

Assignment Social and Large-Scale Networks

5 Points Possible

10/3/2024



Attempt 1



In Progress

NEXT UP: Submit Assignment



Add Comment

Unlimited Attempts Allowed

9/16/2024 to 10/3/2024

▼ Details

Objective:

This assignment aims to enhance your practical expertise in Python programming through the application of class-learned concepts to generate graphical representations. These representations aid in identifying critical edges and nodes. The program's specific tasks include computing cluster coefficients and neighborhood overlaps within an input graph. Furthermore, the graphs will be partitioned to emphasize primary groups and store it in a new graph. Mastery of these skills will not only reinforce your programming capabilities, essential for upcoming assignments but also heighten your proficiency and competitiveness in the dynamic field of Computer Science.

Requirement

The assignment requires that you create the Python program that runs in a terminal. The program accepts optional parameters. The syntax to run the program is shown in the following line:

```
python ./graph_analysis.py graph_file.gml --components n --plot [CINIP] --verify_homophily --verify_balanced_graph --output out_graph_file.gml
```

Description of Parameters

This is the command to execute the Python script graph_analysis.py located in the current directory and reads the file graph_file.gml. Here, graph_file.gml is the file that will be used for the analysis and the format is Graph Modelling Language (.gml), which describes the graph's structure with attributes. The program should read the attributes of the nodes and edges in the file.

```
--components n
```

Specifies that the graph should be partitioned into n components. This divides the graph into n subgraphs or clusters. Compute the betweenness and remove the edge with the highest value. Repeat this process until you have n components.

```
--plot [CINIP]
```

Determines how the graph should be plotted.

C: If this option is selected, the script will plot the graph highlighting clustering coefficients (which measure how nodes are tightly grouped).

The Cluster coefficient is proportional to its size. Let cluster_min , and cluster_max be the min and max cluster coefficients, and let c_v be the clustering coefficient of node v and $p_v = (c_v - \text{cluster_min}) / (\text{cluster_max} - \text{cluster_min})$ of node v . The size of node v is proportional to p_v . Let max_pixel and min_pixel be the minimum and maximum pixel sizes of the nodes. Therefore, the node v will have size $\text{min_pixel} + p_v(\text{max_pixel} - \text{min_pixel})$.

The color of v is proportional to the degree. Similar as before, let d_v be the degree of node v , max be the maximum degree in the graph. Let $s_v = d_v / \text{max}$ be the size of node v . The RGB color of v is set to $(254 * s_v, 0, 254)$ so the node with the lowest degree is Blue and the node with the highest degree is Magenta.

N: If this option is selected, the script will plot the graph highlighting neighborhood overlap (which measures how much overlap there is between neighborhoods of adjacent nodes). Similar to the clustering coefficient.

P: If this option is selected, the script will color the node according to the attribute if it is assigned, or a default color if not.

```
--verify_homophily
```

Tests for homophily in the graph based on the assigned node colors using the Student t-test. Homophily measures whether nodes with the same color are more likely to be connected.

```
--verify_balanced_graph
```

Check if the graph is balanced based on the assigned edge signs. A balanced graph is one where the signs on the edges are consistent with the node attributes.

```
--output out_graph_file.gml
```

Specify the file to which the final graph and results should be saved. In this case, out_graph_file.gml is the output file that will receive the updated graph data—the node attributes and edge attributes should be also saved in the out_graph_file.gml.

Examples

```
python ./graph_analysis.py graph_file.gml --components 3 --plot C --output out_graph_file.gml
```

Read graph_file.gml and partition it into 3 connected components, plot the graph and highlight the clustering coefficient, and save the graph in out_graph_file.gml

```
python ./graph_analysis.py homophily.gml --plot P --verify_homophily
```

Read homophily.gml, plot the graph and verify if there is evidence of homophily in the graph

```
python ./graph_analysis.py balanced_graph.gml --plot P --verify_balanced_graph
```

Read balanced_graph.gml, plot the graph and verify if there is the graph is balanced

Evaluation:


Your implementation will be evaluated based on correctness, efficiency, clarity of code, and the quality of graph visualization. Ensure your program handles edge cases such as empty graphs, non-existent files, and invalid inputs gracefully.

Submission:

Submit your source code files along with a README.md file detailing instructions on how to use your program.

✓ View Rubric

Assignment Social and Large-Scale Networks (1)

Criteria	Ratings				Pts
Plot G and Cluster Coefficients	5 pts Exceptional Always the program plots the Cluster Coefficients correctly	3.5 pts Proficiency Sometimes the program plots the Cluster Coefficients correctly	2 pts Basic A few times the program plots the Cluster Coefficients correctly	0 pts No points The program does not plot the Cluster Coefficients correctly	/ 5 pts
Neighbor  Overlap	5 pts Exceptional Always the program plots the Neighborhood Overlaps	3.5 pts Proficiency Sometimes the program plots the Neighborhood Overlaps	2 pts Basic A few times the program plots the Neighborhood Overlaps	0 pts No points The program does not plot the Neighborhood Overlaps	/ 5 pts

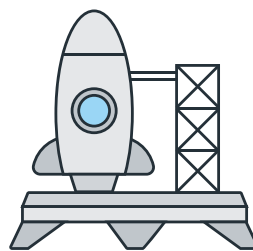
Assignment Social and Large-Scale Networks (1)

Criteria	Ratings				Pts
Partition G	5 pts Exceptional Always the program partitions G correctly	3.5 pts Proficiency Sometimes the program partitions G correctly	2 pts Basic A few times the program partitions G correctly	0 pts No points The program does not partition G correctly	/ 5 pts
Homophily	5 pts Exceptional Always the program computes the homophily test correctly	3.5 pts Proficiency Sometimes the program computes the homophily test correctly	2 pts Basic A few times the program computes the homophily test correctly	0 pts No points The program does not compute the homophily test correctly	/ 5 pts
Balanced Graph	5 pts Exceptional Always the program computes the balanced Graph	3.5 pts Proficiency Sometimes the program computes the balanced Graph	2 pts Basic A few times the program computes the balanced Graph	0 pts No points The program does not compute the balanced Graph	/ 5 pts
Follow directions	15 pts Exceptional The student follows all the directions	10 pts Proficiency The student follows some of the directions	5 pts Basic The student follows a few directions	0 pts No points The student does not follow any instructions	/ 15 pts
					Total Points: 0

Choose a submission type

Upload

More



Choose a file to upload

or



 Webcam Photo

 Canvas Files



<https://csulb.instructure.com/courses/78118/modules/items/4980632>



<https://csulb.instructure.com/courses/78118/modules/items/4976>

