Complex Network Analysis
Project Report

Multipartite Layout for Gephi

Kemal BEŞKARDEŞLER
11411004

MULTIPARTITE LAYOUT for GEPHI

1. PROBLEM

To simplify analyzing connections between groups in multipartite graphs, minimizing edge crossings is a good way to do so. In this work, by using the algorithm suggested in the paper “An Edge Crossing Minimization Algorithm Based on Adjacency Matrix Transformation”(1), implementation for gephi platform is performed. The detailed information about the algorithm and the usage of the layout is described in the next sections.

2. ALGORITHM

The main steps of the algorithm are as follows:

(1) Construct layers according to the chosen parameter of the nodes.
(2) Order layers according to the connections between them
(3) Create adjacency matrices
(4) While there is no more reduction in the number of total edge crossings,
For each adjacency matrix from top to bottom apply column transformation with less edge crossings
(5) While there is no more reduction in the number of total edge crossings;
For each adjacency matrix from bottom to top apply row transformation with less edge crossings
(6) End algorithm

After choosing the “Multipartite Layout” in Layout section of Gephi, in the properties panel speed of the algorithm and layer attribute can be chosen. The waiting time between each transformation is defined according to the speed selected (default value is 2 sec. for speed = 1.0). In the first step of the algorithm, the chosen layer attribute is used as the layer construction basis. If nothing is selected, nodes are automatically assigned to layers such that a multipartite graph is constituted without connections within each layer. If an attribute is specified, nodes with the same attribute value are assigned to the same layer.

In the second step, layers are ordered such that layers with more connections are positioned closer to each other. The possible orders are found by positioning each layer on top and trying to reach to the bottom layer by covering all layers. Due to performance issues, the number of possible orders for each layer placed on top is limited with 100(2). To create adjacency matrices, connection between each adjacent layers must exist. For each possible order of layers, the number of edges between layers that are not adjacent (extra edges) is calculated. As this number gets smaller, the layers with more connections between each other get closer. While calculating total number of edge crossings, these edges between not adjacent layers are omitted.

In the third step, the order with minimum number of extra edges is used to create adjacency matrices. For n layers (n – 1) adjacency matrices are created from top to bottom.

The fourth and fifth steps are the main part of the algorithm which reduce the total number of edge crossings. To decide which rows or columns to transform, row exchange and column exchange matrices are constructed and the effect of transformation in terms of diagonality is calculated using these matrices.

Example:

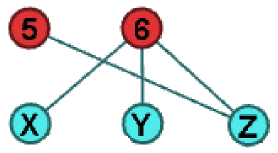


Figure 1

For the adjacent layers in Figure 1, the adjacency matrix, row exchange matrix and column exchange matrix are as follows, respectively:

$$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 2 & 1 \\ 3 & 2 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 3 \end{bmatrix}$$

Row and column exchange matrices are created by summing their distance to diagonal position.

For the row [0 0 1], its distance is 2 for being in the first row; and 1 for being in the second row.
For the row [1 1 1], its distance is 3 for being in the first row; and 2 for being in the second row.

For the first column $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$, its distance is 1 for being in the first column; 0 for being in the second column and 1 for being in the third column (same for the second column $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$).
For the third column $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$, its distance is 1 for being in the first column; 1 for being in the second column and 3 for being in the third column.

Exchange effect is calculated as follows:
For example, the effect of exchanging first column with third column is calculated as;

(First column being in the third column + Third column being in the first column) - (First column being in the first column + Third column being in the third column)
= (1 + 1) - (1 + 3) = -2

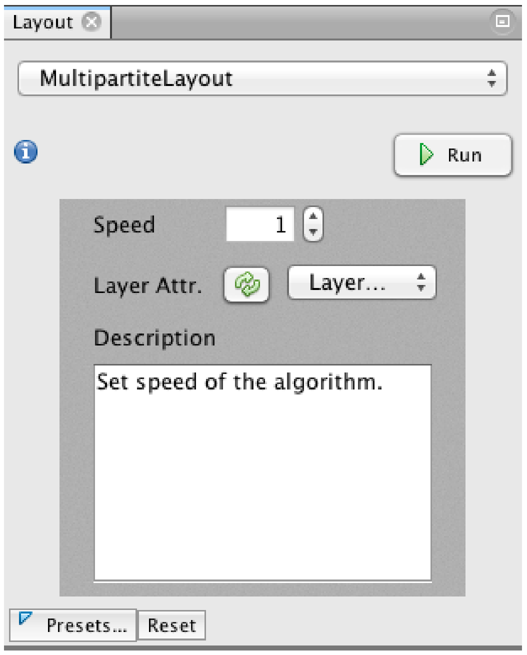
Negative value of the effect means a better diagonal alignment. Effect value of zero means no positive effect in terms of diagonality but it does not imply there will be no positive effect in terms of number of edge crossings. So, we also consider exchanges with effect value zero.

In the fourth step just column transformations are applied to all adjacency matrices from top to bottom. For each adjacency matrix, column exchange matrix is created and the best exchange is found. To perform the best transformation, the effects of all possible exchanges are calculated and transformation with the minimum effect value is applied. If transformation is performed and this adjacency matrix is not the last adjacency matrix, then the next adjacency matrix is reconstructed since its nodes are shifted.

When there is no more improvement in terms of number of edge crossings by appling column transformations, as the last step just row transformations are applied to all adjacency matrices from bottom to top. As for column transformation, this time upper adjacency matrix is recalculated if this adjacency matrix is not the first adjacency matrix.

Algorithm is terminated if there is no more reduction in the number of edge crossings.

3. USAGE



Set speed of the algorithm.

Figure 2 : Properties

- Select MultipartiteLayout from Layout section of Gephi.
- Set speed of the algorithm (optional)
- Select the attribute from the drop-down menu to be used as layer construction basis. (Optional)
- If another statistics are calculated during usage, use refresh button to refresh the drop-down menu to use that attribute.
- Use Reset button to set parameters to their default values.

4. RESULTS

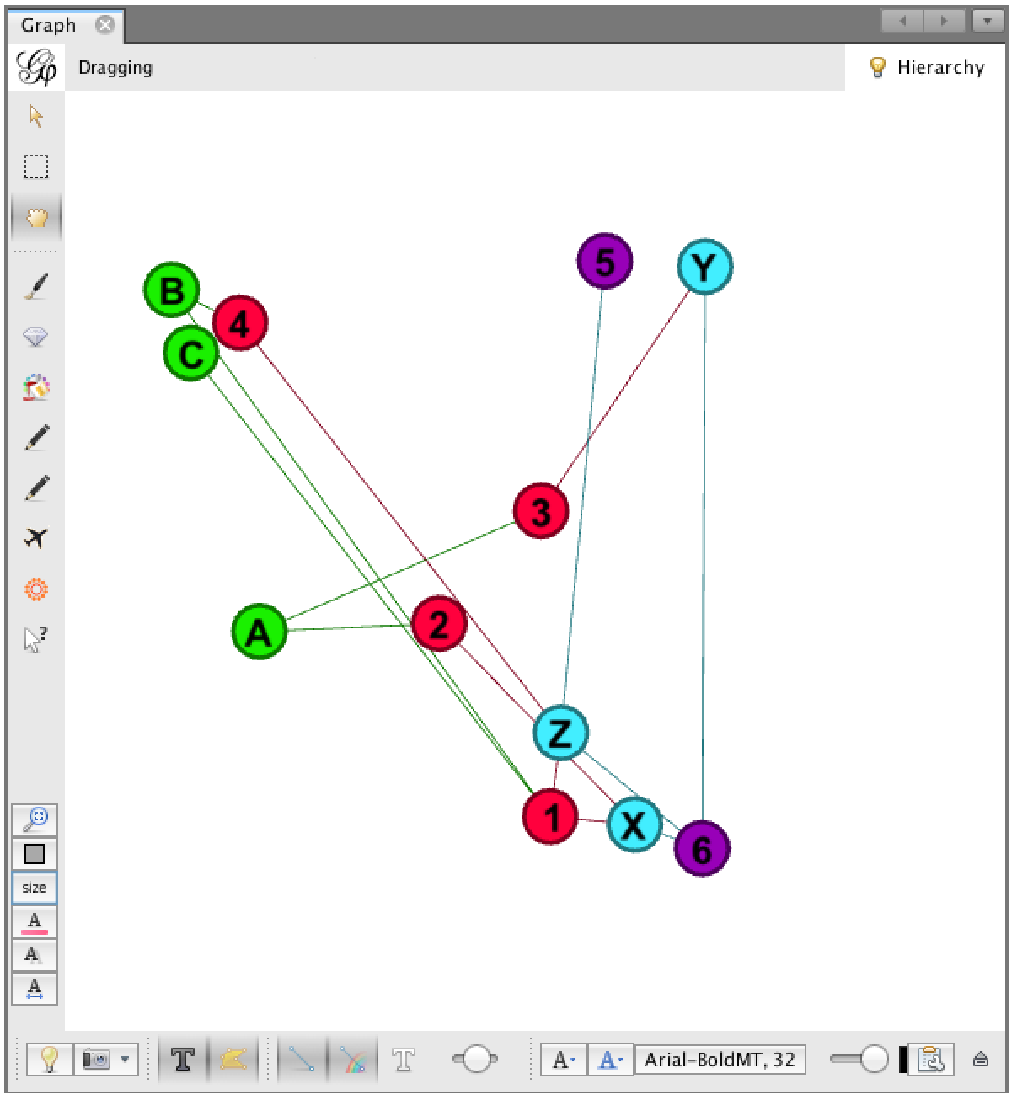


Figure 3: Unordered Graph

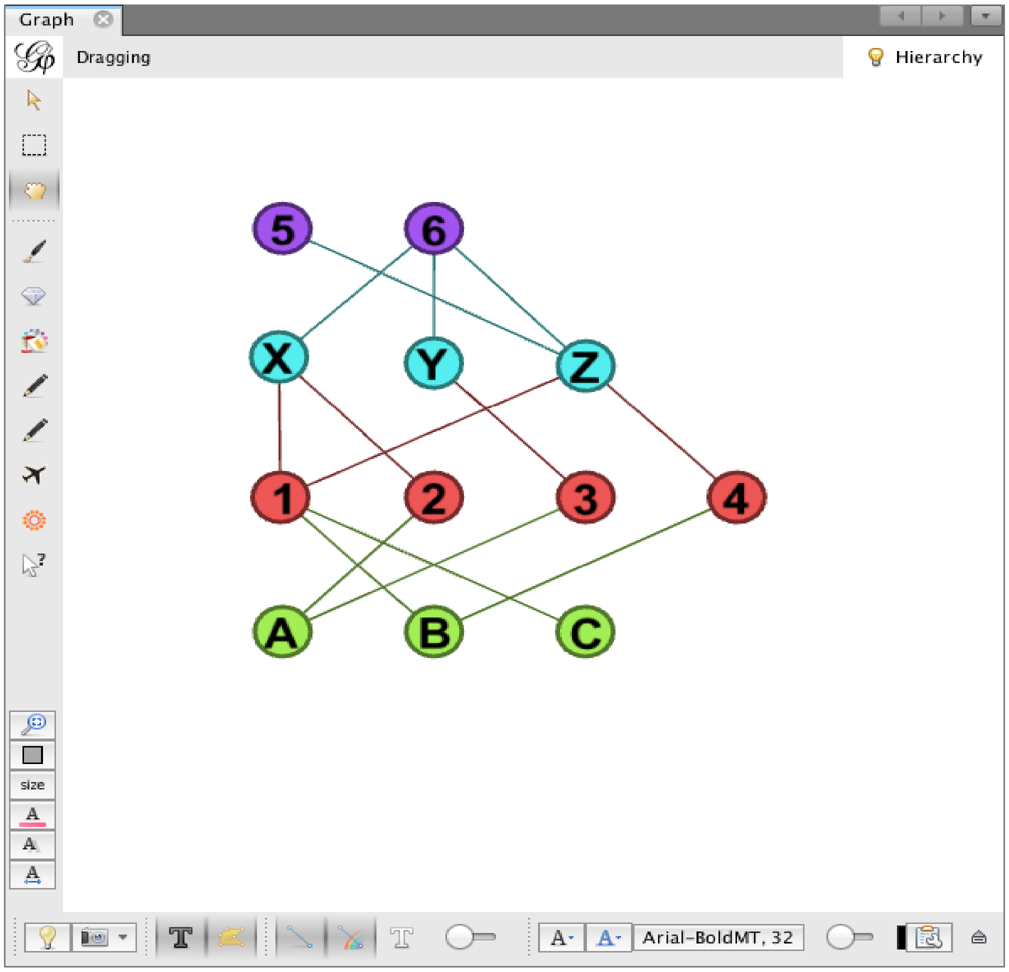


Figure 4: Ordered Graph (Before running algorithm)

As can be seen from Figure 5, by running Multipartite Layout algorithm, layers are constructed and nodes are aligned to minimize the edge crossings of the unordered graph shown in Figure 3.

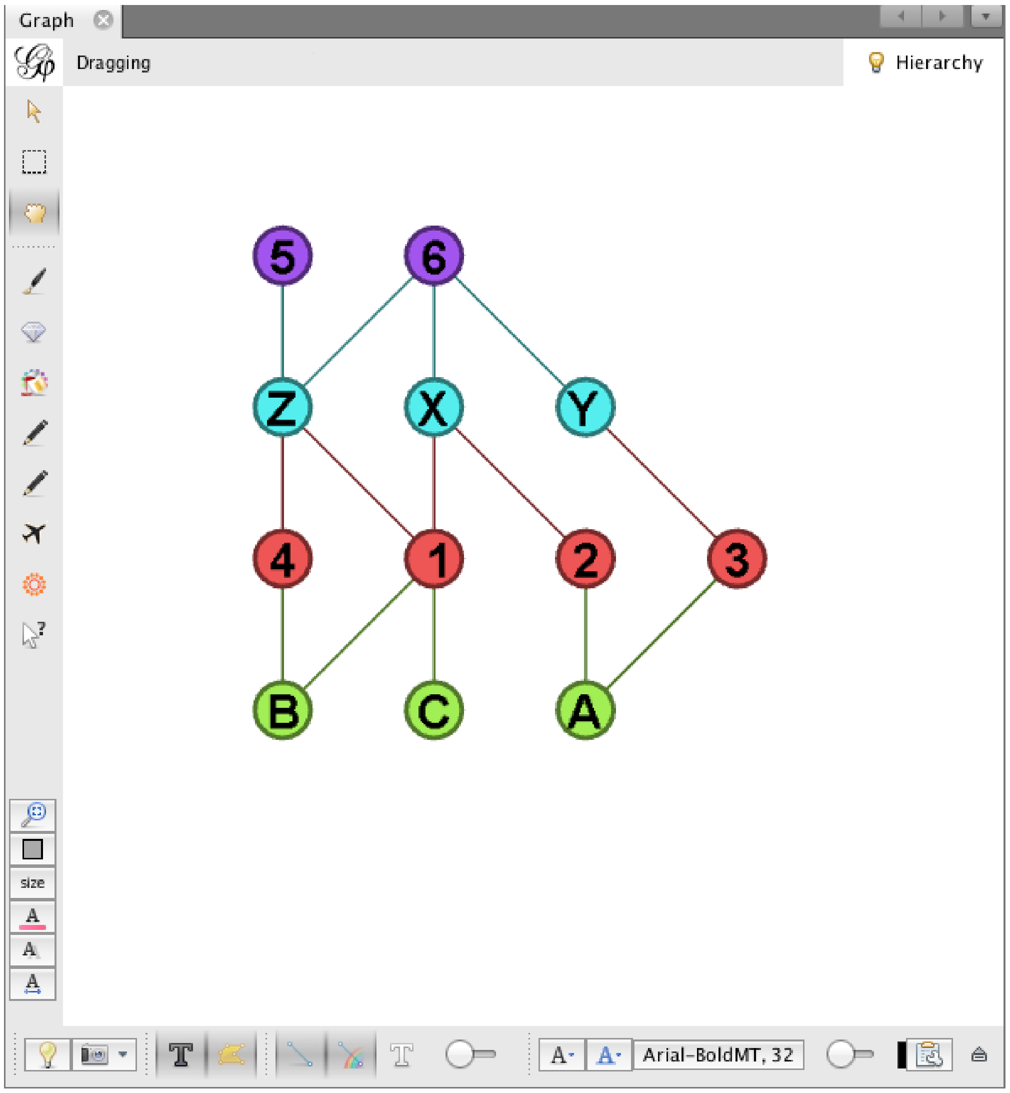


Figure 5: Graph after algorithm ends

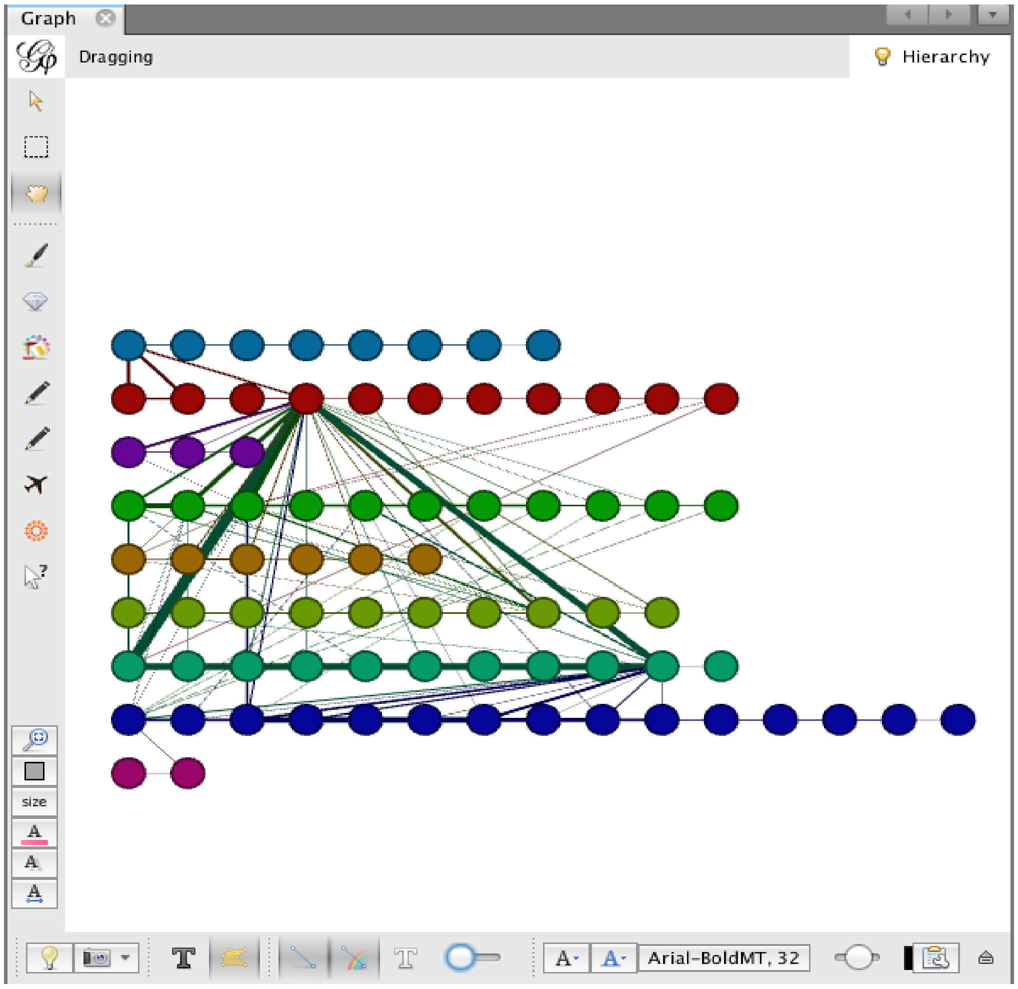


Figure 6: Les Miserables (Layer Attr: Modularity Class)

5. REFERENCES

(1) Y. kun Zhang, H. Chen, D. xin Hua, Y. an Cui, and B. wei Zhang. An edge crossing minimization algorithm based on adjacency matrix transformation. In Proceedings of the 3rd International Conference on Advanced Computer Theory and Engineering '10, volume 1 of ICACTE '10, pages V1-672-V1-675, aug. 2010.