# Assignment 3

Sangeet M
19322
AI M.Tech

## Q1: Spectral Decomposition: 1 iteration

- The first question is to implement a function that executes one iteration of the spectral decomposition problem.
- function implementation:
  - Edge matrix is the argument into the function
  - adj_mat, and L matrix, as in the equation, is calculated
  - The F vector is calculated, and the edges are split based on the sign of the eigenvalues
  - the F vector, along with the adjacency matrix and the split community lists, are returned

Plots:

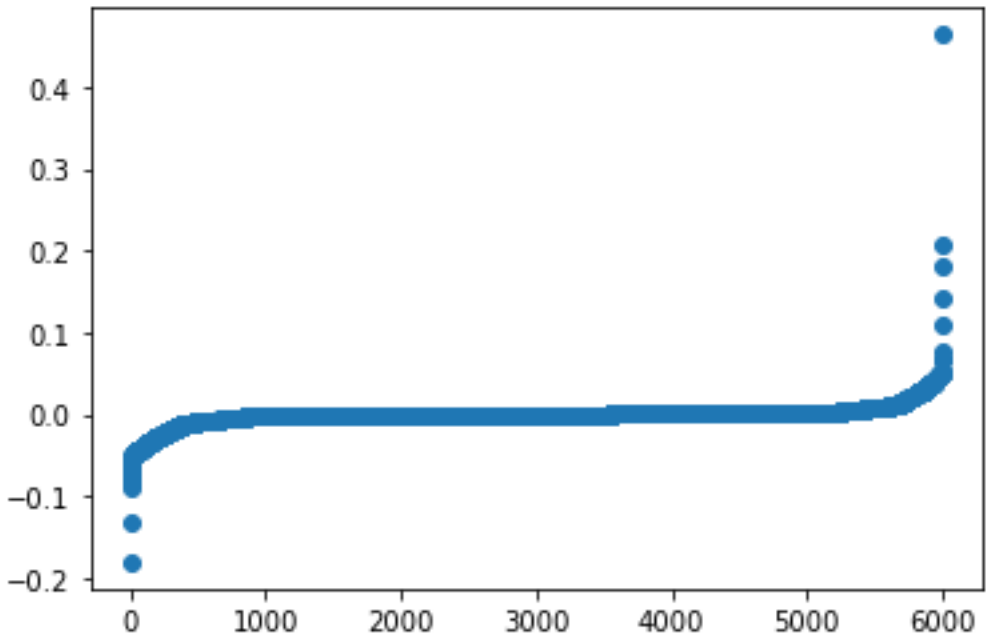


*Figure 1: Fiedler vector plot (Facebook)*

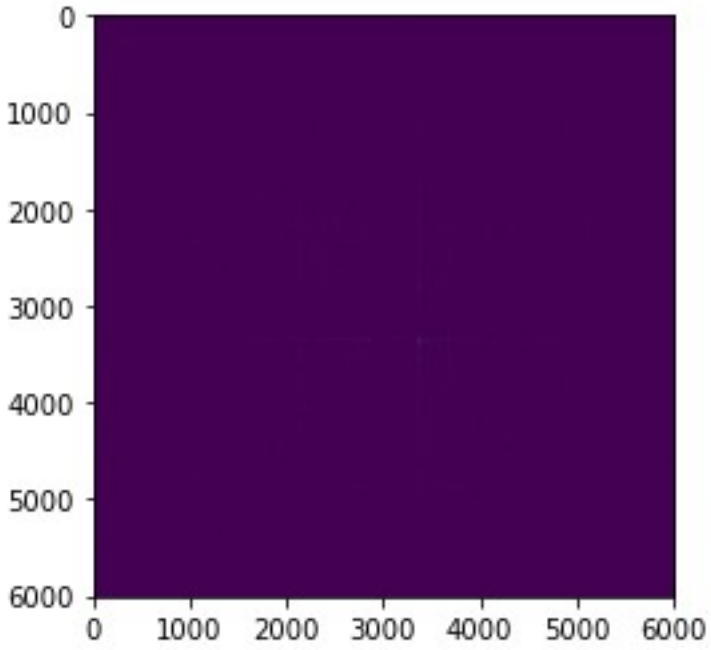


*Figure 2: Adjacency matrix after one iteration (Facebook)*

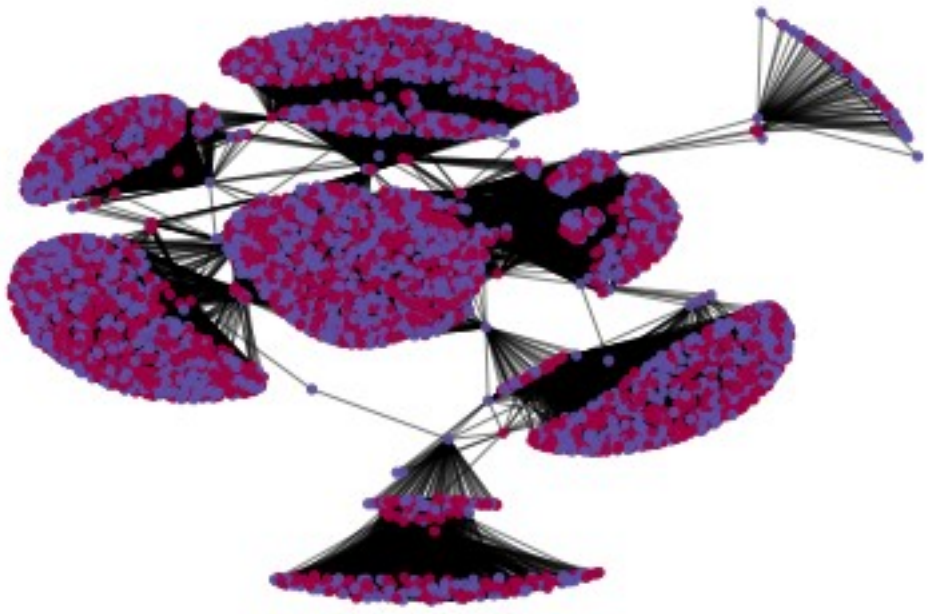

*Figure 3: Facebook: partition after first pass (Spectral decomposition)*

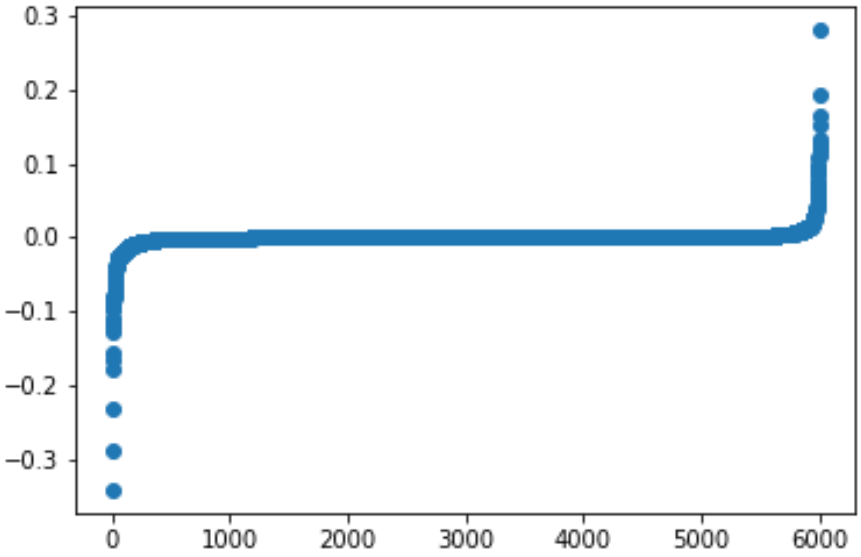


*Figure 4: Fiedler Vector(Bitcoin)*

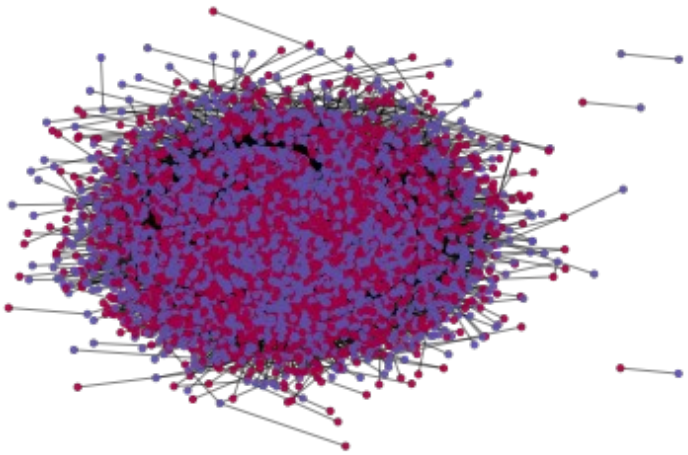

*Figure 5: Bitcoin: partition after first pass (Spectral decomposition)*

*Figure 6: Adjacency matrix after one iteration(Bitcoin)*

# Q2:Spectral Decomposition

- The spectral secomposition function returns the partitioned edge list and their partition ids
- This function uses a mapping from the original vertex ids to local ones
- This helps in the going deep into recursions
- The F-vector in this function is taken from execution the function specified in the question 1
- after the splits are done the function returns if the partitions are in one community
- In the else case the function calls itself recursively until the one of the stopping condition stated above is met

# Q3: Plots

P.S. Please select which dataset you want to load. Mentioned explicitly in lines 161 and 258

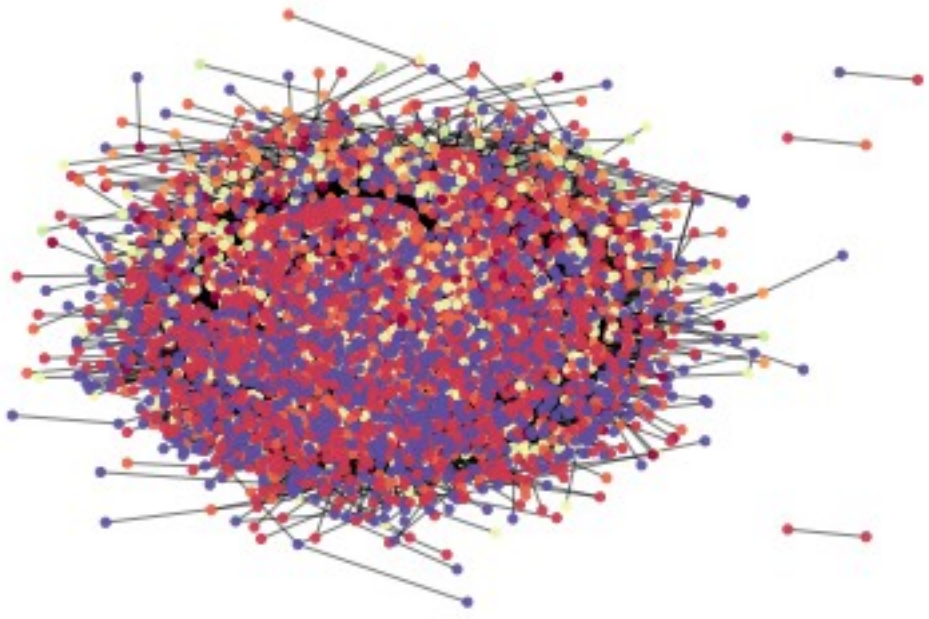

*Figure 7: Bitcoin (Spectral Decomposition)*

*Figure 8: Bitcoin: Adjacency matrix*

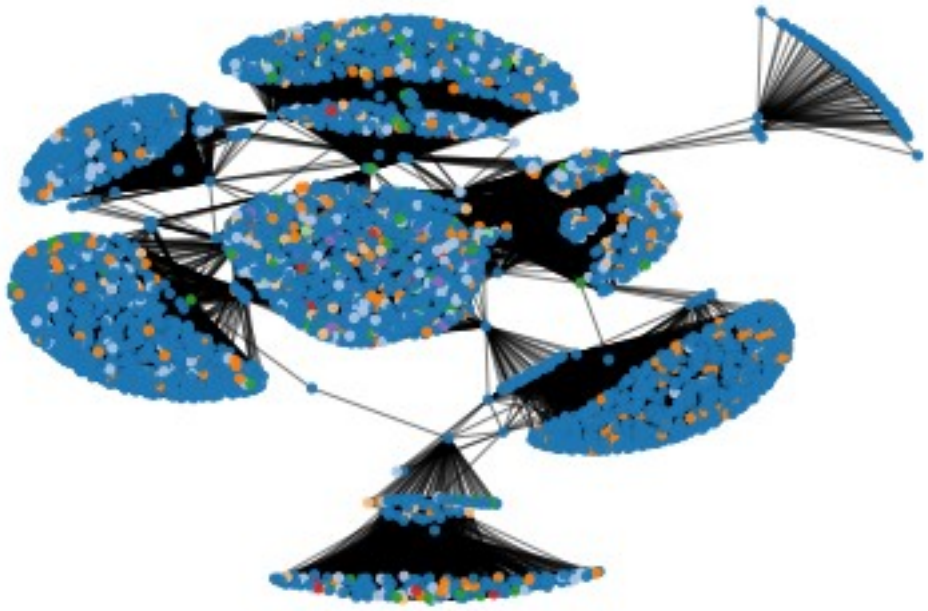

*Figure 9: Facebook (Spectral Decmoposition)*

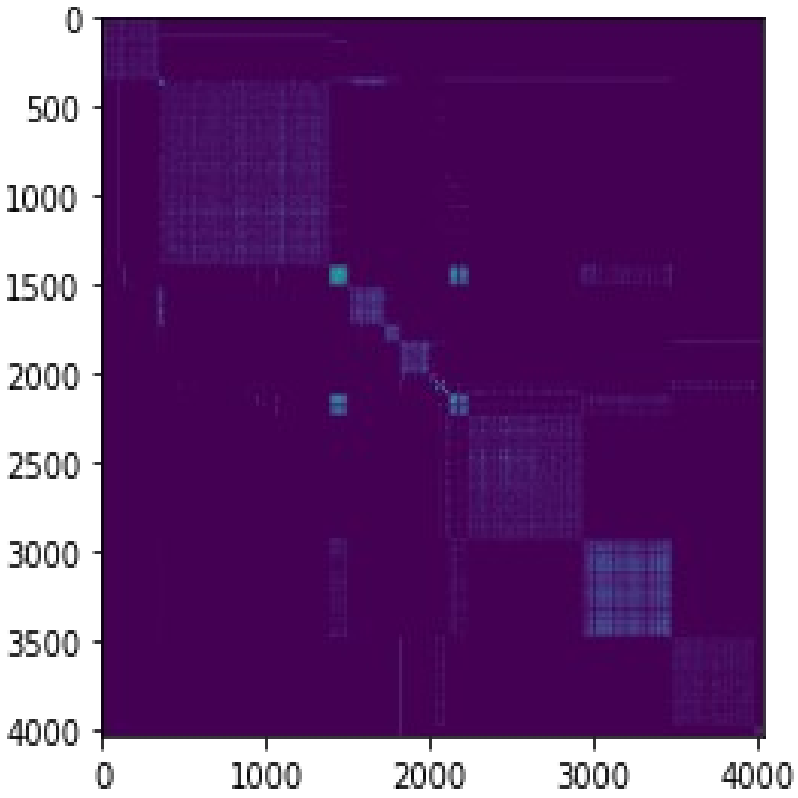


*Figure 10: Facebook: Adjacency Matrix*

# Q4:Louvain Algorithm

- Initialize:
    - adj_mat: Adjacency matrix
    - m: number of edges
- Created a data frame containing pieces of information like nearby nodes and degrees for all vertices.
- When the function iter_once is called, it does a run on all the vertices and joins the best vertex-neighbour
- The function, modularity_Comm, calculates the degree of the goodness of the graph.
- The modularity of the graph before and after the vertex transition is recorded
- This value is used to determine if a vertex shift is good or not, and the best move is made at the end of the loop
- This joining of a vertex to its best neighbors is continued until no more best neighbors exist for all nodes.

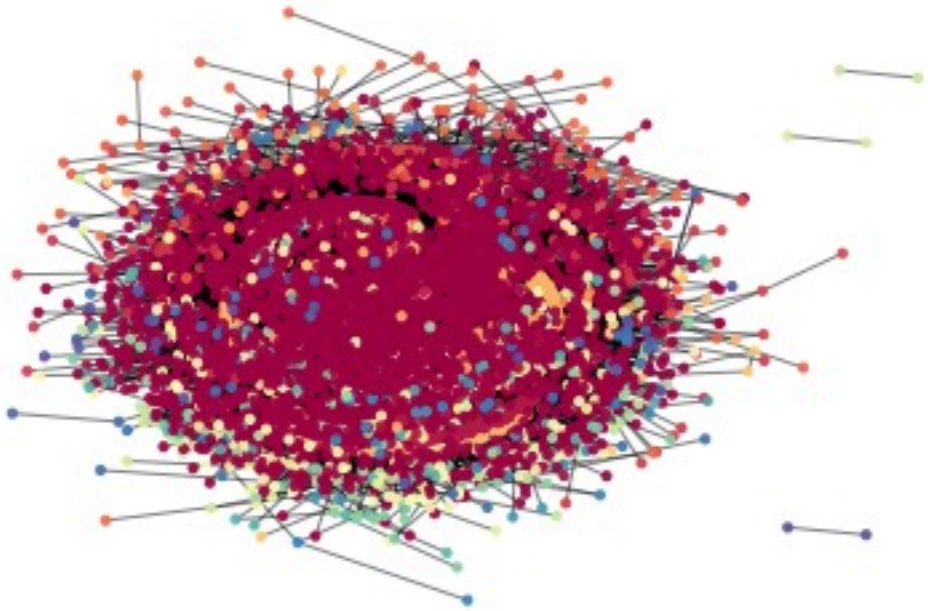

*Figure 11: Lovain plot (Bitcoin)*

*Figure 12: Louvain plot (Facebook)*

## Q5: How would you pick the best decomposition of nodes into communities?

---The best partitioning of a graph into communities is when the resulting communities bring the largest modularity to the graph. Modularity is a degree to measure the effectiveness of decomposition, and it is higher in communities containing more intra-community edges than inter-community edges.

## Q6: What was the running time of the Spectral decomposition algorithm versus the Louvain algorithm on the data sets you were given?

---Here Louvain algorithm is done in a non-online manner. The algorithm makes one transition after each complete traversal over all vertices. And because vertices can go from one community to another, this is a long way to split a graph into communities. Hence it takes 5-6 hours to run with the given dataset. When checked with graphs containing 20 to 50 vertices, it ran in less than a minute.
The spectral decomposition algorithm takes 10-15 minutes to execute, which includes the time for constructing a plot from the partitions

## Q7: In your opinion which algorithm gave rise to better communities, why?

---From the research done on both the algorithm and the program output, Louvain is the better method. Louvain method can be optimized and implemented online, which will output similar communities to the traditional way in less time, O(n.logn) to be precise.