Complex Network Analysis
Project Report

26.05.2013

09411018
Mehmet AKTAŞ
maktasceng@gmail.com
https://github.com/maktas/GephiTest/tree/master/HiveplotModule

A Hiveplot Layout Implementation for Gephi

A HIVEPLOT LAYOUT IMPLEMENTATION FOR GEPHI

1. DESCRIPTION OF THE PRINCIPLE

Force layouts are not very meaningful in the sense that they assign positions to nodes. The position is just random and it is hoped that related nodes appear next to each other. Force layouts are interpreted through intuition rather than reasoning or observation. They make poor use of the most effective visual channel -that is position of the nodes- and as a result they produce an incomprehensible hairball in the worst case. A hairball is a graph that has most of the nodes around the center and less number of nodes in the outer regions, hence looking like a hairball. Hairballs turn complex data into visualizations that are just as complex, or even more so. But, just because they look complex does not mean that they can provide more information. The explicit position encoding in hive plots has the potential to better reveal the network structure while giving additional information. Hive plots can also be extended to show aggregate relationships. The hive plots attempt to address the shortcomings of the conventional layouts. Since hive plots can be tuned, they can identify meaningful structural components of a network.

A hive plot is a network layout algorithm that uses a parallel coordinate plot in which axes are radially arranged and node position is based on structural properties of that node. In a hive plot, nodes are placed onto linear axes. Node-to-axis assignment and node-on-axis position are determined solely by network structure, node attributes or any other meaningful properties of the network. In other words, layout rules are defined by the user, based on properties that are meaningful to the user.

2. ALGORITHM AND IMPLEMENTATION

The algorithm of the hiveplot layout consists of 4 main steps:

1) The computation of the geometrical positions of the axes depending on the number of axes,
2) The computation of the nodes to be assigned to the axes depending on the axis assignment property and the intervals defined by the user,
3) The computation of the positions of the nodes on the axes depending on the on-axis ordering property.
4) If some nodes have the same position on the axis recalculation of their positions so that they look like a group.

The user selects Hiveplot Layout from the layouts window in Gephi and runs some statistics from the statistics window in order to calculate the desired property for the graph. After running the statistics, the user presses the Refresh button to get the newly generated columns on the combo boxes and then selects the axis assignment property and on-axis ordering property from the combo boxes and number of axes from the first slider. After these steps the user specifies the numerical interval from the selection sliders. For example, if s/he wants to assign the nodes to 3 axes according to their degrees and sort them on the axes according to their modularity s/he selects “Degree” in the first combo box, “Modularity” in the second combo box and selects 3 from the number of axes slider. If s/he wants that first axes containing the nodes having degree less than or equal to 10, second axes containing the nodes having degree between 10 and 20 and third axes containing nodes having degree greater than 20 s/he selects 10 from the first slider and 20 from the second slider.

When the layout algorithm is run, it firstly calculates the geometrical positions of the axes depending on the number of axes. Then, it sorts all of the nodes according to the axis assignment property and assigns them to the axes with respect to the intervals chosen. After that, it sorts the nodes on axes according to the on-axis ordering property and calculates their positions on the axes depending on their value. Finally, the algorithm investigates whether there are completely overlapping nodes because of having same values and if there are, it shifts the nodes slightly so that each of them can be seen and they look as a group.

3. TESTS AND THE RESULTS

The reliability of the algorithm is tested with Les Miserables.gml, Java.gexf and Karate.gml graph files and the corectness of the results were checked by ranking the nodes with color codes. Moreover, the labels and the attributes of the nodes in the data table were cross checked and the correctness of the algorithm was determined. The results obtained from the tests can be seen in Figure 1, Figure2 and Figure 3 and Figure 4.

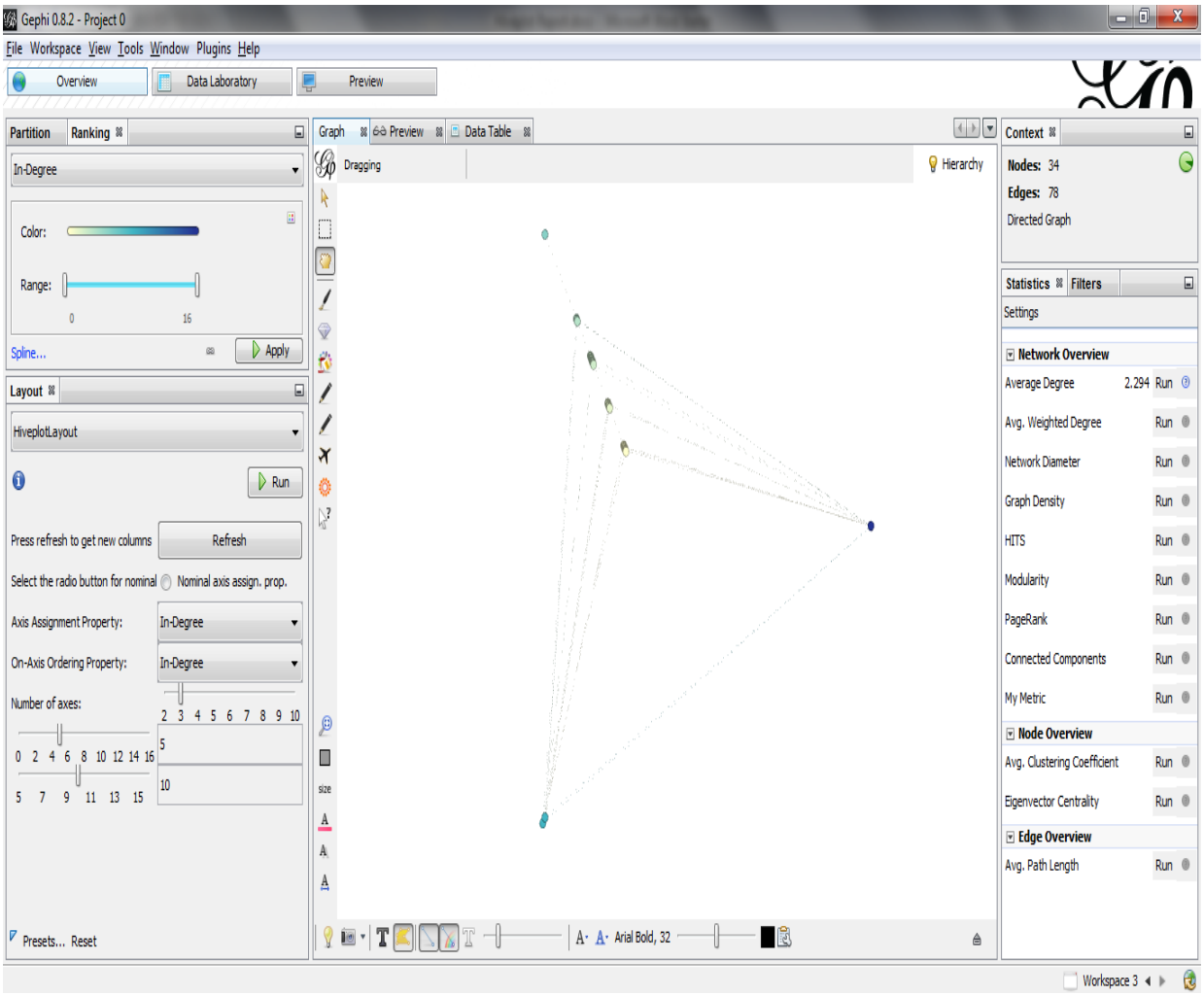

Figure 1 – Zachary's Karate Club dataset sorted by In-Degree Property

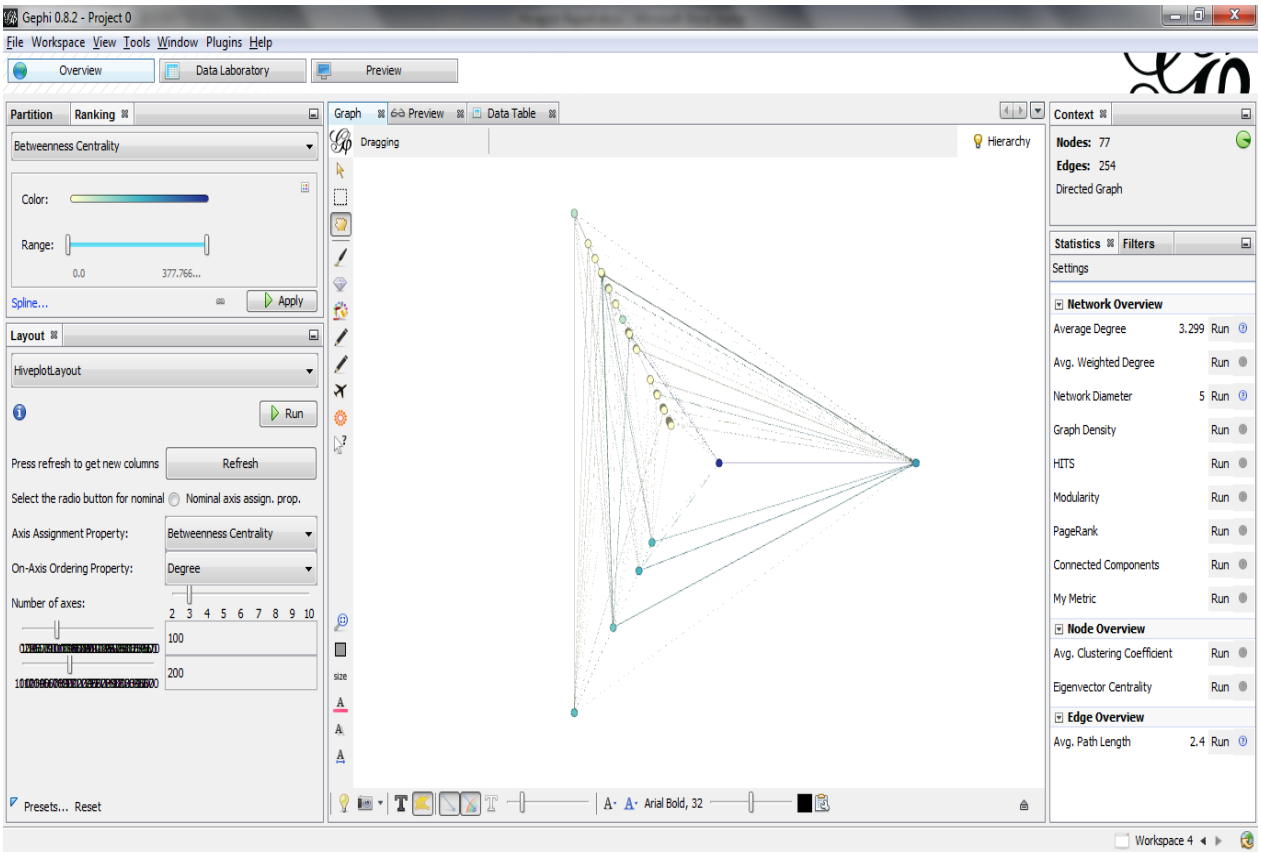

Figure 2 – Les Miserables dataset assigned to axes by Betweenness Centrality and sorted on axes by Degree property. Color code shows Betweenness Centrality.

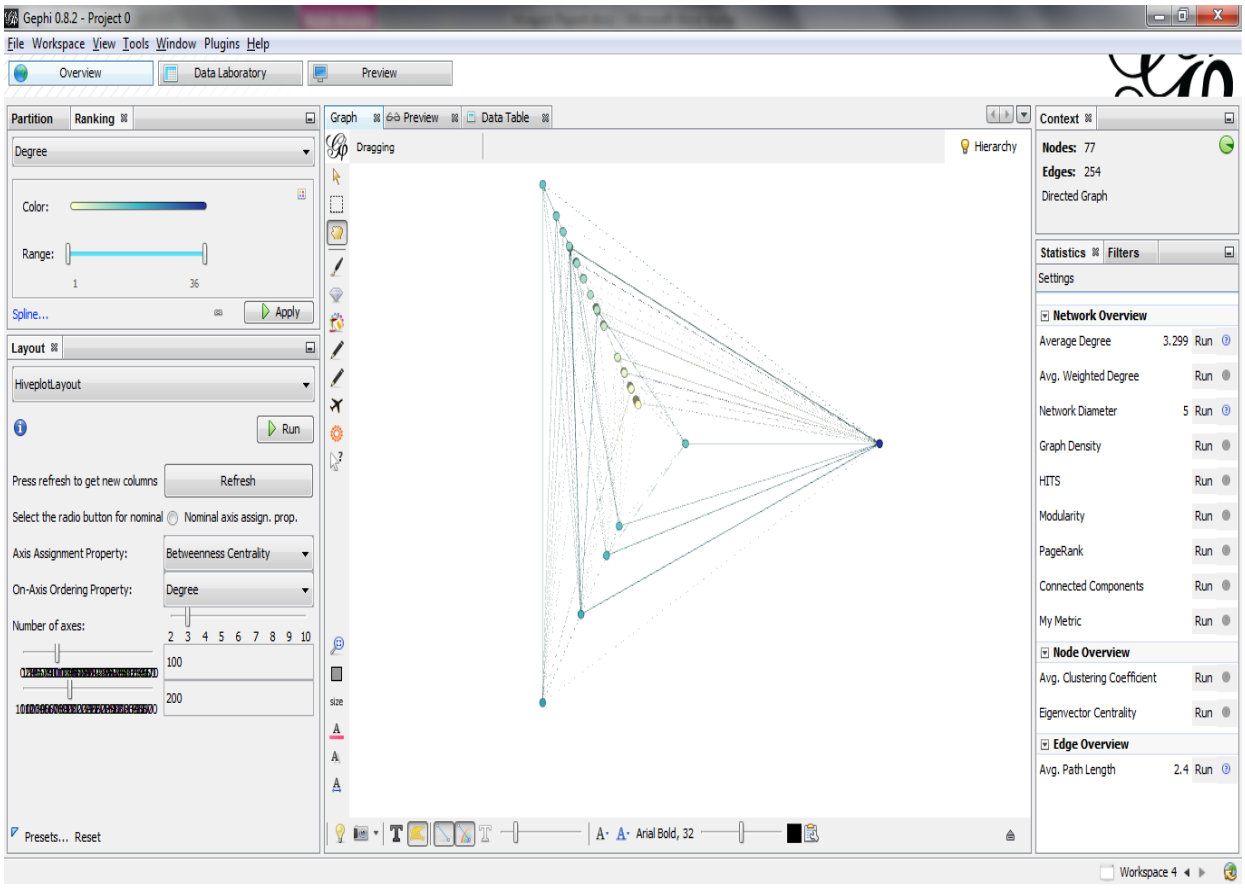

Figure 3 - Les Miserables dataset assigned to axes by Betweenness Centrality and sorted on axes by Degree property. Color code shows Degree.

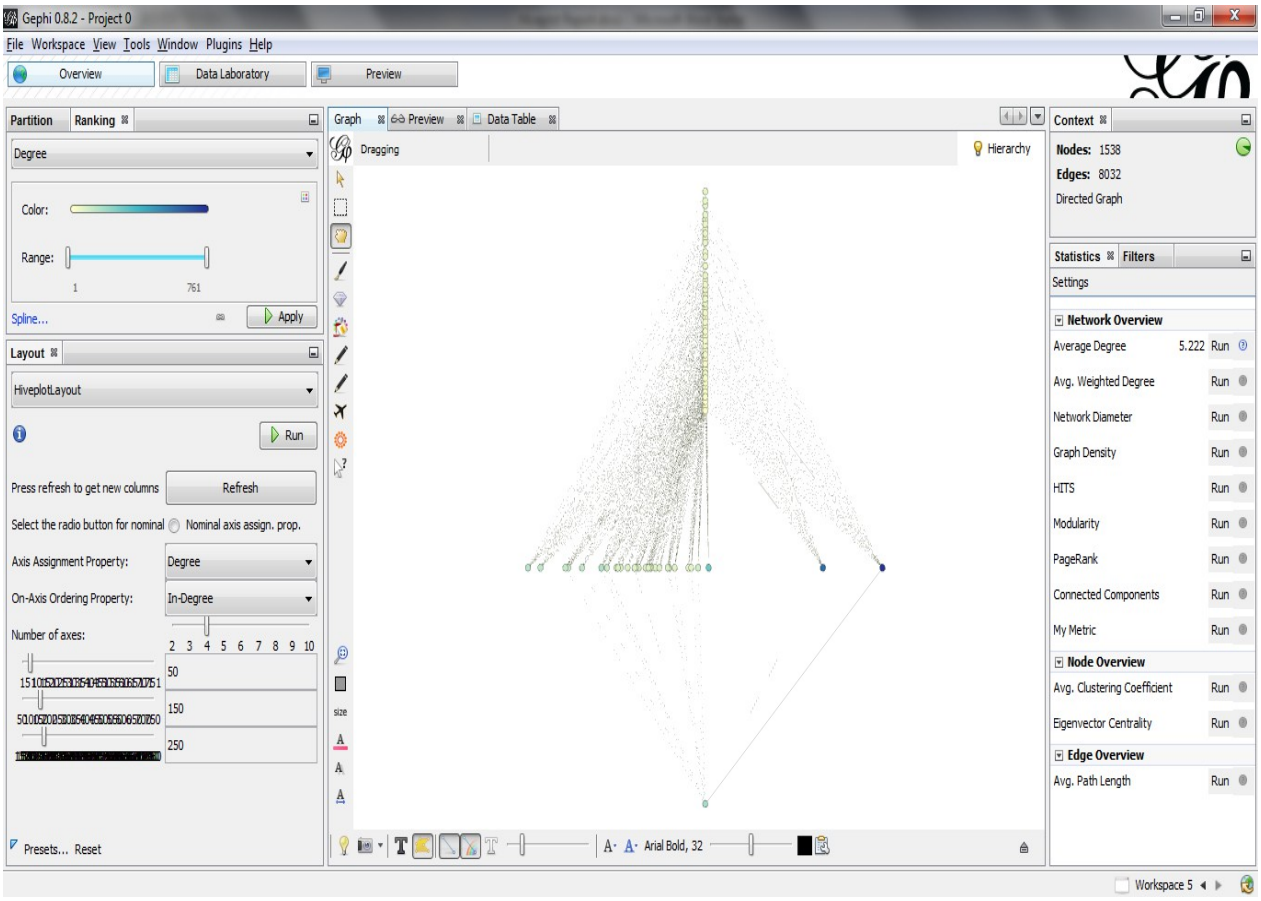

Figure 4 – Java Classes dataset assigned to axes by Degree and sorted on axes by In-Degree property. Color code shows Degree.

4. REFERENCES

1. Krzywinski M, Birol I, Jones S, Marra M (2011). Hive Plots — Rational Approach to Visualizing Networks. Briefings in Bioinformatics (9 December 2011, doi: 10.1093/bib/bbr069).
2. Engle S, Whalen S, Visualizing Distributed Memory Computations with Hive Plots. Proceedings of the 9th International Symposium on Visualization for Cyber Security (VizSec), (October 2012)
3. Hive Plots: Rational Network Visualization, www.hiveplot.com
4. JHive: A cross-platform interactive hive plot Java application, www.hiveplot.com/distro/jhive-0.0.16.zip
5. Gephi Wiki pages, https://wiki.gephi.org