Algorithms 2500
Second Project

Deadlines: First report 4/22, Second report 5/1

Project Demo: Schedule with TA on May 5th and 6th including the Lecture Time

Project goal: Analyze and characterize follower-followee Social Networks with respect to topological metrics

General information

Maximum group size 3.

Any programming language can be used for the implementation.

The project reports should be submitted through the blackboard by the due date.

Problem description: Social Network Analysis

Online social network sites like Facebook, Twitter, and Youtube have emerged as powerful applications to let users share contents, organize events, and communicate with friends over the Internet. An in-depth study of the graph structure of social networks is necessary to understand, evaluate, and improve current social network systems. For example, it is possible to find the influential user in social networks who you may want to follow for credible information by analyzing social network structure. It is also possible to find communities in which users are having the common interests.

The goal of this project is to investigate the social network structure in the followee-follwer social networks.

Algorithm Design

You need to design or apply algorithm(s) for computing shortest paths in directed and undirected graphs, analyzing the weighted and unweighted versions of graphs, and detecting communities. The community detection method will be described in the description of evaluation. You may directly use the algorithm(s) you've learned in the class or come up with new improved algorithms for efficiency.

Program Input

For this task you will need to download an adjacency list of vertices with weights provided by the

instructor.

First column is the source user id, second column is the target user id, and third column is the weight of the edge from source to target. You will perform experiments on directed, undirected, weighted, unweighted versions of the graph given the adjacency list. In a directed graph, you assume that there is an edge from the source to the target user. In an undirected graph, you ignore the direction. (If there are connections from A to B with weight w1 and connections from B to A with weight w2, in the undirected version, use the minimum of w1 and w2 for the weight between A and B). In an unweighted graph, you ignore the weights. And let all the weights be 1.

Evaluation

Analyze and characterize the given graph in terms of following measures.

Directed degree distribution

Create **six graphics** of the in-degree, out-degree, and total distributions for unweighted and weighted graphs when the graph is **directed**. The X-axis represents the degree and the Y-axis represents the number of vertices corresponding to the degree. **Find and output vertices** with the highest in-degree, the highest out-degree, and the highest total degree for **unweighted and weighted graphs**. If there are vertices with highest indegree /outdegree/total degree more than one, report all those vertices

Unweighted degree distribution

In-degree: the number of edges from a vertex to other vertices (the number of incoming edges)

Out-degree: the number of edges (the number of outgoing edges)

Weighted degree distribution

In-degree: the weighted sum of incoming edges

Out-degree: the weighted sum of outgoing edges

Unweighted/weighted shortest path distribution

For all-pair of vertices, compute the shortest path length between two vertices when the graph is directed and undirected. Create **four graphics** about the shortest path length distribution of **unweighted and weighted** versions of **undirected and directed graphs.** The X-axis represents the shortest path length and the Y-axis represents the number of paths corresponding to the shortest path length.

Unweighted/weighted graph diameter

Find and output the length of the longest shortest path between two vertices for **unweighted/weighted versions** of **undirected and directed graphs.**

List the **pairs of vertices with the longest shortest path** for **unweighted/weighted versions** of **undirected and directed graphs.** If there are pairs with the longest shortest path length more than one, report all those pairs of vertices

Closeness centrality

Closeness centrality of vertex i is based on the length of the average shortest path between from vertex i to all vertices, defined as follows.

$$C_c(i) = \left[\sum_{j=1}^{N} d(i,j)\right]^{-1}$$

, where d(i,j) is the shortest path length from i to j.

Create **four graphics** about the **unweighted** and **weighted** closeness centrality distribution when the graph is **undirected** and **directed**. The X-axis represents the closeness centrality value and the Y-axis represents the number of vertices corresponding to closeness centrality values. **Find and output vertices** with the highest closeness centrality values for four cases. If there are vertices with highest closeness centrality value more than one, report all those vertices

Betweenness centrality distribution

Unweighted betweenness centrality of vertex i measures the number of shortest paths that pass through i. The betweeneess centrality of vertex i is defined as follows.

$$C_B(i) = \sum g_{jk}(i) / g_{jk}$$

, where g_{jk} (i) is the number of shortest paths from vertex j to k that pass through i and g_{jk} is the total number of shortest paths from j to k.

Create **two graphics** about the unweighted betweeness centrality of vertices distribution when the graph is **undirected** and **directed**. The X-axis represents the unweighted betweenness centrality value and the Y-axis represents the number of vertices corresponding to unweighted betweenness centrality value. **Find and output vertices** with the highest unweighted betweenness centrality value for **undirected and directed graphs**. If there are vertices with highest unweighted betweenness centrality value more than one, report all those vertices

Unweighted betweenness centrality of edge e measures the number of shortest paths that pass through e. You use the same formula from above but now edge e is supposed to be the parameter instead of i. Create **two graphics** about the unweighted betweeness centrality distribution of edges when the graph is **undirected** and **directed**. The X-axis represents the unweighted betweenness centrality value and the Y-axis represents the number of edges corresponding to

unweighted betweenness centrality value. **Find and output edges** with the highest unweighted betweenness centrality value for **undirected and directed graphs**. If there are edges with highest unweighted betweenness centrality value more than one, report all those edges.

Weighted betweenness of edge e means the unweighted betweeness centrality of edge e divided by the edge weight.

Create **two graphics** about the weighted betweenness centrality distribution of edges when the graph is **undirected** and **directed**. The X-axis represents the weighted betweenness centrality value and the Y-axis represents the number of edges corresponding to weighted betweenness centrality value. **Find and output edges** with the highest weighted betweenness centrality value for **undirected and directed graphs**. If there are edges with highest weighted betweenness centrality value more than one, report all those edges.

Community Detection.

Using unweighted betweenness centrality of edge e for undirected graphs

Sort the unweighted betweenness centrality of edge e in a descending order. Repeatedly remove edges with the highest unweighted betweenness centrality value 5 times (if there are edges with the same centrality value, delete all). For each iteration, recalculate the unweighted graph diameter.

Create **two graphics** about the trend of diameter for undirected graphs. By removing the highest, you may get to have several disconnected components. Take the maximum among the diameters of disconnected components. The X-axis represents the unweighted betweenness centrality of edge that you are removing, the Y-axis represent the diameter.

[reference: https://en.wikipedia.org/wiki/Girvan–Newman_algorithm]

Optional: If you come up with other interesting results, you can add them to the report. If it is significant, you will get extra credits.

Reports

Two reports should be prepared. The first report does not require implementation and should include at least the following sections:

- Abstract: Summary of the following sections
- Introduction and motivation: Overview of social network analysis
- Proposed solutions: Pseudo-code, description and complexity analysis of the proposed algorithms

- Plan of experiments: Description of the methodology that you plan to use in the experiments, significance and expected results.
- Each team member's role in this project

The second report requires implementation and describes the implementation and results. It should include at least the following sections:

- Abstract: Summary of the following sections and results
- Implementation: Description of the methodology used for implementing the proposed solutions (e.g. relevant classes and data structures, structure of the program, etc.).
- Experiments: Description of the experiments performed and obtained results. Results should be represented in a graphical form as well as discussed in the writing.
- Each team member's role in this project
- Conclusions: Summary of the work and final considerations.

Source Code

You need to submit your source code along with 2nd report. **DO NOT** include your source code in the report.

Source code should be in the separate directory. Create .zip or .tar file including your 2nd report and source code directory.