms emphasize improving the readability of graphs by focusing on various vertex and edge rendering techniques. However, revealing the global connectivity structure of a graph by identifying significant vertices is an important and useful part of any graph analytics system. Centrality measures reveal the "most important" vertices of a graph, commonly referred to as central or influential vertices. Hence, a centrality-oriented visualization may highlight these important vertices and give deep insights into graph data. This paper proposes a mathematical optimization-based clustered graph layout called Near-Optimal Concentric Circles (NOCC) layout to visualize medium to large scale-free graphs. We cluster the vertices by their betweenness values and optimally place them on concentric circles to reveal the extensive connectivity structure of the graph while achieving aesthetically pleasing layouts. Besides, we incorporate different edge rendering techniques to improve graph readability and interaction.

**Keywords:** Graph Visualization · Connectivity · Layout Algorithm · Scale-free Networks.

## 1 Introduction

Recent advancements in data collection have been producing big complex data modeled as graphs (entities (vertices/nodes) and their relationships (edges/links)). Effective analysis of medium-to-large graphs is gaining popularity in many application domains, including social sciences, engineering, and natural sciences. Human ability to identify and comprehend visual patterns makes visualization a critical tool to understand graphs, and many studies used it as an effective tool to improve perception in graph exploration [26, 14, 15, 19]. A reasonably well-drawn graph visualization will help users to quickly get deeper insights into the data by highlighting existing patterns and revealing hidden patterns.

Many visualization algorithms such as force-based for large-scale graphs result in space-filling, cluttered visualization like a hairball [11]. This visual clutter, due to overlapping vertices and a large number of edge-crossings, makes visualizations hard to read and interpret. Rendering a large number of vertices in a

*This is an author-formatted version. The final version is at www.springerlink.com, ISVC 2020, LNCS, vol 12510, pp. 570–580. Springer, Cham.*

small space increases vertex overlaps and aggravates the visual clutter.A variety of methods have been proposed to address this problem and the most promising solution is attribute-based clustering of the vertices for visualization [18]. Vertex clustering creates simpler visualizations by grouping vertices with similar attributes to organize the graph in a presentable way. The majority of the vertex-cluster based graph layout algorithms focus on graph readability by reducing vertex overlaps, increasing intra-cluster edges, reducing inter-cluster edges, and rendering only a portion of the edges [18, 25]. They do not attempt to highlight the intrinsic structures of graphs - which is the main focus of our work.

Another solution for clutter reduction is to consider the layout generation as a mathematical optimization problem [30]. An objective function with constraints is defined on the vertex positions based on certain graph drawing aesthetics such as constant edge length. Layout generation algorithms consider these constraints and try to minimize a cost function to produce the final layout. Finding an optimal solution for this minimization cost function is NP-hard in most of the cases. Therefore, heuristics are employed to attain a *"near-optimal"* or an approximate solution for the immediate goal of clutter reduction in graph drawings.

In this paper, we present a novel *scale-free* graph visualization technique that features *centrality-based* clustering of the vertices, called Near-Optimal Concentric Circles (NOCC) layout. Large, scale-free graphs appear in many real-world systems including social, computer and biological networks [8, 28, 12, 34, 20, 32]. These graphs exhibit a power-law vertex degree distribution $P(k) \sim k^{-\gamma}$ where $P(k)$ is the probability that a vertex has $k$ links, and $\gamma$ is the degree exponent which is typically in the range (2,3) [24]. This indicates that a small portion of the vertices in these graphs have very high degrees while many vertices have low degrees. Centrality measures contextually identify influential vertices, and highlighting these vertices is useful in analyzing the topological structures of graphs. There are many centrality measures defined on graphs. Choosing the appropriate one depends on the problem at hand. In our approach, we choose the *betweenness* centrality to generate the vertex clusters as betweenness reveals the critical vertices that effectively connect other pairs of vertices in a graph.

After clustering the vertices based on their betweenness values, we present a concentric circle layout generation algorithm using mathematical optimization that places the vertices on circles representing the clusters. Finally, we combine graph coloring and partial edges to show that our algorithm generates aesthetically pleasing layouts.

Our main contribution in this study is a novel betweenness-based central-vertex graph layout algorithm for scale-free graphs using mathematical optimization that naturally shows the intrinsic connectivity structure by highlighting influential vertices. We use polar coordinates to enforce the constraints strictly rather than in a lazy manner [6].

## 2 Related Work

The history of graph visualization can be traced back over centuries. One of the earliest and the most fundamental visualizations of graphs is a node-link diagram, where vertices are represented as dots and relations between vertices as lines. Knuth's flowchart drawing paper [23] is considered one of the seminal works on graph visualization algorithms.

Ahmed et al. [1] present a variation of a fast-force-directed layout to visualize scale-free graphs in three dimensions. Vertices are constrained to parallel planes or on the surface of a sphere to minimize occlusion. Jia et al. [21] present a better interactive layout for large scale-free graphs that filters a large portion of less important edges while preserving other important features of the graph. Andersen et al. [2] present an algorithm that focuses on partitioning edges into local and global sets. Local or global edges are defined by the size of the maximum short flow between the edges' endpoints. These sets are visualized using a force-directed method emphasizing local edges. Baur et al. [3] describe a 2.5D method where the core hierarchy is partitioned into $k$-cores and these cores are used to visualize the graph structure. Cores are represented by 2D layouts. The inter-dependence for increasing $k$ value is the third dimension, drawn using spectral layout starting with cores with the maximum value. Chan et al. [7] present Out-Degree Layout (ODL) by separating the vertices into multiple hierarchical layers based on the outdegree of each vertex. They demonstrate that their algorithm can produce aesthetically pleasing layouts by naturally drawing related vertices closer to each other. Giot et al. [16] present a layout algorithm that emphasizes cores of very large graphs. They use a combination of the hierarchical coreness decomposition with existing layout algorithms according to cluster topologies to produce vertex-overlap-free drawings. Takac et al. [27] propose a scalable, fast, and easy graph visualization layout called radius degree layout (RDL). Vertices are randomly placed on the circle arc and the distance from the center of the visualization to the vertex is inversely proportional to its degree.

Concentric circle-based visualization is a popular layout in visualizing rooted trees [13]. The root vertex of the tree is at the center and the descendant vertices are placed on subsequent rings (circles), similar to a concentric circle layout. It is used in other layouts like [9] and [6] to reflect the distance between entities of a graph. Chou et al. [9] visualizes the closest neighbors of a selected paper (vertex). The selected vertex is shown at the center and vertices (papers) related to the selected one are shown on a number of concentric discs according to their distance. Vertices are binned into categories, and each ring (circle) represents a bin. Castermans et al. also present a concentric circle based visualization of the closest neighbors of a selected entity [6]. The distances represent the *"near-exact"* distances between the neighbors and the selected vertex. Although [9] and [6] are concentric circle-based layouts, they focus on showing only a small portion of a graph: the vertex of interest and its neighborhood. This is different from our work, where we focus on revealing the extensive connectivity structure of a scale-free graph by identifying the important vertices based on betweenness.

Most of the discussed related work presents layout techniques to improve the readability of scale-free graphs via an existing layout algorithm or showing only a small portion of the graph. In [1, 17, 2] the authors present the results on small to medium-size graphs and the scalabilities of these algorithms are not discussed. In [21], the authors present vertex filtering to avoid rendering all vertices. Interested readers can find more related works in [4] and [10].

In this work, we present a novel concentric circle based graph layout algorithm that emphasizes relatively important vertices based on the betweenness centrality. In this process, we use the benefit of global optimization - not dependent on initial positions or points.

## 3 Near-Optimal Concentric Circles Layout

NOCC places vertices in a 2-dimensional space, with position depending on betweenness. NOCC uses the betweenness centrality - an indicator to identify relatively important vertices emphasizing extensive connectivity structure of scale-free graphs. Removal of a vertex with high betweenness can potentially disconnect a graph. For example, in router-level Internet topology graphs, failure of a router with high betweenness centrality will cause significant service disruptions until recovery processes occur.

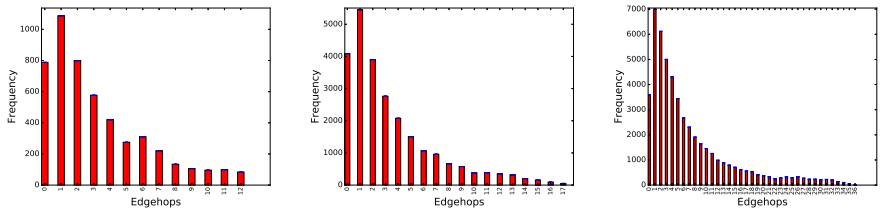
Vertex betweenness gives the scale to which a vertex is present on the shortest paths between pairs of other vertices using the following formula:

$$C_b(v) = \sum_{\substack{s \neq v \neq t \\ s,v,t \in V}} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{1}$$

where $\sigma_{st}$ is the total number of shortest paths from vertex $s$ to $t$ and $\sigma_{st}(v)$ is the total number of these paths through $v$.

We used the ck-means algorithm [29] to cluster the vertices using different centralities - degree, stress, betweenness, closeness, and variations of eigencentrality, to study the inter-cluster relations. We define *Edgehop* as the difference in the cluster number of the two endpoints of an edge. For example, an edge from a vertex in cluster 2 to a vertex in cluster 4 has edgehop of $4 - 2 = 2$. Edgehop is used to identify patterns in inter-cluster and intra-cluster edges. We used the ck-means algorithm [29] to cluster the vertices using betweenness centrality to study the *Edgehop* relations and found a pattern as shown in Fig. 1 to visualize graphs in small display areas. These figures indicate that:

(i) Most of the edges are between the vertices in the outermost cluster that is congruent with the number of vertices at the edge of the scale-free graphs.
(ii) Very few edges span from the outermost toward the innermost clusters.
(iii) Significant number of edges are between the vertices in the same cluster (edgehop = 0) or in the immediately neighboring cluster (edgehop = 1), or in the clusters that are close to each other (edgehop = 2 or 3).




(a) (b) (c)

Fig. 1: Charts showing the edgehop histograms. Edgehop histogram for graphs with (a) $|V| = 1000$, (b) $|V| = 5000$ (c) $|V| = 10000$.

To visualize the clusters in the final layout, we use a concentric circles layout, as it naturally highlights the important vertices relevant to the connectivity structure of a graph while conveying the hierarchy of importance. Vertices with high betweenness values are drawn on circles closer to the common center and vice-versa. Vertices having similar importance are visualized on the same circle. The main idea in generating the NOCC layout are:

(i) For each cluster, generate random vertex positions.
(ii) Generate final vertex positions by minimizing the total edge length.
(iii) Render the final layout to place the vertices in an optimal way.

Therefore, cluster 1 has the vertices with the highest betweenness.

### 3.1 Global Optimization Function

The next challenge is to place vertices on their respective circles to achieve an aesthetically pleasing visualization to convey the connectivity information. Note that a particular placement of vertices on their circles affects the placement of their neighboring vertices cascadingly. We modeled the layout calculation step as a global minimization (optimization) problem with constraints to produce easily readable layouts.

After generating vertex clusters, $\{c_1, c_2, c_3, \ldots, c_m\}$, the vertices are placed on concentric circles according to the clusters that they belong to. We aim for a near-optimal placement of the vertices of a graph $G(V, E)$ on these circles. We calculate the length of each edge and minimize the total edge length of the graph. This will also reduce a considerable amount of long edges, which will, in turn, reduce edge-crossings.

The coordinates of a vertex $v_i \in V$, denoted by denoted by $(x_i, y_i) \in \mathbb{R}^2$, are represented using the Polar coordinates, (r, $\theta$). Vertices are placed on concentric circles/rings, $\{c_1, c_2, c_3, \ldots, c_m\}$, centered at the origin. As all the circles in the visualization are origin-centered, the coordinates of a vertex $v_i$ on a circle of radius $r_i$ at an angle $\theta_i$ are given by:

$$\begin{aligned} x_i &= r_i \cos(\theta_i) \\ y_i &= r_i \sin(\theta_i) \end{aligned} \tag{2}$$

The distance between two points $i, j$ at equal distance from the center, $r$ is given by:

$$d = 2r \sin(\frac{\theta_j - \theta_i}{2}) \tag{3}$$

The following optimization will meet the rationale of placing the neighboring vertices closer to each other by minimizing the total edge length:

$$\begin{aligned} &\text{minimize} && \sum_{(v_i,v_j)\in E} r_i^2 + r_j^2 - 2r_i r_j \cos(\theta_j - \theta_i) \\ &\text{subject to} \\ & && 2\pi r_k \frac{\theta_{k'} - \theta_k}{2\pi} \geq 2a, \quad \forall v_k, v_{k'} \in c_k \end{aligned} \tag{4}$$

where $r_k$ is the radius of the circle for cluster $c_k$ and $a$ is the radius of the filled circles representing the vertices. The constraint ensures that the vertices on the same cluster circle do not overlap. Specifically, the distance between any two vertices of a cluster is at least two times the radius of a vertex.

For each edge, the endpoint vertices are constant throughout the optimization process and thus the radii are constant. The optimization problem is a non-linear optimization problem with a linear inequality constraint as presented in 4.

### 3.2 Generating Vertex Positions

We use the NLOPT[1] library, and specifically its Improved Stochastic Ranking Evolution Strategy (ISRES). The original algorithm can be found here[2].

The layout generation is a two-step process shown in Algorithm 1. The first step generates the layout for the vertices in the innermost cluster. The second step generates the layout for the vertices in the remaining clusters.

Graph vertex clusters and the concentric circle radii are inputs to Algorithm 1 and graph layout is the output. Line 1 places the vertices of the innermost cluster equi-angularly, which are fixed throughout the next steps. The vertices in the innermost cluster are placed at equal arc distances as they are the most important vertices in terms of connectivity and we want to highlight them for easy viewing and interactions using a mouse. Moreover, they serve as the invariants between multiple runs of the optimization algorithm. Line 2 will set the optimization algorithm to ISRES. Line 3 sets the objective function along with its constraints. Line 4 will set the termination threshold which is 0.001 in our experiments. Lines 6 to 15 define the second step in the process that will generate

[1] http://ab-initio.mit.edu/wiki/index.php/NLopt
[2] http://www3.hi.is/ tpr/papers/RuYa05.pdf

**Algorithm 1** Layout Generation Algorithm

**Require:** Graph Vertex Clusters
**Require:** Radii of the concentric circles (rings)
**Ensure:** Graph layout Set the positions of the vertices in cluster 1
1: Place cluster 1 vertices at equal angles on ring 1 Run the optimization for the vertices on ring 1
2: Optimization Algorithm: ISRES
3: Set the objective function and inequality constraints
4: Set the termination condition
5: Place the vertices in each cluster
6: **for** *each cluster from* 2 *to* $m$ **do**
7: Place the vertices at equal angles on their rings
8: **for** *each vertex*$(v)$ *belongsto* $i$ **do**
9: Set position bounds to [0,2$\pi$]
10: **end for**
11: **end for**
12: Run the optimization for all vertices on rings 2 to $m$
13: Set the objective function and inequality constraints
14: Set the termination condition
15: Run the algorithm to produce the final layout

the final layout. Line 6 to 11 place the vertices of rings 2 to $m$ equi-angularly on the rings and set the bounds for the position of each vertex. The optimization procedure described at lines 13 to 15 is similar to the procedure described at lines 2 to 4. Finally, line 15 runs the optimization function with all the above parameters to produce the final layout.

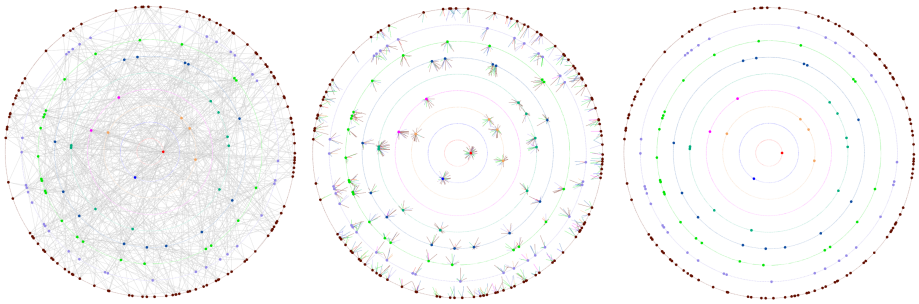
Since the algorithm runs for a limited number of iterations to reach the termination threshold, the output of the vertex placement is not necessarily optimal, so we call it "near-optimal".

### 3.3 Graph Rendering

**Size and Colors of Vertices** Vertex colors are assigned according to their corresponding clusters. Colors are chosen in-order from a list to distinguish the vertices of a cluster from its two immediate neighboring clusters to improve the visual clarity. The cluster ring has the same color as its vertices.

**Cluster Radius** For the radius of each cluster, we tested various radii options and concluded that the visualization is aesthetically pleasing with a constant increment of the cluster radii. If the radius of a ring is too large to fit the display screen, we can re-space rings, re-cluster the vertices on a ring, or allow vertex overlapping.

**Edge Rendering** We render two edge styles. The first is a line segment between two vertices drawn in grey color. Figure 2a shows a synthetic graph with all the vertices and complete edges rendered. The second edge style is partial edge rendering (PED), similar to [25] and [5], drawn to improve the readibility of the NOCC layout visualizations. A partial edge is an edge that is drawn only around its incident vertices. It has become a popular style to curtail edge-crossings for clutter reduction [5]. The default partial edge length is 10% of the actual



(a) (b) (c)

Fig. 2: NOCC Layout for a computer-generated graph with 200 vertices and 597 edges (a) Complete edges, (b) Partial Edges, (c) Just the vertices without edges.

edge length. At each vertex, a partial edge is drawn with the same color of the vertex it is connected to and the direction pointing towards the connected vertex. Figure 2b shows the same layout from Figure 2a with partial edges as described. This rendering will give the user an idea of the degree of each vertex and the clusters of its neighbors through the partial edge colors. Lastly, figure 2c shows only the vertex positions generated by NOCC without any edges. Users can interactively switch between the three types of visualizations.
The main features of the layout are:

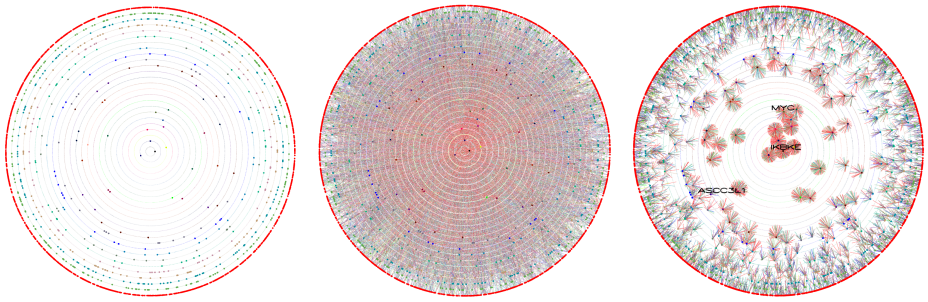
(i) A concentric circle-based 2D layout where each circle represents a cluster.
(ii) Clusters are based on the betweenness values of vertices. The cluster with the smallest radius has the vertices with the highest betweenness. As we move away from the center, the betweenness' of vertices decreases.
(iii) All edges are rendered and PED of each vertex show the directions to the connected vertices. Lengths of partial edges can be tuned.
(iv) All vertices are the same size, and vertices of a cluster have the same color.

## 4 Case Study - Human Protein-Protein Interactions

We present the visualization of a real-world graph: the Human Binary Protein-Protein Interactions (PPI) dataset, freely available from Koblenz Network Collection[3]. Proteins rarely function in isolation and their interactions are of great interest to the biological research community. Protein-Protein Interactions (PPI) are crucial for understanding protein functions, analyzing biological processes, and examining diseases. PPI networks enable study of the characteristics of proteins and their interactions by representing proteins as nodes and their interactions as edges.

In [33] the authors identify bottleneck proteins, which have high betweenness and low degree centralities in PPI networks. These proteins are more likely to be

[3] http://konect.uni-koblenz.de/networks/maayan-figeys




(a) (b) (c)

Fig. 3: NOCC Layout of Human Binary Protein-Protein Interactions graph consisting of 2217 vertices and 6418 edges (a) Complete graph without the edges (b) Complete graph with edges (c) Complete graph with partial edges.

essential for crucial functional and dynamic properties [22], [33]. It is reported that drug-target proteins have high degree and/or betweenness [31]. Therefore, identifying and visualizing the bottleneck proteins will help biologists to examine (i) if these proteins are the targets of various pathogens and (ii) if they can target these proteins to treat certain diseases.

Figure 3c presents the Human Binary Protein-Protein Interactions graph consisting of 2217 vertices and 6418 edges drawn using the NOCC layout with partial edges as a case study. The vertices (proteins) on the center rings (clusters) are the vertices with higher betweenness values. These are the major influential proteins such as IKBKE, MYC and ASCC3L1 through which a lot of communication takes place. In Figure 3, the contextual importance of the vertices decreases as we move away from the center toward the outer circles. Yu et al. divide proteins into four categories based on their degree and betweenness centralities [33]: (i) nonhub-nonbottlenecks, (ii) hub-nonbottlenecks, (iii) nonhub-bottlenecks and (iv) hub-bottlenecks. The NOCC layout naturally demonstrates those important bottleneck proteins that need to be studied further by domain experts. IKBKE in Figure 3c is one of the very important proteins in treating cancer and inflammatory diseases. This is a hub-bottleneck protein with degree 314 and betweenness 295061. Another significant protein naturally revealed in Figure 3c is MYC and it is a nonhub-bottleneck protein. MYC has a relatively low degree of 115 with a high betweenness value of 122303. Similarly, ASCC3L1 in Figure 3c plays an essential role in pre-mRNA splicing. It has a very low degree of 14 with a high betweenness value of 34333.2.

## 5 Conclusions And Future Work

We summarized a visualization model that generates a central-vertex based layout while reducing the visual clutter and improving the visual information. We described a graph layout algorithm that reveals the extensive connectivity struc-

tures of scale-free graphs by naturally highlighting relatively important vertices with a tightly connected core vertex group. The NOCC layout can also be used in graph resilience analysis under targeted vertex failures. It naturally highlights the vertices that play the key role in graph connectivity. Although previous works attempted to generate better layouts to reduce the visual clutter, they weren't successful in highlighting the intrinsic connectivity structures of graphs.

Future work include formal comparison of NOCC with similar concentric circle and large graph visualization layouts to study its usefulness in terms of visual quality, layout generation speed and interactivity.

## 6 Acknowledgements

This work was supported by the National Science Foundation under Grant Number 1429526 and by the Louisiana Board of Regents Support Fund under contract LEQSF(2019-20)-ENH-DE-22.

## References

1. Ahmed, A., Dwyer, T., Hong, S.H., Murray, C., Song, L., Wu, Y.X.: Visualisation and analysis of large and complex scale-free networks. In: EuroGraphics / IEEE EuroVis (2005)
2. Andersen, R., Chung, F., Lu, L.: Drawing power law graphs. In: International Symposium on Graph Drawing. pp. 12–17. New York, NY, USA (Oct 2004)
3. Baur, M., Brandes, U., Gaertler, M., Wagner, D.: Drawing the AS graph in 2.5 dimensions. In: International Conference on Graph Drawing. pp. 43–48 (Sep 2004)
4. Bertini, E.: Social networks visualization: A brief survey (2005)
5. Binucci, C., Liotta, G., Montecchiani, F., Tappini, A.: Partial edge drawing: Homogeneity is more important than crossings and ink. 2016 7th International Conference on Information, Intelligence, Systems And Applications(IISA) pp. 1–6 (2016)
6. Castermans, T., Verbeek, K., Speckmann, B., Westenberg, M.A., Koopman, R., Wang, S., van den Berg, H., Betti, A.: Solarview: Low distortion radial embedding with a focus. IEEE Transactions on Visualization and Computer Graphics **25**, 2969–2982 (2018)
7. Chan, D.S.M., Chua, K.S., Leckie, C., Parhar, A.: Visualisation of power-law network topologies. In: IEEE International Conference on Networks. pp. 69–74. Sydney, Australia (Sept 2003)
8. Chiasserini, C.F., Garetto, M., Leonardi, E.: Social network de-anonymization under scale-free user relations. IEEE/ACM Trans. Netw. **24**(6), 3756–3769 (2016), `http://dblp.uni-trier.de/db/journals/ton/ton24.html#ChiasseriniGL16`
9. Chou, J.K., Yang, C.K.: PaperVis: Literature Review Made Easy. Computer Graphics Forum (2011). https://doi.org/10.1111/j.1467-8659.2011.01921.x
10. Correa, C.D., Ma, K.L.: Visualizing Social Networks, pp. 307–326. Springer US, Boston, MA (2011)
11. Dianati, N.: Unwinding the hairball graph: Pruning algorithms for weighted complex networks. Phys. Rev. E **93**, 012304 (Jan 2016)
12. Ding, Y., Li, X., Tian, Y., Ledwich, G., Mishra, Y., Zhou, C.: Generating scale-free topology for wireless neighbourhood area networks in smart grid. IEEE Transactions on Smart Grid pp. 1–1 (2018). https://doi.org/10.1109/TSG.2018.2854645

13. Eades, P.: Drawing Free Trees. IIAS-RR, International Institute for Advanced Study of Social Information Science, Fujitsu Limited (1991)
14. Etemad, K., Samavati, F., Carpendale, S.: Daisy visualization for graphs. In: Proceedings of the Joint Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering. pp. 103–112. Expresive '16, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2016), http://dl.acm.org/citation.cfm?id=2981324.2981340
15. Fujita, Y., Fujiwara, Y., Souma, W.: Visualizing large-scale structure of a million-firms economic network. In: SIGGRAPH Asia 2015 Visualization in High Performance Computing. pp. 13:1–13:4. SA '15, ACM, New York, NY, USA (2015). https://doi.org/10.1145/2818517.2818525, http://doi.acm.org/10.1145/2818517.2818525
16. Giot, R., Bourqui, R.: Fast graph drawing algorithm revealing networks cores. In: International Conference on Information Visualisation. pp. 259–264. Barcelona, Spain (Jul 2015)
17. Guo, X., Chen, H., Liu, X., Xu, X., Chen, Z.: The scale-free network of passwords : Visualization and estimation of empirical passwords. CoRR **abs/1511.08324** (2015)
18. Ho, J., Hong, S.H.: Drawing clustered graphs in three dimensions. Lecture Notes in Computer Science **3843**, 492–502 (2006). https://doi.org/10.1007/11618058_44
19. Itoh, T., Klein, K.: Key-node-separated graph clustering and layouts for human relationship graph visualization. IEEE Computer Graphics and Applications **35**(6), 30–40 (2015), http://dblp.uni-trier.de/db/journals/cga/cga35.html#ItohK15
20. Jeong, H., Mason, S., Barabási, A.L., Oltvai, Z.: Lethality and centrality in protein networks. Nature **411** (2001)
21. Jia, Y., Hoberock, J., Garland, M., Hart, J.: On the visualization of social and other scale-free networks. IEEE Transactions on Visualization and Computer Graphics **14**(6), 1285–1292 (Nov 2008)
22. Joy, M., Brock, A., E Ingber, D., Huang, S.: High-betweenness proteins in the yeast protein interaction network. Journal of biomedicine & biotechnology **2005**, 96–103 (Jul 2005)
23. Knuth, D.E.: Computer-drawn flowcharts. Communications of the ACM **6**(9), 555–563 (Sep 1963)
24. Newman, M.: Power laws, Pareto distributions and Zipf's law. Contemporary Physics **46**(5), 323–351 (2005)
25. Sathiyanarayanan, M., Pirozzi, D.: Social network visualization: Does partial edges affect user comprehension? In: International Conference on Communication Systems and Networks. pp. 570–575 (Jan 2017)
26. Schulz, C., Nocaj, A., Görtler, J., Deussen, O., Brandes, U., Weiskopf, D.: Probabilistic graph layout for uncertain network visualization. IEEE Trans. Vis. Comput. Graph. **23**(1), 531–540 (2017), http://dblp.uni-trier.de/db/journals/tvcg/tvcg23.html#SchulzNGDBW17
27. Takac, L., Zabovsky, M.: Radius degree layout - fast and easy graph visualization layout. In: International Conference on Digital Technologies. pp. 338–343. Zilina, Slovakia (July 2014)
28. Verma, T., Araújo, N.A.M., Herrmann, H.J.: Revealing the structure of the world airline network. CoRR **abs/1404.1368** (2014), http://dblp.uni-trier.de/db/journals/corr/corr1404.html#VermaAH14

29. Wang, H., Song, M.: Ckmeans.1d.dp: Optimal k-means clustering in one dimension by dynamic programming. R Foundation for Statistical Computing **28**(17), 29 (2012)
30. Ware, C., Purchase, H., Colpoys, L., McGill, M.: Cognitive measurements of graph aesthetics. Information Visualization **1**(2), 103–110 (2002)
31. Yamada, T., Bork, P.: Evolution of biomolecular networks lessons from metabolic and protein interactions. Nature Reviews Molecular Cell Biology **10**, 791–803 (2009)
32. Yu, H., Greenbaum, D., Lu, H.X., wei Zhu, X., Gerstein, M.: Genomic analysis of essentiality within protein networks. Trends in genetics : TIG **20 6**, 227–31 (2004)
33. Yu, H., Kim, P.M., Sprecher, E., Trifonov, V., Gerstein, M.: The importance of bottlenecks in protein networks: Correlation with gene essentiality and expression dynamics. PLoS Computational Biology **3**(4) (2007)
34. Zhang, B., Liu, R., Massey, D., Zhang, L.: Collecting the internet as-level topology. SIGCOMM Comput. Commun. Rev. **35**(1), 53–61 (Jan 2005). https://doi.org/10.1145/1052812.1052825, `http://doi.acm.org/10.1145/1052812.1052825`