When working with network data in Python and wanting to add functionality similar to Autologistic Actor Attribute Models (ALAAMs), we must consider different types of statistical measures and analyses that can be meaningful in the context of networks. Although direct implementations of ALAAM are not readily available in Python libraries like NetworkX or igraph, you can implement similar concepts by focusing on node attributes, their relationships, and network structure.

Here are three advanced network analysis functions you can implement using Python and NetworkX, which align with the spirit of ALAAM by considering node attributes and their influence on network structure:

**1. Node Attribute Assortativity**

Node attribute assortativity measures the similarity of connections in the network with respect to a given attribute. This can be a proxy to how homophilous the network is regarding that attribute (i.e., the tendency of nodes to be connected to others with similar attributes).

**2. Clustering Coefficient by Node Attribute**

The clustering coefficient measures the degree to which nodes in a graph tend to cluster together. Calculating this metric by grouping nodes based on a specific attribute can help understand the tight-knit communities within the network that share common attributes.

**3. Influence Spread using Influence Maximization**

This involves identifying the most influential nodes in a network, given a model of influence propagation. This is akin to simulating a process where attributes (like information or disease) spread through the network, and can be particularly useful in social network analysis for marketing or epidemiological modeling.

Here's how you can extend the function to include these analyses and visualize the results:

python

import pandas as pd

import networkx as nx

import matplotlib.pyplot as plt

def advanced\_network\_analysis(csv\_path, attribute):

# Load data and assume that `attribute` is a column in CSV

data = pd.read\_csv(csv\_path)

G = nx.from\_pandas\_edgelist(data, 'source', 'target')

# Assume node attributes are provided in another column and add them to nodes

attr\_dict = data.set\_index('source')[attribute].to\_dict()

nx.set\_node\_attributes(G, attr\_dict, 'attribute')

# 1. Node Attribute Assortativity

assortativity = nx.attribute\_assortativity\_coefficient(G, 'attribute')

# 2. Clustering Coefficient by Node Attribute

clustering = nx.clustering(G)

# We average clustering coefficients of nodes having the same attribute value

attr\_clustering = {}

for node, cluster\_value in clustering.items():

attr\_value = G.nodes[node]['attribute']

if attr\_value in attr\_clustering:

attr\_clustering[attr\_value].append(cluster\_value)

else:

attr\_clustering[attr\_value] = [cluster\_value]

attr\_clustering\_avg = {k: sum(v) / len(v) for k, v in attr\_clustering.items()}

# 3. Influence Maximization (using degree centrality here as a proxy)

degrees = nx.degree\_centrality(G)

top\_influential = sorted(degrees.items(), key=lambda x: x[1], reverse=True)[:5] # top 5 influential nodes

# Save results to a text file

summary\_path = 'advanced\_network\_analysis.txt'

with open(summary\_path, 'w') as file:

file.write(f"Node Attribute Assortativity: {assortativity}\n")

file.write(f"Clustering Coefficients by Attribute: {attr\_clustering\_avg}\n")

file.write(f"Top 5 Influential Nodes: {top\_influential}\n")

# Plotting for visualization

plt.figure(figsize=(10, 8))

pos = nx.spring\_layout(G)

nx.draw(G, pos, with\_labels=True, node\_color=list(degrees.values()), cmap=plt.cm.viridis,

node\_size=[v \* 5000 for v in degrees.values()], edge\_color='gray')

plt.savefig('network\_visualization.png')

plt.show()

return summary\_path, 'network\_visualization.png'

# Example usage

try:

summary\_file\_path, graph\_image\_path = advanced\_network\_analysis('your\_data.csv', 'attribute\_name')

print(f"Summary file saved at: {summary\_file\_path}")

print(f"Graph image saved at: {graph\_image\_path}")

except Exception as e:

print(f"An error occurred: {e}")

**Notes:**

* **Attribute Column**: Replace 'attribute\_name' with the actual name of the column in your CSV that contains the attribute you want to analyze.
* **CSV Format**: Ensure your CSV includes columns for source, target, and the attribute of interest.
* **Visualization**: The graph visualization uses node sizes based on their degree centrality and colors based on the same metric.

This extended functionality provides deeper insights into the network structure and attribute dynamics, akin to what you might explore with ALAAMs in statistical packages. If you need further customization or have additional data dimensions to consider, let me know, and we can refine the approach further!