University of Essex

Department of Mathematics, Statistics and Actuarial Science

MA214: Network Analysis

Group Assignment

Analysis of London Multiplex Transport Network

Group Name: Optimal Networkers

Registration No: 2310158

Team members

Student ID	Student name	Email id
2309719	Sharath Babu Gadhala Venkataraman	sg22958@essex.ac.uk
2310158	Ashwini Sailappan	as23772@essex.ac.uk
2310825	Shalini Gulshan Sharma	ss23794@essex.ac.uk
2310982	Shaik Fazil Feroze	sf23686@essex.ac.uk
2312420	Prawin Thiyagarajan Veeramani	pt23757@essex.ac.uk

TABLE OF CONTENTS

1 ABSTRACT:

Assessing the efficiency and impact of London's railway network is crucial for understanding its role in Great Britain's economic, social, and financial spheres. This study utilizes data from Transport for London's 2013 records, encompassing DLR, Overground, and Tube categories, to conduct a thorough network analysis. The findings reveal a network characterized by optimization of coverage over redundancy, potentially affecting its resilience to disruptions. Degree Centrality distribution highlights varied connections among nodes, emphasizing the need for congestion management at crucial hubs. Additionally, analysis of Betweenness Centrality identifies pivotal nodes influencing network flow, guiding strategic planning efforts. Furthermore, the study examines the network's modularity, clustering coefficient, and algorithms to identify community structures and potential enhancements. Strategic recommendations focus on enhancing core resilience, optimizing design for efficiency and redundancy, and leveraging localized connectivity to enhance operational resilience. In conclusion, the London transport network demonstrates a balanced design aimed at optimizing coverage, efficiency, and resilience. Insights from this analysis provide a roadmap for targeted enhancements to ensure operational reliability, robustness, and service quality, effectively addressing present and future challenges.

2 INTRODUCTION:

Network analysis of London's rail transportation is vital for assessing efficiency and impact on Great Britain's factors. Economic, social, and financial aspects thrive with a robust transportation network. Data from Transport for London's website, collected in 2013, includes DLR, Overground, and Tube categories. Nodes represent London train stations, and edges signify existing routes. The multiplex network incorporates three layers: aggregated weighted graphs of underground lines, Overground connections, and DLR stations. Raw data and station coordinates are available, along with multiplex networks reflecting real disruptions. Visit https://www.tfl.gov.uk/ for details.

Layer ID	Layer Name	Layer Colour	Count of Nodes
0	DLR	Blue	37
1	Overground	Red	99
2	Tube	Yellow	343
Total			**479**

Ref: Manlio De Domenico, Albert Solé-Ribalta, Sergio Gómez, and Alex Arenas, Navigability of interconnected networks under random failures. PNAS 111, 8351-8356 (2014)

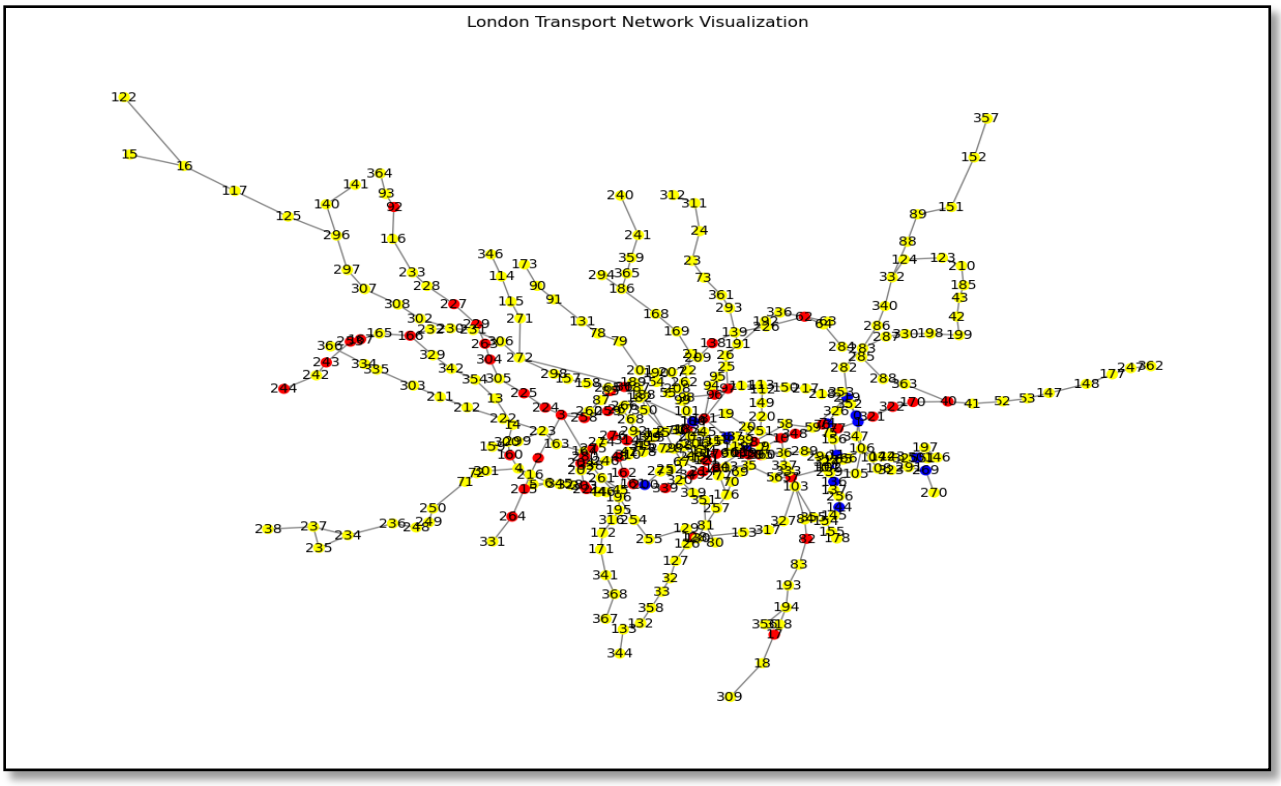

Fig 2.1 Representing 479 nodes categorized as three layers DLR, Overground and Tube

2.1 ANALYSIS:

We have found the below insights and literature documents related to the London rail transportation network analysis.

2.2 LITERATURE REVIEW:

The London train network is critical to the city's economy, transporting an estimated 1.1 million people daily. Transport for London (TfL) operates the London Underground, Docklands Light Railway (DLR), and Tramlink. The Tube, which spans 402 kilometres and operates 11 lines, handles an amazing 5 million journeys per day. To ensure its seamless operation, a complete network analysis and disruption free networking is required. This includes reviewing its performance, capacity, and areas for improvement. By analysing the network', we would like to inference on the potential efficiency increase, reduce congestion, and meet the evolving demands of London people.

A study demonstrates the resilience of rail network is influenced by the speed of backup restoration failure in network vulnerability during peak hours increases 63% of increase in passenger delays compared to off- peak hours. While studies have made substantive contributions to research

knowledge of the resilience of rail networks, they are limited in their approach to analysing the dynamic knock-on effect of train disruptions on timetabling, train and passenger delays and how these propagate across the interdependent network. Also, these existing studies are limited in their approach to the study of the impact of system-of-systems interdependencies on the resilience of rail networks and their practical application to real-word networks, with a dearth in research literature of robust methodologies and models for resilience assessment of large, complex rail networks at regional and national scale. *Ilalokhoin et al. (2023)*

A study on causes of disruptions is associated with weather condition mainly. Great Britain shows that the train operators apparently have been better able to manage delays with the extreme network conditions. John Preston, Graham Wall, Richard Batley, J. Nicolás Ibáñez,and Jeremy Shires(2019). Also,an analysis shows that assumptions on train path and delays are not accounted for multi train tracks. The model enhance risk analysis for large rail networks like in Great Britain while prioritising the disruptions enhance network resilience. *Ohis Ilalokhoin, Raghav Pant & Jim W. Hall (2022)*

The reliability of London rail transportation is high, as they are monocentric networks. In general, the London rail network boasts peripheral hub connections, which contribute to its robustness in comparison to the Shanghai and Randstad rail networks. Furthermore, the connectivity of radial networks is highly susceptible to disruption, resulting in the isolation of a single branch from the remaining network. The research study reveals that the correlation between network structure and its resilience is non-trivial *(Cats & Krishnakumari, 2020)*

The London rail transportation networks conduct a thorough assessment of assets based on their failure probability, failure consequences, and associated expenses and inspections. This model introduces a multilayer network infrastructure, simulation of failure propagation dynamics, incorporating network skeletonization to reduce computational complexity. The methodology is adaptable to other infrastructure networks and facilitates the evaluation of mitigation measures. The limitations encompass presumptions regarding the synchronization of failure timescales and rail passenger behavior, which can be addressed in future research, along with the enhancement of model detail and the incorporation of asset recovery dynamics..*Wee et al. (2023)*

A study on the Density and dispersion on London rail network exhibits the relationship between the diverse density population and the dispersed London rail structures as underground and overground shaping the cities in a better way. The network structure helps to reduce the crowd towards the central London, dispersing equally across to maintain the minimal disruption between the passengers. The research also generalises on the invent of new modes developed in the London networks. *Levinson (2008)*

A graph-based approach to model cascading delays in the British railway network, acknowledging the intricate nonlinear interactions inherent in such disruptions. Utilizing the Spatial-Temporal Graph Convolutional Network (STGCN) architecture, predicting delays throughout the network, outperforming traditional statistical models. This method captures both spatial and temporal dimensions, enhancing accuracy in predicting delays for trains traversing each link. Future directions include extensive comparisons with existing models, refining problem formulations to

account for specific routes and inbound/outbound traffic, and deeper exploration of delay causes and propagation for real-world deployment of our models. *Heglund et al. (n.d.)*

A recent study provides valuable insight into how to mitigate future heat-related risks and improve the resilience of rail infrastructure to extreme weather events. It addresses a crucial matter pertaining to the vulnerability of rail infrastructure, specifically the London Underground (LU), to extreme weather events, with a particular emphasis on the impact of heat. The study's recommendations encompass the expansion of the analysis to the LU asset scale, incorporating the local environment to comprehend failure causality, and scrutinizing delay capture methodologies to facilitate climate resilience benchmarking among infrastructure networks. *Greenham et al., 2020*

A wide range of research are undertaken in accordance to minimise the disruptions with London rail transportation network. Though the existing rail network seems to be robust when compared to Paris and other European countries, there are significant areas of improvement in handling the disruptions in terms of improvement in association with economic, social and financial and population density aspects.

2.3 ANALYSIS

2.3.1 Network Density

Network density measures at 0.006333215506068104, indicating a sparse network common in large-scale transportation systems, optimizing coverage over redundancy. This may affect resilience to disruptions but also suggests efficient design.

2.3.2 Degree Centrality

The Degree Centrality distribution in the London transport network showcases varied connections, with outliers indicating crucial hubs that may require congestion management. The histogram highlights how degree centrality values distribute across nodes, skewing towards lower values typical in real-world networks like transportation systems. This indicates most stations have few direct connections. Despite lower heights, hubs signify key stations, enhancing network efficiency. The overall shape reveals a decentralized structure, with critical hubs ensuring network cohesion and efficient movement. Enhancing these hubs' capacity could significantly boost network performance, while their vulnerability underscores the need for contingency planning. Understanding centrality distribution informs planning efforts, aiming to bolster connectivity for less central nodes and optimize flow through key hubs.

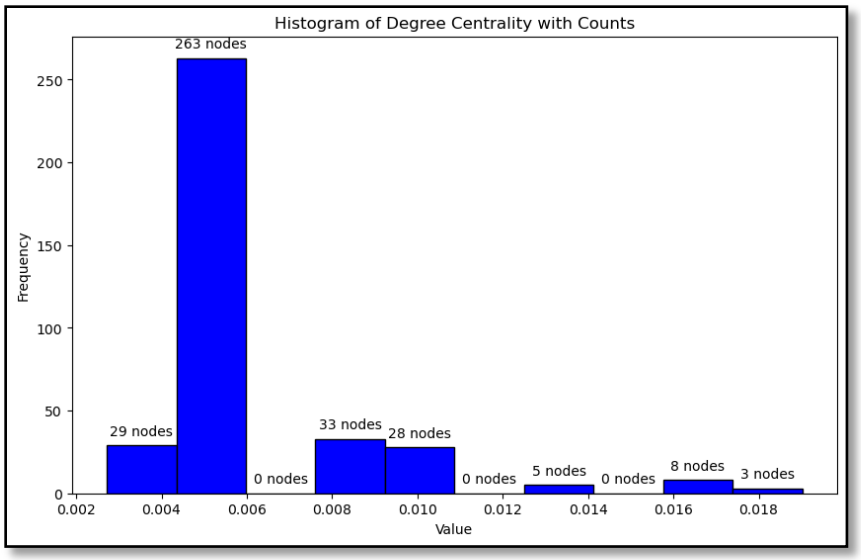

Fig 2.3.2.1 visual representation the number of nodes with degree centrality values distributed across the nodes in the London transport network.

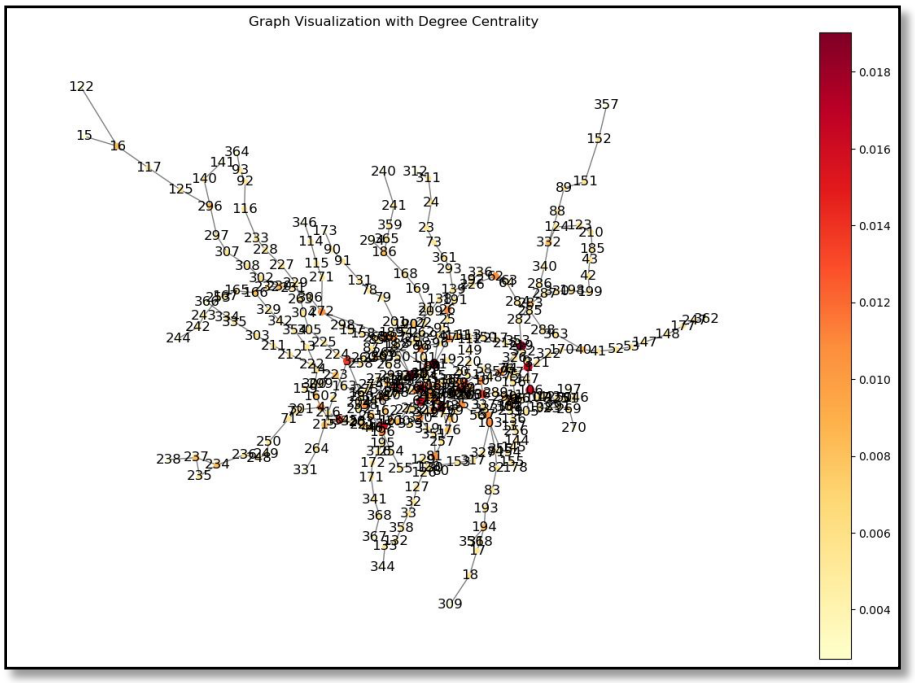

Fig 2.3.2.2 Network visualization - degree centrality on London rail transportation

2.3.3 Betweenness Centrality

The histogram for betweenness centrality likely displays a right-skewed distribution, typical in networks, indicating most stations have low betweenness centrality, lying on fewer shortest paths. Despite relatively low bars on the right, few stations act as significant connectors, facilitating major flow between others. These critical nodes influence network flow, representing potential bottleneck points affecting efficiency and resilience. Stations with high betweenness centrality are pivotal for connectivity; disruptions here cause delays. Enhancing infrastructure at these stations disproportionately improves network functionality. Analysis of betweenness centrality guides strategic planning, highlighting key nodes for improvement to enhance performance and resilience against disruptions, providing crucial insights into the London transport network's flow dynamics.

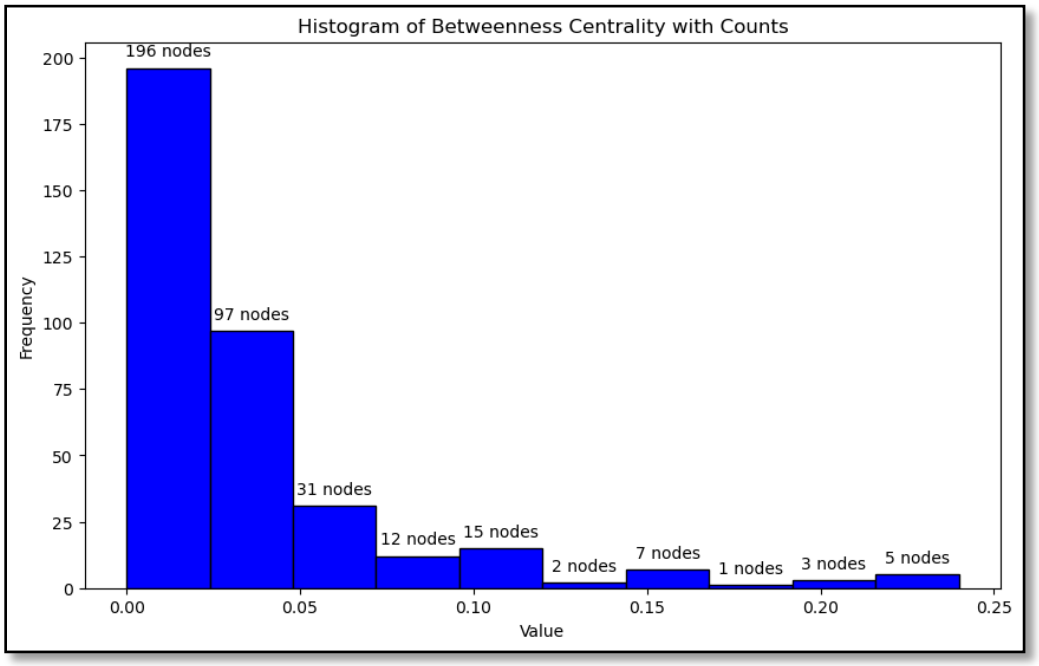

Fig 2.3.3.1 Histogram of Betweenness Centrality illustrates how betweenness centrality values are distributed among the nodes within the London transport network.

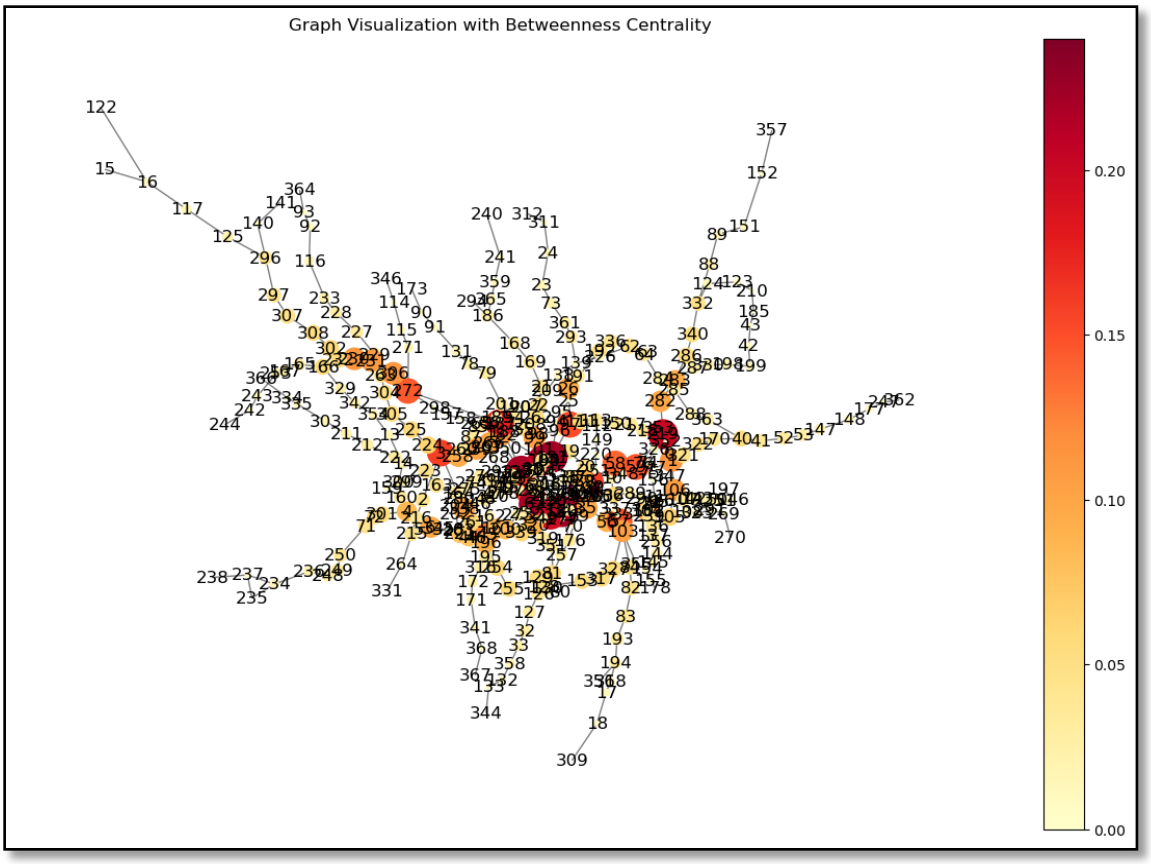

Fig 2.3.3.2 Network visualization - betweenness centrality on London rail transportation

2.3.4 Closeness Centrality

Variability across ranges indicates certain stations are more centrally connected. Stations with high closeness centrality are pivotal, ensuring network accessibility and efficiency by facilitating shorter travel times. Enhancing services at these strategic points can significantly improve network performance. Conversely, stations with lower closeness centrality may need improvement to enhance overall accessibility. Analyzing closeness centrality aids in identifying accessible and isolated nodes, guiding planning efforts to improve connectivity and reduce travel times. This perspective offers insights into the network's cohesion and accessibility, guiding targeted improvements for a more user-friendly London transport network.

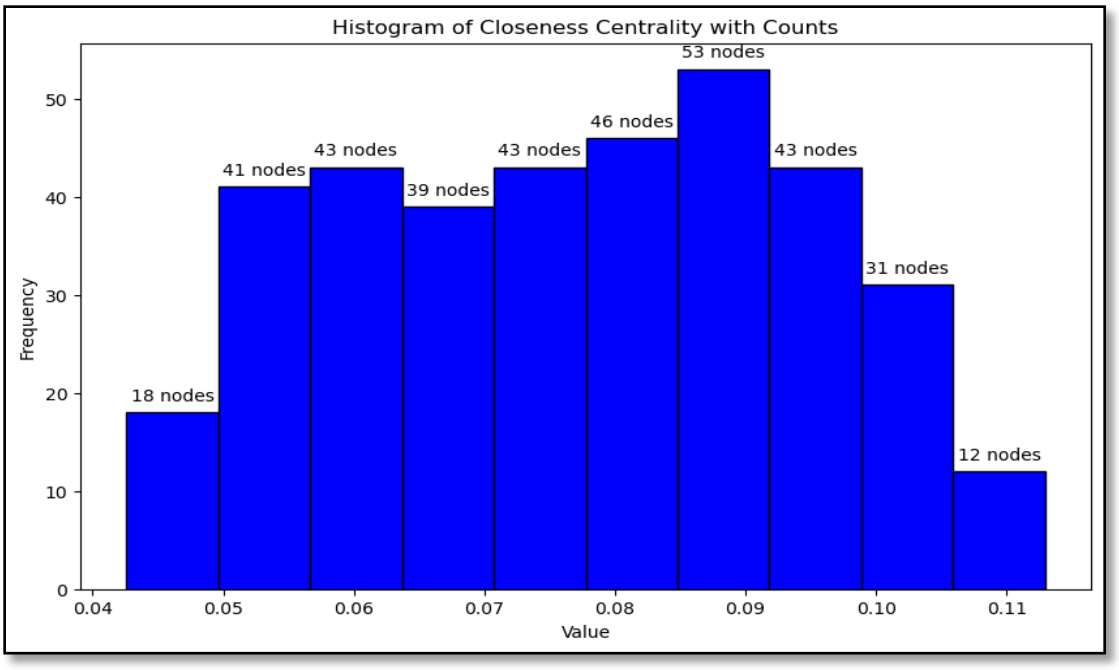

Fig 2.3.4.1 concentration of closeness centrality values within a specific range, suggesting uniformity in station connections.

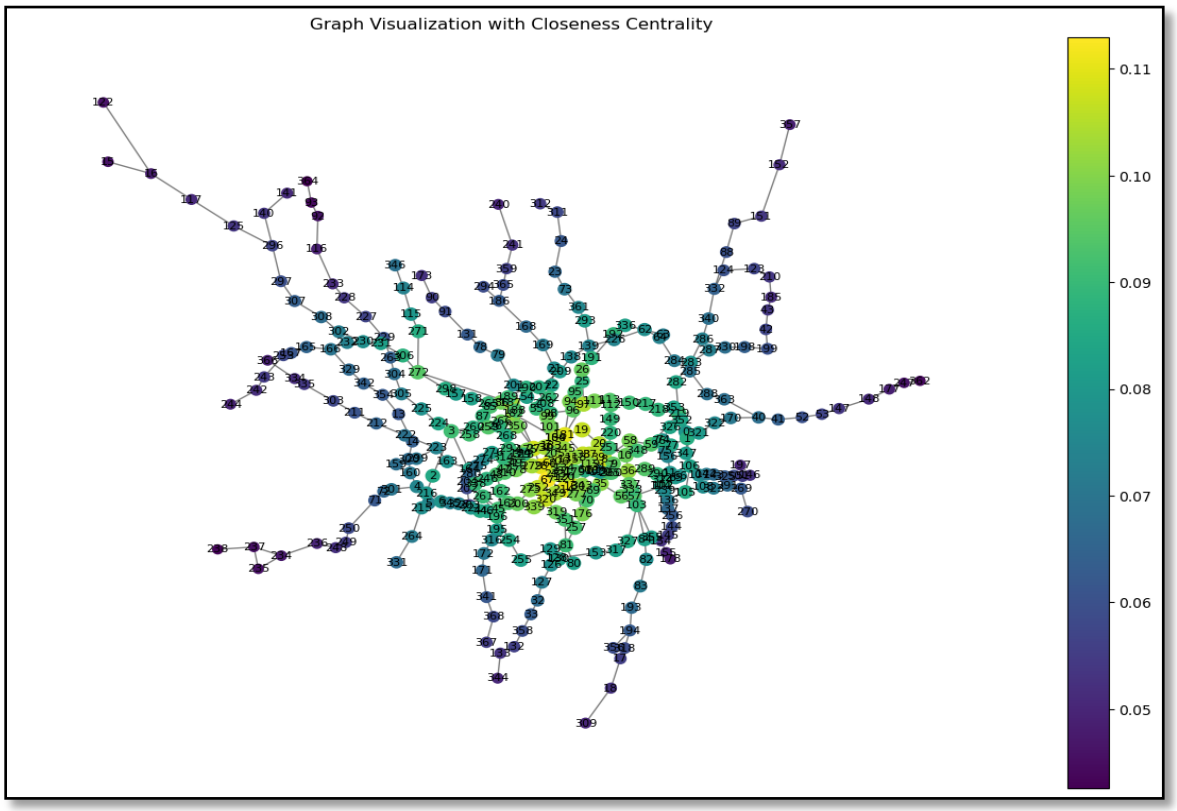

Fig 2.3.4.2 Network visualization - closeness centrality on London rail transportation

2.3.5 K Core

K-Core Nodes count of 221 offers insight into the London transport network's structure. A k-core is a maximal subgraph where each node has at least k connections to others within it, revealing cohesive subgroups and network resilience. With 221 nodes, this k-core represents a highly connected and cohesive subgraph, crucial for network integrity. These nodes likely include major stations or junctions, vital for passenger movement and network flow. While advantageous for resilience, high connectivity may lead to congestion, necessitating capacity enhancements. Targeted improvements within this core can significantly boost network efficiency and reliability. Understanding the k-core aids emergency planning, ensuring strategies to mitigate disruptions. Enhancing connectivity to and from this core enhances network accessibility. Identifying 221 k-core nodes emphasizes their importance in maintaining network cohesion, resilience, and efficiency within the London transport network.

2.3.6 Connected Components

The London transport network's single connected component indicates complete interconnectivity, allowing travel between any stations without leaving the network, ensuring comprehensive access for passengers. However, this unity poses vulnerability to disruptions if critical nodes fail, lacking isolated components or redundancies. While advantageous for simplifying route planning and resource deployment, it requires strategic improvements to enhance redundancy, reduce travel times, and alleviate congestion. Increasing redundancy in critical areas and focusing on enhancing efficiency at key nodes are crucial for resilience and smooth network operation. Addressing areas with lower connectivity within the single component can improve inclusivity and utility for all users. Overall, recognizing the network's unity lays the groundwork for strategic planning to optimize efficiency, resilience, and accessibility within the London transport system.

2.3.7 Cliques

Num of Cliques: 405

Max Clique Size: 3

The London transport network comprises 405 cliques, each with a maximum size of 3, providing crucial insights into its structure. These cliques, where every pair is directly connected, highlight efficiency over extensive direct connections. Strategic improvements at these transfer hubs can enhance network performance, emphasizing efficient pathways through key points rather than creating more direct connections.

2.3.8 Clustering Co-efficient

The clustering coefficient of around 0.029 in London's transport network reveals that neighboring stations often share connections, indicating a lack of tight interconnection into clusters. This design prioritizes system-wide efficient travel over dense clusters, aligning with urban transport needs. However, optimizing opportunities exist, particularly in underserved areas, where increased local clustering could enhance resilience and accessibility. Strategic enhancements, such as additional routes or services, can improve local connectivity and passenger experience, contributing to network resilience. Understanding the clustering coefficient aids in identifying areas for strategic enhancements, enhancing the network's functionality, resilience, and accessibility.

2.3.9 Modularity

The network's modularity reveals its community structure, identified by a specific algorithm. High modularity signifies strong community divisions, with dense intra-community connections. The data highlights two distinct communities within the transport network, indicating separate regions where stations are densely interconnected. These communities likely reflect geographical regions or functional divisions like commuter zones versus business districts. Understanding these communities aids targeted planning and optimization efforts, allowing for tailored improvements and service adjustments to meet specific regional needs. Inter-community connections are vital for overall network functionality, facilitating efficient travel across regions. Tailoring services within each community and directing infrastructure investments strategically can enhance connectivity and capacity. Emergency planning considers community disruptions' ripple effects on the broader network, ensuring alternative routes and services to maintain functionality. Identifying communities offers insight into internal connectivity and critical interconnections, guiding strategic planning to optimize the London transport system's resilience and functionality for diverse user needs.

2.3.10 Louvain method – to optimize the modularity

Number of communities detected: 19

Community 1: [2, 3, 92, 93, 116, 215, 216, 224, 225, 227, 228, 229, 233, 263, 264, 304, 305, 331, 364]

Community 2: [4, 5, 6, 71, 72, 221, 234, 235, 236, 237, 238, 248, 249, 250, 301, 328, 345]

Community 3: [7, 8, 9, 19, 20, 30, 37, 38, 39, 60, 61, 109, 110, 179, 181, 184, 295]

Community 5: [0, 1, 10, 11, 12, 36, 58, 59, 65, 66, 74, 75, 76, 77, 106, 108, 111, 112, 113, 149, 150, 156, 217, 218, 219, 220, 251, 269, 270, 289, 290, 291, 323, 324, 326, 347, 348, 352, 353, 360]

Community 7: [13, 14, 15, 16, 117, 122, 125, 140, 141, 165, 166, 167, 230, 232, 242, 243, 244, 253, 296, 297, 300, 302, 307, 308, 329, 342, 354]

Community 8: [17, 18, 82, 83, 84, 103, 128, 129, 153, 193, 194, 254, 255, 309, 317, 318, 327, 355, 356]

Community 9: [21, 22, 168, 169, 186, 240, 241, 262, 294, 359, 365]

The Louvain method identifies 17 communities in London's transport network, revealing its intricate modular structure. This method efficiently uncovers densely interconnected station groups, suggesting geographical or functional correspondence. Variations in community sizes indicate differences in network density, guiding optimization efforts. Understanding each community's dynamics allows tailored strategies to enhance network efficiency and resilience. Recognizing community structures aids emergency planning and improves overall network satisfaction. Leveraging Louvain insights facilitates strategic planning and infrastructure development, enhancing service quality to meet diverse user needs.

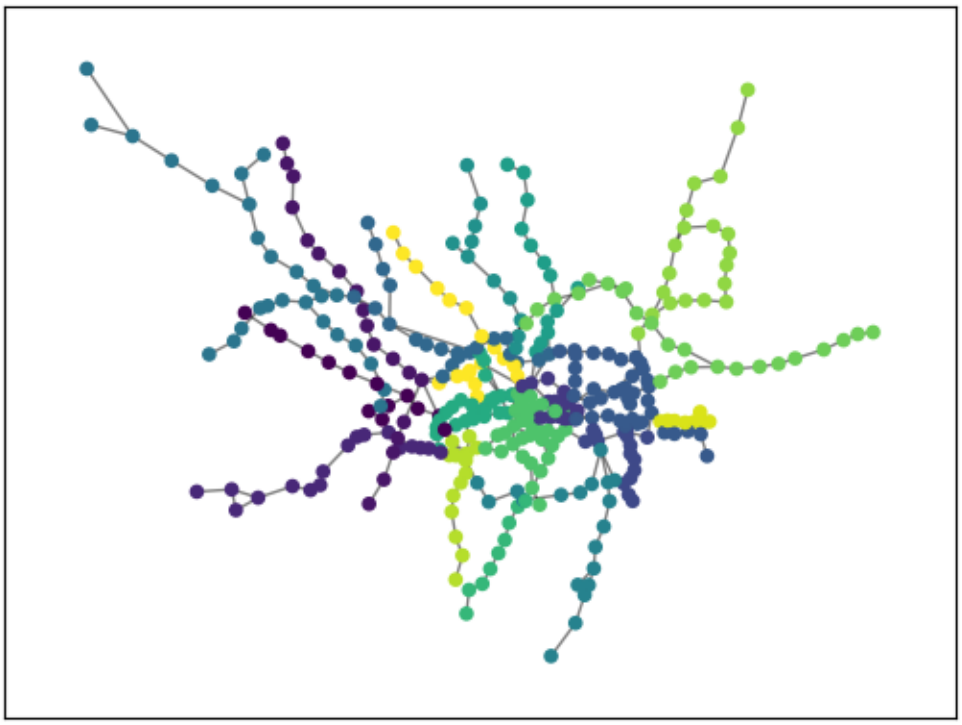

Fig 2.3.10.1 Network visualization of community generated through Louvain Method

2.3.11 Girvan-Newman algorithm – to identify the communities

The Girvan-Newman algorithm in network science identifies communities by iteratively removing edges with high betweenness centrality, dividing the graph into denser communities guided by the modularity score. Detecting 17 communities within London's transport network suggests significant specialization, potentially representing geographical areas or transport lines. High modularity scores reflect strong community structure, guiding optimization strategies for enhanced connectivity and infrastructure investments. Resilience planning considers community detection

to mitigate disruptions, ensuring alternative routes for network functionality. Leveraging Girvan-Newman insights enhances network efficiency and resilience in London's transport.

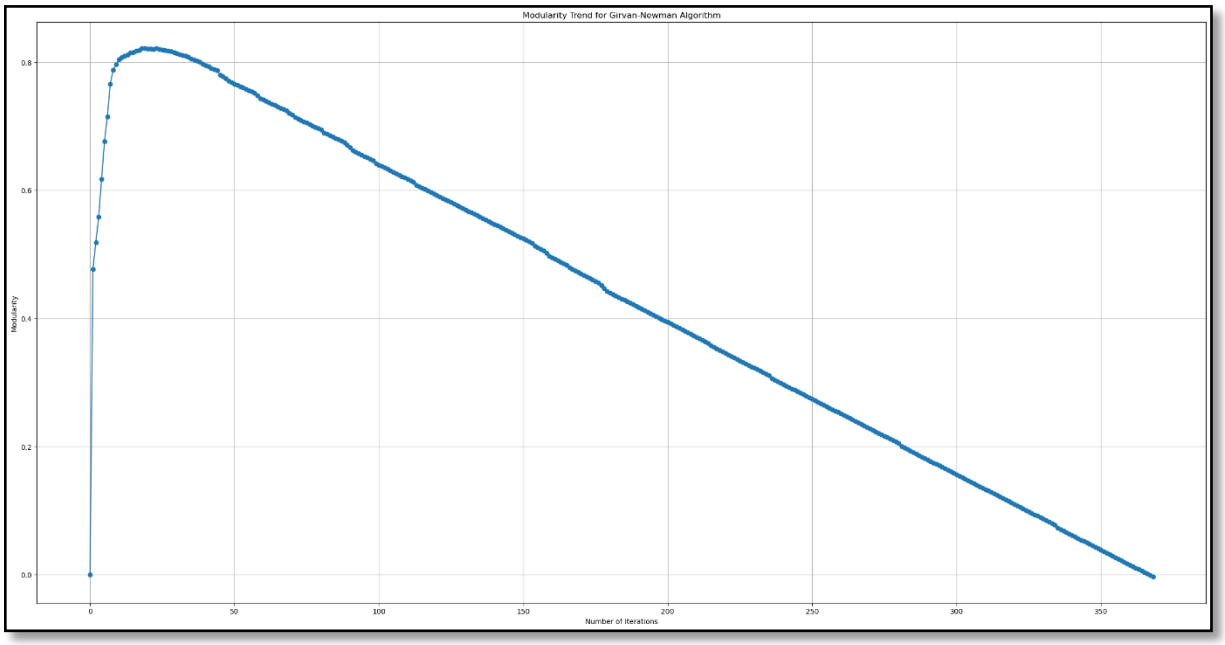

Fig 2.3.11.1 Modularity Trend based on Iterations

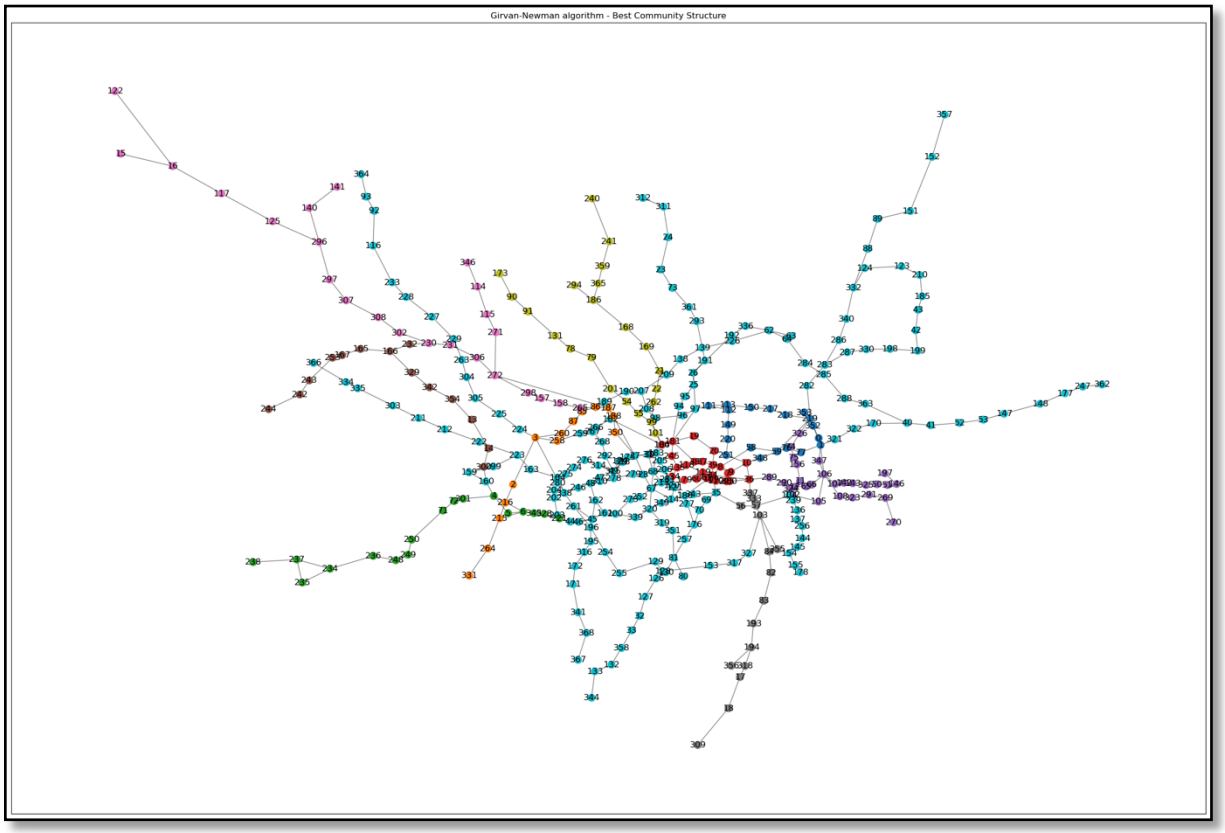

Fig 2.3.11.2 Network visualization of community generated through Girvan-Newman Algorithm

2.3.12 Strategic Recommendations

2.3.12.1 Enhancing Core and Component Resilience:

Investing in the K-core and significant K-components of the network can enhance its resilience by upgrading infrastructure, boosting service frequency, and deploying advanced monitoring and rapid response systems to maintain critical nodes and paths during different scenarios.

2.3.12.2 Optimizing Design for Redundancy and Efficiency:

Strategically enhancing sparse design with redundancy can mitigate vulnerabilities by creating alternative routes between cliques.

2.3.12.3 Leveraging Localized Connectivity:

Utilizing cliques for localized connectivity and redundancy enhances passenger experience and operational resilience through targeted improvements in these areas.

3 CONCLUSION

The London transport network exhibits a carefully balanced design that optimizes for coverage, efficiency, and, to a certain extent, resilience. The insights from this analysis not only highlight the network's structural strengths and potential vulnerabilities but also offer a roadmap for targeted enhancements. By focusing on strengthening critical nodes and connections, optimizing for both efficiency and redundancy, and leveraging localized connectivity, the network can improve its robustness, operational reliability, and service quality, ensuring it remains well-equipped to meet both current and future challenges.

4 References:

- https://www.tfl.gov.uk/ for details.
- Ref: Manlio De Domenico, Albert Solé-Ribalta, Sergio Gómez, and Alex Arenas, Navigability of interconnected networks under random failures. PNAS 111, 8351-8356 (2014)
- https://tfl.gov.uk/corporate/about-tfl/what-we-do#:~:text=London%20Underground,-London%20Underground%2C%20better&text=The%20Tube%20handles%20up%20to,trains%20whizzing%20around%20the%20capital.
- Heglund, J. S. W., Taleongpong, P., Hu, S., & Tran, H. T. (n.d.). *Railway Delay Prediction with Spatial-Temporal Graph Convolutional Networks*.
- Ilalokhoin, O., Pant, R., & Hall, J. W. (2023). A model and methodology for resilience assessment of interdependent rail networks – Case study of Great Britain's rail network. *Reliability Engineering and System Safety*, *229*. https://doi.org/10.1016/j.ress.2022.108895

- Impact of Delays on Passenger,Train Services,Evidence from Great Britain. John Preston, Graham Wall, Richard Batley, J. Nicolás Ibáñez,and Jeremy Shires 2009
- Levinson, D. (2008). Density and dispersion: The co-development of land use and rail in London. *Journal of Economic Geography*, *8*(1), 55–77. https://doi.org/10.1093/jeg/lbm038
- Wee, X. Bin, Herrera, M., Hadjidemetriou, G. M., & Parlikad, A. K. (2023). Simulation and Criticality Assessment of Urban Rail and Interdependent Infrastructure Networks. In *Transportation Research Record* (Vol. 2677, Issue 1, pp. 1181–1196). SAGE Publications Ltd. https://doi.org/10.1177/03611981221103594
- Cats, O., & Krishnakumari, P. (2020). Metropolitan rail network robustness. *Physica A: Statistical Mechanics and Its Applications*, *549*. https://doi.org/10.1016/j.physa.2020.124317
- Heglund, J. S. W., Taleongpong, P., Hu, S., & Tran, H. T. (n.d.). *Railway Delay Prediction with Spatial-Temporal Graph Convolutional Networks*.
- Greenham, S., Ferranti, E., Quinn, A., & Drayson, K. (2020). The impact of high temperatures and extreme heat to delays on the London Underground rail network: An empirical study. *Meteorological Applications*, *27*(3). https://doi.org/10.1002/met.1910

5 Appendix:

```
# Importing required libraries

import pandas as pd
import networkx as nx
import os
import matplotlib.pyplot as plt
import statistics as st
import community as community_louvain
import matplotlib.cm as cm
from networkx.algorithms.community import girvan_newman
from networkx.algorithms.community import modularity

#-----------------------------------------------------------------------------------------------------------------------

# Setting up the datasource paths
dataset_dir = 'Dataset'
nodes_file_path = os.path.join(dataset_dir, 'london_transport_nodes.txt')
multiplex_edges_file_path = os.path.join(dataset_dir, 'london_transport_multiplex.edges')
disruptions_summary_file_path = os.path.join(dataset_dir, 'london_transport_disruptions_summary.txt')

# Load node information
nodes_df = pd.read_csv(nodes_file_path, sep=' ', names=['nodeID', 'nodeLabel', 'nodeLat', 'nodeLong'],
                       skiprows=1)

# Load multiplex edges information
multiplex_edges_df = pd.read_csv(multiplex_edges_file_path, header=None, delim_whitespace=True,
                                 names=['layer', 'sourceLabel', 'targetLabel','targetlayer'])

# Load disruptions summary information
disruptions_df = pd.read_csv(disruptions_summary_file_path, sep=' ', skiprows=1,
                             names=['DISRUP_ID', 'LAYER',
                                    'STATION_A', 'STATION_B', 'FREQUENCY%', 'DAMAGED_NODES%'])

# Load layer information
layer_data = {
    'layer': [1, 2, 3],
    'Description': ['Tube', 'Overground', 'DLR']
}

layer_df = pd.DataFrame(layer_data)

multiplex_edges_source_df = multiplex_edges_df[['layer', 'sourceLabel']].rename(columns={'layer': 'layer
', 'sourceLabel': 'nodeID'})
multiplex_edges_target_df = multiplex_edges_df[['targetlayer', 'targetLabel']].rename(columns={'targetla
yer': 'layer', 'targetLabel': 'nodeID'})
node_layer_df = pd.concat([multiplex_edges_source_df, multiplex_edges_target_df], ignore_index=True)
node_layer_df = node_layer_df.drop_duplicates()
node_layer_df = pd.merge(node_layer_df, layer_df, on='layer', how='left')
nodes_df = pd.merge(nodes_df, node_layer_df, on='nodeID', how='left')
```

```
node_layer_count = nodes_df.groupby('Description').count().reset_index()

node_layer_count = node_layer_count[['Description', 'layer']].rename(columns={'Description': 'layer', 'layer': 'Count'})
display(node_layer_count)

# Creating a multiplex graph
G_multiplex = nx.Graph()

# Add nodes with attributes into multiplex graph
for index, row in nodes_df.iterrows():
    G_multiplex.add_node(row['nodeID'], label=row['nodeLabel'], lat=row['nodeLat'], long=row['nodeLong'], layer=row['Description'])

# Add multiplex edges with layer information into multiplex graph
for index, row in multiplex_edges_df.iterrows():
    if row['sourceLabel'] in G_multiplex.nodes and row['targetLabel'] in G_multiplex.nodes:
        G_multiplex.add_edge(row['sourceLabel'], row['targetLabel'], layer=row['layer'])

# Preprocess disruptions data
label_to_id = {row['label']: node for node, row in G_multiplex.nodes(data=True)}
nx.set_edge_attributes(G_multiplex, 0, 'FREQUENCY%')
nx.set_edge_attributes(G_multiplex, 0, 'DAMAGED_NODES%')

# Incorporate disruptions into multiplex graph
for index, row in disruptions_df.iterrows():
    station_a_id = label_to_id.get(row['STATION_A'])
    station_b_id = label_to_id.get(row['STATION_B'])
    if station_a_id and station_b_id and G_multiplex.has_edge(station_a_id, station_b_id):
        G_multiplex[station_a_id][station_b_id]['FREQUENCY%'] = row['FREQUENCY%']
        G_multiplex[station_a_id][station_b_id]['DAMAGED_NODES%'] = row['DAMAGED_NODES%']


# Remove self-loops from the graph
G_multiplex.remove_edges_from(nx.selfloop_edges(G_multiplex))

# Remove isolated nodes from the graph
G_multiplex.remove_nodes_from(list(nx.isolates(G_multiplex)))

# Graph Info
print(f"Graph has {G_multiplex.number_of_nodes()} nodes and {G_multiplex.number_of_edges()} edges.")

#-----------------------------------------------------------------------------------------------------------------------

# Visualizing the Graph
#Defining the Color Map
layer_color_map = {'Tube': 'yellow', 'Overground': 'red', 'DLR': 'blue'}
```

```
# Geographical layout
# 'lat' and 'long' attributes are correctly assigned
pos = {node: (data['long'], data['lat']) for node, data in G_multiplex.nodes(data=True)}


# Setting node colors
node_colors = [layer_color_map[node[1]['layer']] for node in G_multiplex.nodes(data=True)]

# Visualize the Network
plt.figure(figsize=(12, 8))


# Using the geographical 'pos' for layou
nx.draw(G_multiplex, pos, with_labels=True, node_size=50, font_size=10, node_color=node_colors, edg
e_color='gray')
plt.title('London Transport Network Visualization')
plt.show()

#-----------------------------------------------------------------------------------------------------------------------

density = nx.density(G_multiplex)
print(f"Network Density: {density}")

#-----------------------------------------------------------------------------------------------------------------------

degree_centrality = nx.degree_centrality(G_multiplex)

# Converting to a DataFrame for further analysis
dc_df = pd.DataFrame(degree_centrality.items(), columns=['Node', 'Degree Centrality']).sort_values('Deg
ree Centrality', ascending=False)

# Plotting the histogram
dc_df = dc_df.sort_values(by='Degree Centrality')
plt.figure(figsize=(10, 6))  # Optional: Adjust the figure size

n, bins, patches = plt.hist(dc_df['Degree Centrality'], bins=10, color='blue', edgecolor='black')

# Adding count labels above each bar
for i in range(len(patches)):
    plt.annotate(str(int(n[i]))+" nodes", xy=(patches[i].get_x() + patches[i].get_width() / 2, patches[i].get_h
eight()),
                 xytext=(0, 5), textcoords="offset points",
                 ha='center', va='bottom')

plt.title('Histogram of Degree Centrality with Counts')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```

```
# Normalize the betweenness values to use them for node size
# Multiplying by a constant to better see size differences
scaled_degree = [3000 * v for v in degree_centrality.values()]

# Create a color map based on the betweenness centrality
color_map = [v for v in degree_centrality.values()]

plt.figure(figsize=(15, 10))

# Draw the graph with node color indicating betweenness centrality
pos = {node: (data['long'], data['lat']) for node, data in G_multiplex.nodes(data=True)}
nx.draw_networkx_edges(G_multiplex, pos, alpha=0.5)
nodes = nx.draw_networkx_nodes(
    G_multiplex,
    pos,
    node_size=scaled_degree,  # Use scaled betweenness for node size
    node_color=color_map,  # Use raw betweenness values for color mapping
    cmap=plt.cm.YlOrRd  # Color map for betweenness centrality
)
nx.draw_networkx_labels(G_multiplex, pos)
plt.colorbar(nodes)  # Add color bar to show mapping from color to betweenness centrality
plt.title('Graph Visualization with Degree Centrality')
plt.axis('off')  # Turn off the axis
plt.show()

#-----------------------------------------------------------------------------------------------------------------------

closeness_centrality = nx.closeness_centrality(G_multiplex)

# Converting to a DataFrame for further analysis
cc_df = pd.DataFrame(closeness_centrality.items(), columns=['Node', 'Closeness Centrality']).sort_values
('Closeness Centrality', ascending=False)


# Plotting the histogram
cc_df = cc_df.sort_values(by='Closeness Centrality')
plt.figure(figsize=(10, 6))  # Optional: Adjust the figure size

n, bins, patches = plt.hist(cc_df['Closeness Centrality'], bins=10, color='blue', edgecolor='black')

# Adding count labels above each bar
for i in range(len(patches)):
    plt.annotate(str(int(n[i]))+" nodes", xy=(patches[i].get_x() + patches[i].get_width() / 2, patches[i].get_h
eight()),
                 xytext=(0, 5), textcoords="offset points",
                 ha='center', va='bottom')

plt.title('Histogram of Closeness Centrality with Counts')
plt.xlabel('Value')
```

```
plt.ylabel('Frequency')
plt.show()

# Calculate closeness centrality for each node
closeness_centrality = nx.closeness_centrality(G_multiplex)

# Normalize the closeness values to use them for node size
# You may want to multiply by a constant to better see size differences
scaled_closeness = [1000 * v for v in closeness_centrality.values()]

# Create a color map based on the closeness centrality
color_map = [v for v in closeness_centrality.values()]

plt.figure(figsize=(15, 10))
# Draw the graph with node color indicating closeness centrality
pos = {node: (data['long'], data['lat']) for node, data in G_multiplex.nodes(data=True)}
nx.draw_networkx_edges(G_multiplex, pos, alpha=0.5)
nodes = nx.draw_networkx_nodes(
    G_multiplex,
    pos,
    node_size=scaled_closeness,  # Use scaled closeness for node size
    node_color=color_map,  # Use raw closeness values for color mapping
    cmap=plt.cm.viridis  # Color map for closeness centrality
)
nx.draw_networkx_labels(G_multiplex, pos, font_size=8)  # Reduce font size if it's too cluttered
plt.colorbar(nodes)  # Add color bar to show mapping from color to closeness centrality
plt.title('Graph Visualization with Closeness Centrality')
plt.axis('off')  # Turn off the axis
plt.show()

#-----------------------------------------------------------------------------------------------------------------------

connected_components = list(nx.connected_components(G_multiplex))
print(f"Number of Connected Components: {len(connected_components)}")

#-----------------------------------------------------------------------------------------------------------------------

k_core = nx.k_core(G_multiplex).nodes()
print(f"K-Core Nodes: {len(k_core)}")

#-----------------------------------------------------------------------------------------------------------------------

cliques = list(nx.find_cliques(G_multiplex))
num_cliques = len(cliques)
max_clique_size = max(len(clique) for clique in cliques)
cliques
print(f"Num of Cliques: {num_cliques}")
print(f"Max Clique Size: {max_clique_size}")

#-----------------------------------------------------------------------------------------------------------------------
```

```
#Clustering
cluster_coef = nx.clustering(G_multiplex, nodes=None, weight=None)
st.mean(cluster_coef.values())

#-----------------------------------------------------------------------------------------------------------------------

#Modularity
main_core = set(nx.k_core(G_multiplex))
core=nx.core_number(G_multiplex)
core_values = list(set(core.values()))
nodes = list(G_multiplex.nodes())

nodes_partition = []

for i in range(len(core_values)):
    nodes_partition.append(set())

for i in range(0,len(nodes)):
    for j in range(0,len(core_values)):
        if list(core.values())[i] == core_values[j]:
            nodes_partition[j].add(nodes[i])

#-----------------------------------------------------------------------------------------------------------------------

#Louvain method
# Assuming G_multiplex is your network graph

# Detecting communities with the Louvain method
partition = community_louvain.best_partition(G_multiplex)

# Visualizing the detected communities
# Each community will be colored with a different color in the network
pos = {node: (data['long'], data['lat']) for node, data in G_multiplex.nodes(data=True)}

# color the nodes according to their partition
cmap = cm.get_cmap('viridis', max(partition.values()) + 1)
nx.draw_networkx_nodes(G_multiplex, pos, partition.keys(), node_size=20,
                       cmap=cmap, node_color=list(partition.values()))
nx.draw_networkx_edges(G_multiplex, pos, alpha=0.5)
plt.show()

# Print the number of communities detected
print(f"Number of communities detected: {max(partition.values()) + 1}")

# Optionally, to inspect nodes in each community:
from collections import defaultdict
communities = defaultdict(list)
for node, community in partition.items():
    communities[community].append(node)

# Print out nodes in the first few communities
for i, (community, nodes) in enumerate(communities.items()):
```

```
    print(f"Community {community}: {nodes}")
    if i > 5:  # Limit to printing only the first few communities
        break

#-----------------------------------------------------------------------------------------------------------------------

# Apply the Girvan-Newman algorithm
communities_generator = girvan_newman(G_multiplex)
modularity_values = []
num_iterations = []
communities_list = []

# Calculate initial modularity
initial_communities = tuple(sorted(c) for c in nx.connected_components(G_multiplex))
initial_modularity = modularity(G_multiplex, initial_communities)
modularity_values.append(initial_modularity)
num_iterations.append(0)
communities_list.append(initial_communities)

# Girvan-Newman iterative process
i = 1
try:
    while True:
        communities = next(communities_generator)
        curr_modularity = modularity(G_multiplex, communities)
        modularity_values.append(curr_modularity)
        num_iterations.append(i)
        communities_list.append(communities)
        i += 1
except StopIteration:
    pass  # When the generator has no more communities, it will raise StopIteration.

# Plotting the trend
plt.figure(figsize=(30, 15))
plt.plot(num_iterations, modularity_values, marker='o')
plt.title('Modularity Trend for Girvan-Newman Algorithm')
plt.xlabel('Number of Iterations')
plt.ylabel('Modularity')
plt.grid(True)
plt.show()

# Assuming you want to visualize the community with the highest modularity
best_iteration = modularity_values.index(max(modularity_values))
best_communities = communities_list[best_iteration]

# Create a color map for the graph with the best community structure
color_map = []
for node in G_multiplex:
    for index, community in enumerate(best_communities):
        if node in community:
            color_map.append(plt.cm.tab10(index))
```

```
plt.figure(figsize=(30, 20))
# Draw the graph
pos = {node: (data['long'], data['lat']) for node, data in G_multiplex.nodes(data=True)}
nx.draw_networkx_edges(G_multiplex, pos, alpha=0.5)
nx.draw_networkx_nodes(G_multiplex, pos, node_color=color_map, node_size=100)
nx.draw_networkx_labels(G_multiplex, pos)

# Set the title of the plot
plt.title("Girvan-Newman algorithm - Best Community Structure")

# Display the plot
plt.show()

#-------------------------------------------------------------------------------------------------------------------------
```