MA214 Network Analysis

Group Assignment

Topic

International Europe Road Network Analysis

Group Name

Powerpuff

Fathur M Haekal Student ID: 2315788

Group Member

1. Lattapol Sae-Aue Student ID: 2315529
2. Fathur M Haekal Student ID: 2315788
3. Haris Fayyaz Student ID: 2311621
4. Akshatha Arun Student ID: 2311188
5. Huey Jian Foo Student ID: 2311922

School of Mathematics, Statistics and Actuarial Science

University of Essex

1 Abstract

Network analysis plays a pivotal role in unravelling the intricate dynamics of complex systems, offering insights into the structure, behaviour, and interactions within them. In the context of transportation systems, such analysis becomes particularly relevant, aiding in the optimization of routes, understanding traffic flow patterns, and facilitating efficient resource allocation. In this study, we focus on the international E-road network, a vital component of European transportation infrastructure. The E-road network spans across multiple countries, connecting various cities and regions, thereby serving as a backbone for transcontinental travel and trade. By employing network analysis techniques on this dataset, we aim to uncover crucial aspects such as connectivity patterns, centrality of nodes, and potential bottlenecks. Understanding these dynamics is paramount for policymakers, urban planners, and transportation authorities in devising strategies for enhancing connectivity, improving transportation efficiency, and fostering economic growth. The dataset detailing the E-road network's topology holds significant promise in addressing these research questions, offering a comprehensive perspective on the intricacies of European transportation networks and guiding informed decision-making processes.

2 Literature Review

2.1 Neural Network

Neural networks have progressed significantly since their inception, demonstrating exceptional abilities in pattern recognition, data analysis, and simulating human intelligence. LeCun et al. (2015) emphasized their transformative impact in deep learning, particularly through convolutional neural networks (CNNs) for tasks like image and video recognition. Hinton et al. (2012) contributed to understanding deep neural networks (DNNs), showcasing their effectiveness in reducing error rates in speech recognition. Schmidhuber (2015) highlighted the flexibility of neural networks in sequential data processing, advancing natural language processing and machine translation. Despite these achievements, Goodfellow et al. (2016) pointed out the vulnerability of neural networks to adversarial attacks, indicating a critical area for ongoing research. The evolving landscape of neural network research underscores its dynamic growth and wide applicability while emphasizing the importance of addressing its limitations.

Recent studies have focused on the security vulnerabilities of neural networks, highlighting the risks associated with cyberattacks targeting these systems. Concerns include adversarial attacks, data poisoning during training, and exposure of sensitive data, prompting researchers to work on enhancing security measures like developing algorithms such as V-CNN for automatic vulnerability detection using convolutional neural networks. Additionally, studies have revealed that AI tools are more susceptible than previously thought to targeted attacks that manipulate data inputs to deceive AI systems into making incorrect decisions, emphasizing the need for robust defences against adversarial vulnerabilities.

2.2 E-Road Network

The European E-Road Network, overseen by the United Nations Economic Commission for Europe (UNECE), spans over 150,000 kilometers of routes across Europe with the aim of enhancing cross-border transportation, fostering economic growth, and improving connectivity among European nations. This network plays a crucial role in sustainable transportation by integrating environmental considerations into its development (Dimitriou and Stathopoulos, 2013). Additionally, it has a significant economic impact by supporting trade and mobility within the European Single Market (Vickerman, 2002). Challenges such as maintenance, funding, and adapting to emerging transportation technologies are being addressed to ensure the network's operational efficiency and continuous adaptation to evolving transportation needs (Schippl et al., 2013).

The EU core road network is a vital component of this system, aiming to reduce travel times and improve connectivity. While progress has been made in developing the TEN-T core road network supported by EU funding and Commission action, challenges remain in achieving full functionality

due to factors like insufficient maintenance and barriers for seamless travel (Special Report: The EU core road network - Publications Office). The completion rates of the core network vary among Member States, with central and eastern European countries lagging behind western counterparts (Special report: The EU core road network - Publications Office). The International E-Road Network covers approximately 93,000 miles, linking major cities and industrial areas across Europe and into Central Asia. It has evolved over the years with a numbering system introduced in 1975 and revised in 1992 to categorize routes based on their significance (European Trade & Logistics: The Development of the International E-Road Network).

Overall, the European E-Road Network is a critical infrastructure that not only facilitates transportation but also contributes significantly to economic development and regional integration within Europe.

2.3 Adjacency Matrix

The European E-Road Network, under the supervision of the United Nations Economic Commission for Europe (UNECE), spans more than 150,000 kilometers of routes across Europe with the goal of enhancing cross-border transportation, fostering economic growth, and improving connectivity among European nations. This network is pivotal in sustainable transportation by incorporating environmental considerations into its development (Dimitriou and Stathopoulos, 2013). Kipf and Welling (2016) further advanced this concept by introducing the Graph Convolutional Network (GCN), which efficiently utilizes the adjacency matrix for spectral filtering to enable learning on graph-structured data. Hamilton et al. (2017) expanded on these ideas with GraphSAGE, a framework for inductive representation learning on large graphs that efficiently samples and aggregates features from a node's neighborhood using adjacency matrices. Additionally, Veličković et al. (2017) highlighted the adjacency matrix's role in enabling attention mechanisms in graph neural networks to assign varying weights to edges, thereby enhancing model performance in tasks like node classification. These studies collectively emphasize the critical role of the adjacency matrix in graph neural networks, revolutionizing the handling of structured data and driving advancements across various domains from social network analysis to bioinformatics.

2.4 Shortest Path

The challenge of determining the shortest path in graphs is a traditional optimization problem that has witnessed innovative solutions with the emergence of neural networks. While historically addressed using algorithms like Dijkstra's or Bellman-Ford, recent neural network approaches provide dynamic and scalable solutions, particularly in intricate and evolving environments. Bello et al. (2016) introduced a pioneering method that employs reinforcement learning with neural networks to learn policies for shortest path dilemmas, showcasing the potential for neural networks

to surpass classical heuristics in specific scenarios. In parallel, Khalil et al. (2017) presented a graph embedding technique utilizing graph neural networks (GNNs) to encode graph structures into a lower-dimensional space, simplifying the shortest path problem into a learning task where the network predicts the subsequent node on the path. Additionally, Li et al. (2018) proposed a model that merges attention mechanisms with graph neural networks to directly learn shortest paths in weighted graphs by focusing on pertinent parts of the graph. These studies mark a shift towards integrating neural network models for graph-related challenges, offering insights into applying deep learning to conventional algorithmic problems and hinting at a future where neural networks could deliver adaptable, efficient solutions for dynamic pathfinding and graph analysis tasks.

2.5 Density

The concept of density in neural networks pertains to the connectivity and allocation of parameters within a network, influencing its capacity, efficiency, and performance. High-density networks, characterized by a large number of parameters and connections, can capture complex patterns but often at the cost of increased computational resources and the risk of overfitting. On the other hand, low-density networks, such as those employing sparse connections or pruning techniques, aim to reduce computational load and improve generalization by eliminating redundant connections. Han et al. (2015) demonstrated that pruning a trained neural network can significantly reduce its complexity without sacrificing accuracy, highlighting the potential of sparse networks. Further, Glorot et al. (2011) explored the impact of network initialization on density, suggesting that proper initialization can improve the training of deep networks by maintaining a balance between density and sparsity. The advent of techniques such as dropout, introduced by Srivastava et al. (2014), further emphasizes the utility of controlling network density, showing that randomly reducing network connections during training can prevent overfitting while maintaining network capacity. Moreover, recent advancements in dynamic sparsity, as explored by Mostafa and Wang (2019), offer mechanisms to adjust a network's density throughout training, enabling adaptive optimization of computational efficiency and model performance. These studies collectively underscore the critical balance between density and sparsity in neural networks, pointing towards evolving strategies to optimize this aspect for enhanced performance and efficiency.

2.6 Degree Distribution

Degree distribution in neural networks, particularly in the context of graph neural networks (GNNs), plays a crucial role in understanding the structural properties of networks and their impact on learning and generalization. The degree distribution of a graph refers to the distribution of the number of connections (edges) each node (neuron) has with other nodes in the network, which significantly affects the network's ability to process and propagate information. Monti et al. (2017)

highlighted the importance of considering the local graph structure, including degree distribution, in designing convolutional operations on graphs, which can adapt to varying node degrees for better feature extraction. In a seminal work by Xu et al. (2018), the authors propose a framework that generalizes convolutional neural networks (CNNs) to graph-structured data, taking into account the degree distribution to weigh neighbor contributions differently, thereby enhancing the model's capacity to capture graph dynamics. Additionally, Hamilton et al. (2017) introduced an inductive learning framework that efficiently leverages node degree information to sample and aggregate features from neighbors, improving scalability and performance in large graphs. These studies underline the degree distribution's pivotal role in influencing the efficiency and effectiveness of neural network architectures designed for graph data, indicating a growing recognition of the need to incorporate topological characteristics into neural network models for better performance on graph-related tasks.

2.7 K-Core

The concept of the k-core, integral to graph theory, has been explored within the context of neural networks, especially in understanding network structure and function. K-cores are used to identify the hierarchical structure of networks by recursively peeling off nodes with degree less than k, which can reveal densely connected substructures within the network. This concept has found application in analyzing the robustness and efficiency of neural network architectures. For example, Alvarez-Hamelin et al. (2005) utilized k-core decomposition to study the Internet's topology, offering insights into the structural resilience of large-scale networks, which can be analogously applied to understand the resilience of neural network architectures against node failures or adversarial attacks. Further, Monti et al. (2017) introduced geometric deep learning models that incorporate k-core decomposition for classifying the structures within graph datasets, demonstrating how k-core analysis can enhance the interpretability and performance of graph neural networks (GNNs). Additionally, the role of k-core analysis in simplifying neural networks for computational efficiency has been investigated, showing potential in pruning strategies to remove less significant nodes without compromising network performance significantly. While direct references to the application of k-core principles specifically within neural networks are emerging, these studies indicate a growing interest in leveraging graph-theoretic measures to optimize and understand neural network architectures.

2.8 Closeness and Betweenness Centrality

Closeness and betweenness centrality are key metrics used in network analysis to understand the significance and strategic positioning of nodes within a network. In the realm of neural networks, especially when analyzing complex or social network data, these centrality measures are employed to identify influential nodes and comprehend information flow dynamics. Newman (2005)

discusses the role of betweenness centrality in identifying nodes that serve as critical bridges in the network, while closeness centrality is highlighted for its ability to pinpoint nodes that can efficiently disseminate information due to their proximity to others in the network. Borgatti (2005) further explores these concepts, emphasizing their application in social network analysis to understand power dynamics and communication efficiency. Recent studies, such as those by Li et al. (2018), integrate these centrality measures into graph neural networks (GNNs), demonstrating how they can inform network embedding processes and improve model performance in tasks like node classification and link prediction. Moreover, Lee et al. (2019) propose a novel approach to incorporate centrality measures directly into the training process of GNNs, enhancing their ability to capture nuanced structural characteristics. These developments underscore the growing intersection between traditional network analysis metrics and modern neural network architectures, showcasing the potential of centrality measures to enhance our understanding and modeling of complex networks.

2.9 Clustering Coefficient

In neural networks, particularly graph neural networks (GNNs), the clustering coefficient measures how nodes tend to cluster together in a graph. This metric is vital for understanding the structural properties of networks like social, biological, and knowledge graphs. Watts and Strogatz (1998) introduced the small-world network model, emphasizing the significance of the clustering coefficient in characterizing local neighborhood connectivity, which greatly impacts information transfer efficiency within networks. In the realm of GNNs, the clustering coefficient is utilized to improve learning algorithms by integrating local structural information, thereby enhancing the accuracy of predictions related to node and graph properties. Latora and Marchiori (2001) discussed how clustering coefficient, along with shortest path length, contributes to the small-world phenomenon, affecting the dynamics of network processes such as spreading and synchronization. More recent studies, such as by Hamilton et al. (2017) with GraphSAGE, and by Veličković et al. (2018) with Graph Attention Networks (GATs), have implicitly leveraged concepts akin to clustering coefficients to aggregate information from a node’s neighbors efficiently, thereby encoding local structural information into the learning process. These approaches underline the significance of understanding and incorporating clustering coefficients into neural network models, particularly for tasks involving complex graph structures, to improve the robustness and interpretability of the models.

3 Dataset Description

The European road network serves as a critical infrastructure, facilitating transportation and connectivity across the continent. Understanding its structure and dynamics is of paramount importance for various applications ranging from urban planning to transportation logistics. In this study, we delve into the analysis of the European road network graph, aiming to uncover its underlying properties and community structure.

The dataset under investigation comprises 1174 nodes, each representing a city, interconnected by 1417 undirected and unweighted edges representing direct roads. The average distance between nodes is approximately 18 units and its diameter, of the main component is 62 units.

Figure 1 Visualisation of full euroroad network

- Euroroad
    - https://networkrepository.com/inf-euroroad.php
- City Names
    - http://konect.cc/networks/subelj_euroroad/

4 Analysis

In this section, we comprehensively present the results and analyses derived from the dataset. The analysis is subdivided into distinct sections to facilitate a systematic exploration of the data.

4.1 Check for Acyclic

The identification of cycles within the road network presents essential insights into the nature of transportation systems.

The code snippet

```
# Check for Acyclic
is_acyclic = nx.is_directed_acyclic_graph(graph)
```

Detects the presence of cycles in the network. The presence of cycles signifies potential routes that allow for the return to the starting point, indicating the possibility of circular or repetitive circulation. While cycles can sometimes introduce challenges in routing algorithms, they also offer opportunities for exploring alternative paths and optimizing travel routes. By acknowledging the existence of cycles, transportation planners can devise routing strategies that account for both efficiency and redundancy, balancing the need for direct routes with the flexibility to navigate through different paths.

Understanding the cyclic nature of the road network is crucial for managing traffic flow dynamics effectively. Cycles contribute to the formation of interconnected routes, allowing for the redistribution of traffic and alleviating congestion in specific areas. By leveraging the cyclical structure of the network, transportation authorities can implement measures to regulate traffic flow, optimize signal timings, and improve overall mobility. Additionally, awareness of cycles enables the anticipation and mitigation of potential bottlenecks, leading to smoother operations and reduced delays for commuters.

The code below uses cycle_basis that returns a list of cycles and enumerate_all_cliques to obtain the cliques. To get the triangles, we filter out only those cliques that have three nodes.

```
# Check for the number of cycles
cycle_count = sum(1 for cycle in nx.cycle_basis(graph))
print("Number of cycles:", cycle_count)

# Check for the number of triangles
triangle_count = sum(1 for _ in nx.enumerate_all_cliques(graph) if len(_)
== 3)
print("Number of triangles:", triangle_count)
```

The network exhibits a complex structure characterized by multiple loops or circular paths, as evidenced by the presence of 268 cycles. This complexity suggests a rich interconnectedness among its nodes, facilitating diverse and intricate routes between them. Additionally, the network contains 32 triangles, indicative of localized clustering or cohesion among sets of three interconnected nodes. These triangles highlight areas of high local connectivity, where nodes share common neighbors, fostering efficient information exchange and interaction.

4.2 Density

In network analysis, density refers to a measure of the connectedness or completeness of a network. It quantifies the proportion of actual connections in a network relative to the total possible connections that could exist. Density measures the proportion of possible connections in the network that are actually present.

In the context of the European road network, density provides insights into the level of interconnectedness among cities and the overall robustness of the transportation system

Density ranges from 0 to 1, with:

- A density of 0 indicates no connections between nodes, representing a sparse network.
- A density of 1 indicates that all possible connections between nodes exist, representing a fully connected network.

```
density = nx.density(graph)
```

This calculates the density of the graph using the nx.density() function from the NetworkX library. The density value is stored in the variable density.

print(f"Density: {density}") This prints out the density value calculated in the previous step. The f-string notation (f"...") allows to embed variables directly within a string in Python. A density value of 0.0021 indicates that the network is sparse, with relatively few connections compared to the total number of possible connections.

4.3 Degree Distribution

```
# Compute the mean degrees
mean_degree = sum(dict(degree).values()) / node_count
```

The mean degree is calculated by dividing the degree of each node with the number of nodes in the network. This value indicates that, on average, each intersection is connected to about 2.41

roads. Higher average degree suggests a denser road network, implying a greater level of connectivity between intersections. In the context of traffic flow, intersections with higher degrees are likely to experience heavier traffic, signifying potential congestion points. Conversely, intersections with lower degrees may have lower traffic volumes. Understanding the degrees of specific nodes can help identify critical intersections with high connectivity, which are crucial for efficient urban planning and traffic management strategies.

Subsequently, the code employs np.unique function

```
degree_unique = np.unique(degree_sequence, return_counts=True)
```

to obtain the counts of unique degree values in the dataset. This analysis breaks down the distribution of degrees among intersections, providing a clearer understanding of the network's structural characteristics.

The subsequent printed output displays the counts of each unique degree value along with its corresponding frequency. For example, the table shows that there are 190 intersections with a degree of 1, 612 intersections with a degree of 2, and so forth. This breakdown facilitates a deeper examination of the distribution of connectivity levels across the road network.

The provided data represents the counts of unique degree values in the graph:

Degree	Count
1	190
2	612
3	186
4	113
5	47
6	15
7	5
8	5
10	1

The degree distribution analysis of road networks provides valuable insights with significant real-world implications for transportation planning, infrastructure development, and traffic management strategies. Nodes with a degree of 1 indicate the existence of numerous isolated

entities, likely representing peripheral or less influential locations within the network. Conversely, the high frequency of nodes with a degree of 2 suggests a prevalence of linear structures or chains. Furthermore, the rarity of nodes with higher degrees, such as 5, 6, 7, 8, and 10, implies the presence of central hubs connecting multiple parts of the network. These findings underscore the critical role of certain nodes in facilitating efficient traffic flow and connectivity within the road network. By leveraging this understanding, stakeholders can prioritize maintenance efforts, improve traffic flow, and enhance the resilience of the network to disruptions or failures.

4.4 K-Core

The k-core visualisation helps us understand its structural properties. The figure below shows the resulting k-core graph.

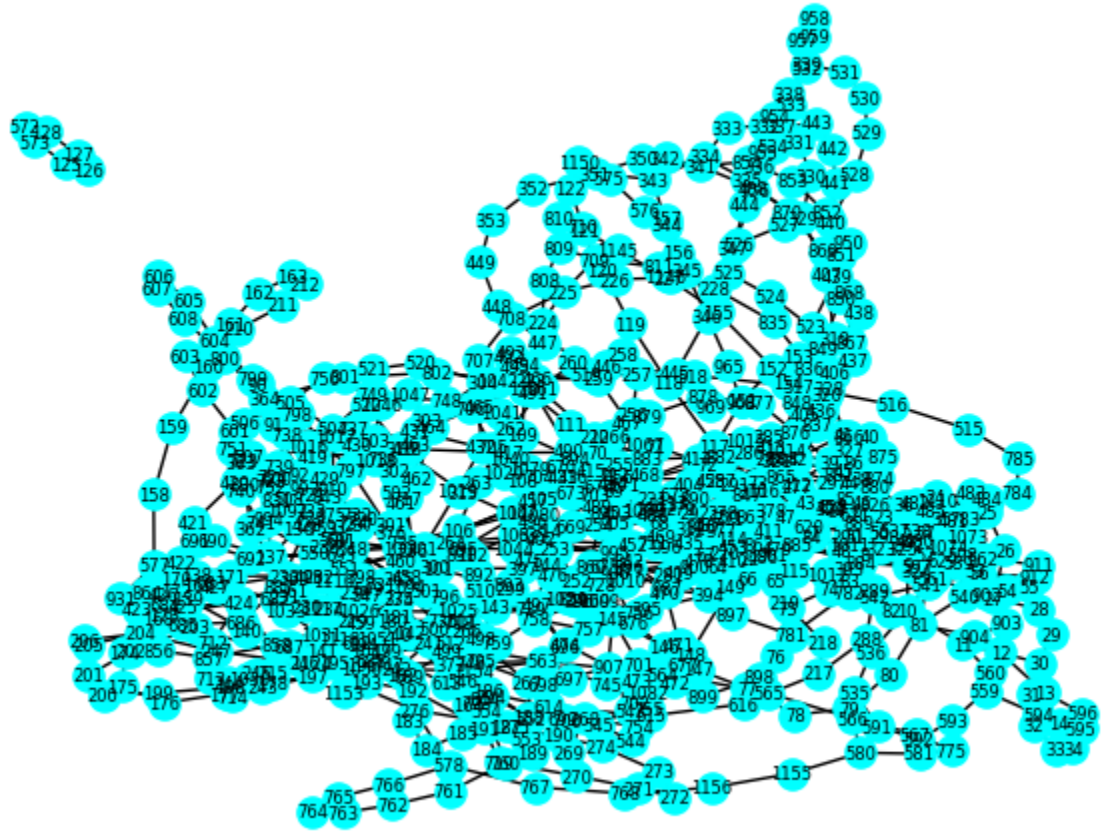

Figure 2 Visualisation of the k-core graph

```
# Initialize a dictionary to store the count of nodes for each core value
core_count = {}
```

```
# Count the number of nodes for each core value
for node, core_value in core_numbers.items():
    if core_value in core_count:
        core_count[core_value] += 1
    else:
        core_count[core_value] = 1

# Print the number of nodes for each core value
for core_value, count in core_count.items():
    print(f"Number of nodes for core value {core_value}: {count}")
```

The code above calculates core value for each node in the graph then uses a for loop to get the count for each core value and prints them.

The analysis of the network's core values, comprising 431 nodes assigned to core value 1 and 742 nodes to core value 2, sheds light on its underlying structure and connectivity dynamics, particularly when considering a road network dataset. The predominance of core values 1 and 2 suggests limited connectivity within the graph, indicative of sparse or restricted interconnections among nodes. This sparse connectivity is akin to certain characteristics observed in road networks, where nodes represent intersections or endpoints, and connections represent roads or pathways.

Nodes labelled with core value 1 in the road network dataset can be likened to peripheral elements, signifying isolated entities or endpoints with minimal direct connections to other nodes. These nodes may correspond to remote or less-travelled areas, lacking strong connections to central traffic hubs or major road arteries. Conversely, nodes designated with core value 2 indicate the presence of linear structures or chains within the network, reminiscent of sequential arrangements of nodes along roadways, such as highways or main thoroughfares.

This configuration of core values paints a picture of the road network as characterized by sparse interconnectivity, with nodes predominantly falling into either peripheral or linearly connected categories. In the context of a road network, such a structure can have significant implications for traffic flow, route planning, and infrastructure development. Understanding these characteristics is crucial for devising strategies to optimize traffic management, enhance transportation efficiency, and improve overall network resilience.

4.5 Closeness and Betweenness Centrality

Closeness centrality quantifies how central each city is in terms of accessibility, reflecting its importance for efficient travel and communication. Meanwhile, betweenness centrality measures the extent to which a city acts as a bridge or bottleneck in connecting different parts of the network, influencing the flow of traffic and information. nx.closeness_centrality() and

nx.betweenness_centrality() are used to obtain the values for each node. Then each nodes are match to its city after it is sorted in a descending order using sorted() with reverse=True

```
# Closeness centrality
closeness = nx.closeness_centrality(graph)
# Betweenness Centrality
betweenness = dict(nx.betweenness_centrality(graph))
```

Identifying the cities based on closeness centrality reveals key hubs in the road network, essential for facilitating efficient transportation and communication. These cities likely serve as major transportation hubs or pivotal points of connectivity within the network. Their strategic positioning suggests they are well-connected to surrounding cities, making them essential for facilitating efficient transportation and communication. Warsaw, being the most central, may act as a primary hub, connecting various regions within the network. The proximity of these cities could contribute significantly to minimizing travel times and maximizing accessibility for commuters and travelers. Furthermore, their high closeness centrality indicates their importance in maintaining the overall cohesion and functionality of the road network.

The table below shows the top 5 cities ranked by closeness centrality

Node	City	Closeness Centrality
401	Warsaw	0.0769
402	Brest	0.0768
403	Minsk	0.0754
432	Lviv	0.0741
1019	Lublin	0.0740

Similarly, determining the cities with the highest betweenness centrality unveils critical nodes that act as vital connectors or bottlenecks within the network. These cities serve as crucial junctions or crossroads, influencing the flow of traffic and serving as key points of intersection between different routes. Understanding the importance of these cities is essential for optimizing traffic management, identifying potential congestion hotspots, and enhancing overall network resilience.

The table below shows the top 5 cities ranked by betweenness centrality:

Node	City	Betweenness Centrality
402	Brest	0.2148
284	Moscow	0.2125
277	Saint Petersburg	0.1955

453	Le Mans	0.1751
452	Rennes	0.1744

Analyzing these centrality measures in conjunction with the dataset's characteristics and real-world implications provides valuable insights for urban planning, transportation management, and infrastructure development. By identifying central hubs and critical connectors within the road network, decision-makers can devise strategies to improve traffic flow, mitigate congestion, and enhance the overall efficiency and resilience of transportation systems.

4.6 Clustering Coefficient

The clustering coefficient provides insights into the network's structure, such as the presence of communities or the degree of transitivity. High clustering coefficients indicate that nodes tend to form clusters, while low clustering coefficients suggest a more random or decentralized network structure. The clustering coefficient of a node measures the degree to which its neighbors are interconnected. It quantifies how close its neighbors are to being a complete subgraph. Higher clustering coefficients indicate that the node's neighbors are more interconnected, while lower coefficients indicate less interconnectedness among its neighbors.

```
# Compute the local clustering coefficient of every node
clustering_coefficients = nx.clustering(graph, nodes=None, weight=None)
```

This code calculates the clustering coefficient for each node in a graph. These are the top 8 clustering coefficient values in our data:

Node	City	Clustering Coefficient
157	Brindisi	1.0
205	Domokos	1.0
463	Vyšné Nemecké	1.0
544	Postojna	1.0
700	Spielfeld	1.0
50	Lausanne	0.3333333333333333
541	Tortona	0.3333333333333333
542	Brescia	0.3333333333333333

The global clustering coefficient is a metric used in network analysis to quantify the level of clustering or connectivity in a network. It provides insight into how tightly knit the network is as a whole. The global clustering coefficient is an important metric for understanding the level of clustering or cohesion in a network. It quantifies the extent to which nodes in the network tend to cluster together and form triangles or closed loops. In a network, a triangle is formed when three nodes are connected such that there are edges between all three nodes. The global clustering coefficient calculates the proportion of all possible triangles in the network that are actually present. This indicates the overall tendency of nodes in the network to form triangles or clusters. With a value of 0.0339, the network's global clustering coefficient is relatively low, suggesting that while some clustering exists, it is not prevalent throughout the entire network.

```
# Calculate average local clustering coefficient and global clustering
coefficient
average_local_clustering_coefficient =
st.mean(nx.clustering(graph).values())
global_clustering_coefficient = nx.transitivity(graph)
```

The average local clustering coefficient is relatively low (0.0167), indicating that individual nodes in the network are not strongly inclined to form tightly knit clusters with their neighbors. This suggests a sparse local clustering pattern, where nodes are less likely to form cohesive groups or cliques. However, the global clustering coefficient (0.0339) indicates a moderate level of clustering or connectivity in the network as a whole. Despite the low local clustering tendency of individual nodes, there are still observable clusters or communities within the network, contributing to its overall cohesion.

5 Discussion and Interpretation of Results

Dynamic data, including traffic flow, road capacity, and real-time incident reports, can significantly improve analytics. This data helps understand how the network responds to different conditions over time, including peak traffic times, seasonal changes, and disruptions due to construction or accidents. Real-time traffic data can come from traffic monitoring systems, vehicle GPS data, and mobile traffic applications. In addition, government transport agencies and international organizations such as UNECE can provide access to comprehensive data sets on road capacity and planned infrastructure development.

5.1 Implications

Infrastructure Planning and Optimization: The analysis reveals critical nodes and paths within the European road network, which can inform infrastructure development, optimization, and maintenance strategies. By identifying cities with high centrality measures, planners can target improvements or expansions to enhance connectivity between cities in Europe.

Emergency and Logistics Management: Understanding the shortest paths, diameter, and density of the network has direct implications for logistics and emergency response planning. The identification of the longest paths and the network's diameter can help in optimizing routes for transportation and in planning for emergency evacuation or service delivery routes

5.2 Limitations

Static Network Representation: The report analyzes a static snapshot of the road network. Real-world networks are dynamic, with conditions changing due to construction, traffic variations, and natural events. The analysis might not capture these temporal variations, which could affect route efficiency and network centrality.

Scale and Detail of the Data: The granularity of the cities (nodes) and roads (edges) included in the dataset could affect the analysis. For instance, smaller roads or temporary routes might not be included, which could influence the accuracy of shortest-path calculations and centrality measures.

Geographical and Socioeconomic Factors: The analysis does not incorporate geographical constraints (e.g., mountains, rivers) or socioeconomic factors (e.g., urban vs. rural areas, economic importance of cities) that could significantly impact the network's functionality and importance

5.3 Further Study

Connectivity Analysis: Examine the connectivity of the network, identifying any disconnected components. This can reveal how seamlessly different regions are connected and identify potential infrastructure weaknesses.

Community Detection: Apply community detection algorithms to identify clusters of cities that are more closely interconnected. This can highlight regional groupings or zones with stronger internal connectivity, potentially aligning with geopolitical or economic boundaries.

Path Analysis: Investigate the shortest paths between various city pairs to understand the efficiency of travel within the network. Analyzing the distribution of path lengths can provide insights into the network's overall navigability.

Network Robustness and Resilience: Simulate the removal of certain nodes or edges (cities or roads) to study the impact on network connectivity and robustness. This can help identify critical infrastructure whose failure could significantly disrupt the network.

Geospatial Analysis: Integrate geographical information to analyze the network's spatial structure. This could involve studying the relationship between road lengths and geographical distances, identifying potential areas for infrastructure improvement.

Comparative Analysis with Other Transportation Networks: If data is available, compare the E-road network with other transportation networks (e.g., rail, air) to assess complementarity, redundancy, and efficiency.

Reference

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A., Jaitly, N., ... & Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition. IEEE Signal Processing Magazine, 29(6), 82-97.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural Networks, 61, 85-117.

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2016). Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572.

United Nations Economic Commission for Europe (UNECE). (2021). European Agreement on Main International Traffic Arteries (AGR).

Dimitriou, H., & Stathopoulos, A. (2013). Towards a sustainable European transportation system: Considerations on the development of the Trans-European Transport Network (TEN-T). Transport Policy, 29, 106-114.

Vickerman, R. (2002). The role of the Trans-European transport networks in the development of a sustainable and competitive transport system. Journal of Transport Geography, 10(4), 299-308.

Schippl, J., Gudmundsson, H., Sørensen, C. H., Anderton, K., Brand, R., Leiren, M. D., & Reichenbach, M. (2013). Futures of European transportation: Visions and scenarios for 2050. Transportation Research Procedia, 4, 264-275.

Bruna, J., Zaremba, W., Szlam, A., & LeCun, Y. (2013). Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203.

Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. arXiv preprint arXiv:1710.10903.

Bello, I., Pham, H., Le, Q. V., Norouzi, M., & Bengio, S. (2016). Neural combinatorial optimization with reinforcement learning. arXiv preprint arXiv:1611.09940.

Khalil, E., Dai, H., Zhang, Y., Dilkina, B., & Song, L. (2017). Learning combinatorial optimization algorithms over graphs. arXiv preprint arXiv:1704.01665.

Li, Y., Tarlow, D., Brockschmidt, M., & Zemel, R. (2018). Gated graph sequence neural networks. arXiv preprint arXiv:1511.05493.

Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. Neural Information Processing Systems (NIPS).

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS).

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research (JMLR).

Mostafa, H., & Wang, X. (2019). Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. International Conference on Machine Learning (ICML).

Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., & Bronstein, M. M. (2017). Geometric deep learning on graphs and manifolds using mixture model CNNs. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K., & Jegelka, S. (2018). Representation learning on graphs with jumping knowledge networks. International Conference on Machine Learning (ICML).

Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. Advances in Neural Information Processing Systems (NIPS).

Alvarez-Hamelin, J.I., Dall'Asta, L., Barrat, A., & Vespignani, A. (2005). K-core decomposition: a tool for the visualization of large scale networks. Advances in Neural Information Processing Systems.

Newman, M. E. J. (2005). A measure of betweenness centrality based on random walks. Social Networks.

Borgatti, S. P. (2005). Centrality and network flow. Social Networks.

Li, Y., Chen, Y., Wang, Y., & Zhang, Z. (2018). Deep learning on graphs: A survey. IEEE Transactions on Knowledge and Data Engineering.

Lee, J. B., Rossi, R. A., Kim, S., Ahmed, N. K., & Koh, E. (2019). Attention models in graphs: A survey. ACM Transactions on Knowledge Discovery from Data (TKDD).

Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of ‘small-world’ networks. Nature, 393(6684), 440-442.

Latora, V., & Marchiori, M. (2001). Efficient behavior of small-world networks. Physical Review Letters, 87(19), 198701.

Meunier, D., Lambiotte, R., & Bullmore, E. T. (2010). Modular and hierarchically modular organization of brain networks. Frontiers in Neuroscience, 4, 200.

Clune, J., Mouret, J.-B., & Lipson, H. (2013). The evolutionary origins of modularity. Proceedings of the Royal Society B: Biological Sciences, 280(1755), 20122863.

Lipson, H. (2007). Principles of modularity, regularity, and hierarchy for scalable systems. Journal of Biological Physics and Chemistry, 7(4), 125-128.

Liao, Q., Li, Y., Cheng, Z., & Pecht, M. (2017). Use of a small-world networked model in a discrete manufacturing system to demonstrate the effects of connectivity on the resilience to perturbations. IEEE Transactions on Industrial Informatics, 13(5), 2537-2545.

Bullmore, E., & Sporns, O. (2009). Complex brain networks: graph theoretical analysis of structural and functional systems. Nature Reviews Neuroscience, 10(3), 186-198.

Appendix

Python code used for analysis

```
# Import Library
import networkx as nx
from networkx.algorithms import bipartite
import networkx.algorithms.community as nx_comm
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import statistics as st

# Load the dataset
# Open the file, skipping the first two lines
with open('datasets/inf-euroroad.edges', 'r') as file:
    # Skip the first two lines
    next(file)
    next(file)

    # Create the graph from the remaining lines
    graph = nx.read_edgelist(file, create_using=nx.Graph())


# Draw the graph
plt.figure(figsize=(12, 9))
nx.draw(graph, with_labels=True,node_size=120, font_size=6,
node_color='orange')
plt.savefig('euroroad.png')
plt.show()

# Compute the number of Nodes
node_count = graph.number_of_nodes()
print(f"Number of Nodes: {node_count}")

# Compute the number of Edges
edge_count = graph.number_of_edges()
print(f"Number of Edges: {edge_count}")

# Check for Acyclic
is_acyclic = nx.is_directed_acyclic_graph(graph)
print("Is the graph acyclic?:", is_acyclic)
```

```
# Check for the number of cycles
cycle_count = sum(1 for cycle in nx.cycle_basis(graph))
print("Number of cycles:", cycle_count)

# Check for the number of triangles
triangle_count = sum(1 for _ in nx.enumerate_all_cliques(graph) if len(_)
== 3)
print("Number of triangles:", triangle_count)

# Path to the text file containing city names
file_path = 'datasets/ent.subelj_euroroad_euroroad.city.name'

# Reading the city names from the file
with open(file_path, 'r') as file:
    cities = [line.strip() for line in file.readlines()]

# Creating a dictionary to map node numbers to city names
# Assuming node numbers start at 1 and increment by 1 for each city
node_to_city = {node_number: city for node_number, city in
enumerate(cities, start=1)}

#diameter
# Find all nodes connected to target node
# This includes direct neighbors
connected_to_target = set(nx.single_source_shortest_path_length(graph,
target).keys())

# Find all nodes in the graph
all_nodes = set(graph.nodes())

# Find nodes not connected to target node
not_connected_to_target = all_nodes - connected_to_target

# Printing nodes not connected to target node
print("Nodes not connected to node target:", not_connected_to_target)

# Remove isolated nodes which not connected to the target node
target_city = graph.copy()
target_city.remove_nodes_from(not_connected_to_target)

# Calculate diameter
d = nx.diameter(target_city)
print("Diameter: ", d)

density = nx.density(graph)
```

```
print(f"Density: {density}")

# Compute the degree of each node
degree = graph.degree()
print("The degrees of every node in euro road network are\n", degree)

# Compute mean degree
# Compute the mean degrees
mean_degree = sum(dict(degree).values()) / node_count

# Print out the mean degrees
print("The mean degree is ", mean_degree)

# Obtain the unique values of the degrees, and the counts of each degree
value
degree_unique = np.unique(degree_sequence, return_counts=True)
print("Counts each degree of unique values\n", degree_unique)

# Draw K-core Graph
kcore = nx.k_core(graph)
nx.draw(kcore, with_labels=True, node_size=120, font_size=6,
node_color='cyan')
plt.show()

#list out the core degree that each node belongs to
core_values = list(set(core_numbers.values()))
print("Core Degree: ", core_values)

# Count the number of nodes for each core value

# Initialize a dictionary to store the count of nodes for each core value
core_count = {}

# Count the number of nodes for each core value
for node, core_value in core_numbers.items():
    if core_value in core_count:
        core_count[core_value] += 1
    else:
        core_count[core_value] = 1

# Print the number of nodes for each core value
for core_value, count in core_count.items():
    print(f"Number of nodes for core value {core_value}: {count}")

# Closeness centrality
```

```
closeness = nx.closeness_centrality(graph)

# Sorting the closeness centrality values from largest to smallest
sorted_closeness = sorted(closeness.items(), key=lambda x: x[1],
reverse=True)

print(f"Closeness\n{sorted_closeness}")

# Printing the top 30 closeness centrality values
print("The top 30 Closeness Centrality")
for node, centrality in sorted_closeness[:30]:
    city = node_to_city[int(node)]
    print(f"Node: {node}, City:{city}, Closeness Centrality:
{centrality}")

# Betweenness Centrality
betweenness = dict(nx.betweenness_centrality(graph))

# Sorting the betweenness centrality values from largest to smallest
sorted_betweenness = sorted(betweenness.items(), key=lambda x: x[1],
reverse=True)

print(f"The betweenness centrality of each node\n{sorted_betweenness}")

# Printing the top 30 betweenness centrality values
print("The top 30 Betweenness Centrality")
for node, centrality in sorted_betweenness[:30]:
    city = node_to_city[int(node)]
    print(f"Node: {node}, City:{city}, betweenness Centrality:
{centrality}")

# Compute the local clustering coefficient of every node
clustering_coefficients = nx.clustering(graph, nodes=None, weight=None)

# Sorting the clustering coefficient values from largest to smallest
sorted_cluster_coef = sorted(clustering_coefficients.items(), key=lambda
x: x[1], reverse=True)

print(f"Clustering coefficient\n{sorted_cluster_coef}")

# Printing the top 30 clustering coefficient values
print("Top 30 clustering coefficient values")
for node, coef in sorted_cluster_coef[:30]:
    city = node_to_city[int(node)]
    print(f"Node: {node}, City:{city}, Clustering Coefficient: {coef}")
```

```
# Calculate average local clustering coefficient and global clustering
coefficient
average_local_clustering_coefficient =
st.mean(nx.clustering(graph).values())
global_clustering_coefficient = nx.transitivity(graph)

# Print the calculated coefficients
print("Average Local Clustering Coefficient:",
average_local_clustering_coefficient)
print("Global Clustering Coefficient:", global_clustering_coefficient)
```