‘Car’ Class represents individual cars in the simulation. Each car has a current location, a destination, and a path which is initially empty.

Attributes:

- current_location: The current node where the car is located.
- destination: The node where the car is headed.
- path: The sequence of nodes that the car will follow to reach its destination.

‘TrafficSimulation’ Class manages the traffic simulation on a given graph.

Attributes:

- G: A networkx.Graph object representing the road network.
- cars: A list of Car objects participating in the simulation.

Methods:

- simulate: The simulation runs for a specified number of steps and in each step, the move_cars method is called to move each car along its path.
- move_cars: For each car, if it doesn't have a path or has reached its destination, the simulation finds a new path using the shortest path algorithm (nx.shortest_path), considering travel_time as the weight. If a path is available, the car moves to the next node in its path (the first node in the path list is popped and set as the car's new current location).

- print_car_counts: After each simulation step, the script prints the number of cars at each node. This is done by counting how many cars have their current_location set to each node.

‘TrafficSimulator’ Class does the same as before but this class was created to help us visualize without mixing things up with the previous simulation.

Attributes:

- G: A NetworkX graph representing the road network. For our case, Adalbertstraße 58, Berlin, Germany.
- cars: A list of Car objects that participate in the simulation.

Key Methods:

- _create_cars: Initializes the specified number of Car objects with random starting and destination nodes.
- move_cars: Moves each car along its path for a single simulation step. It calculates the shortest path for each car using Dijkstra's algorithm, accounting for 'travel_time' as the path weight.
- run_simulation: Executes the traffic simulation for a given number of steps, calling move_cars in each step.
- edge_car_counts: Counts the number of cars on each edge of the graph at the end of the simulation.

The simulation initializes a specified number (num_cars) of cars on the graph G. For each simulation step, each car either continues along its predetermined path or, if it doesn't have one,

computes the shortest path to its destination. It takes traffic jams into account by limiting car movements. If more than 5 cars are trying to move onto the same edge, the cars do not advance, simulating a traffic jam. After running the simulation, the plot_network_with_traffic function visualizes the road network. It uses the car_counts data to adjust edge colors and widths on the graph based on the number of cars on each edge. The edges are colored and widened proportionally to the number of cars, and car counts are annotated on the edges. From the observations, the larger the number of cars on the graph, the wider the edges which suggests busier traffic. The script is set to run the simulation on a specific area in Berlin, centered around 'Adalbertstraße 58'. The road network is loaded from OpenStreetMap, and 'travel_time' is calculated for each edge based on speed limits.

b.

Firstly, the use of a graph to represent the road network is a realistic approach, where intersections are nodes and roads are edges. This setup reflects the actual layout and connectivity of urban road systems. Secondly, cars in the simulation use the shortest path to reach their destinations, akin to real-world navigation systems that often route vehicles based on the shortest or fastest paths.

There are also traffic jams. The model includes a basic form of traffic congestion simulation. When more than a certain number of cars (5 in this case) attempt to use the same road segment (edge), it leads to a traffic jam, causing a delay. This reflects real-world scenarios where roads can become congested when they are heavily used.

Additionally, the simulation considers different travel times for different road segments, calculated based on the speed limits. This variability in travel time for different road segments is a common aspect of real-world driving conditions.

While the simulation captures these aspects, it also has several limitations in its ability to fully emulate real-world traffic:

- Uniform Vehicle Behavior: It treats all cars as identical units, ignoring variations in vehicle size, speed capabilities, and driving behaviors.
- Static Traffic Conditions: The simulation does not account for dynamic changes in traffic conditions, such as fluctuations in traffic volume during different times of the day.
- Lack of Traffic Control Features: Real-world traffic is influenced by traffic lights, stop signs, roundabouts, and other control measures, which are not included in the simulation.
- Absence of Human Factors: Factors like driver behavior, decision-making, and errors are not considered.
- No Environmental Factors: The model does not take into account environmental factors like weather conditions, road closures, or accidents.
- Simplified Congestion Model: The traffic jam model is quite basic, using a fixed threshold for congestion without considering the capacity or type of the road.

The simulation effectively demonstrates basic principles of route planning, road network structures, and elementary congestion scenarios. However, for more accurate and realistic traffic modeling, additional factors and complexities would need to be incorporated.

c.

Simplifications:

- All cars in the simulation are treated identically, without differences in size, speed, type, or driving behavior.
- The road network is static and does not account for dynamic changes like road closures, construction, or temporary traffic re-routing.
- Traffic congestion is modeled simply by an arbitrary threshold (5 cars on an edge). This does not reflect the complex nature of traffic jams, which can be influenced by various factors like road capacity, traffic signals, and driving behaviors.
- The model does not include traffic control elements like traffic lights, stop signs, or roundabouts, which significantly impact traffic flow and congestion.
- The travel times are calculated once based on speed limits and do not dynamically change with traffic conditions.
- External factors like weather conditions, accidents, or road conditions are not considered.

Limitations:

- Cars do not adjust their routes in response to changing traffic conditions, which is a common behavior in real-world traffic as drivers seek to avoid congestion.
- The model lacks the sophistication to simulate nuanced traffic flow dynamics, such as the reaction to a sudden slowdown, merging lanes, or the impact of traffic signals.
- For large and complex road networks, the simulation might face performance issues, and the simplistic congestion model may not scale well.

- The model does not incorporate human factors like driver behavior, decision-making under uncertainty, or errors.
- The simulation only includes cars, omitting other forms of transportation like buses, bikes, or pedestrians that also impact traffic dynamics.
- The model does not integrate real-time traffic data, which can significantly affect traffic simulation accuracy and applicability for predictive purposes.

While the simulation provides valuable insights into basic traffic patterns and network usage, its simplifications and limitations mean it is better suited for theoretical analysis and educational purposes rather than detailed, real-world traffic planning or prediction. For more accurate traffic modeling, a more sophisticated approach should be used.

d.

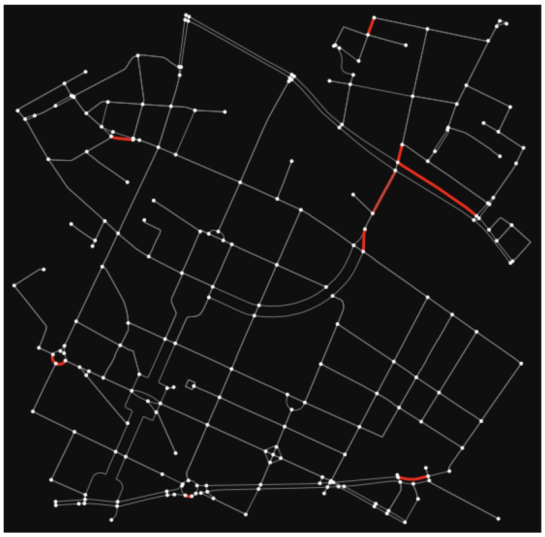
The current model determines congestion based on a simple, arbitrary threshold: if more than 5 cars attempt to move onto the same edge (road segment) at the same time, it's considered congested. This approach, while straightforward, doesn't accurately represent the complex nature of traffic congestion. The improvements could be:

- Using road-specific attributes like width, number of lanes, and typical traffic capacity to set dynamic congestion thresholds.
- Time-of-day factors can also be considered, using peak or off-peak traffic conditions.
- Rather than a binary congested/non-congested state, we can implement a gradient approach where the speed of cars gradually decreases as traffic density increases.

- If available, we can integrate real-time traffic data to adjust the simulation parameters and more accurately reflect current road conditions.
- We can utilize more sophisticated traffic simulation models that consider various vehicle behaviors, different types of vehicles, and more complex driver behavior models.
- We can also modify the speed of the cars. We would create a function within the move_cars method to adjust the speed of cars based on the current traffic density on their path.
    - This could involve reducing the speed as the number of cars on an edge approaches its congestion threshold.

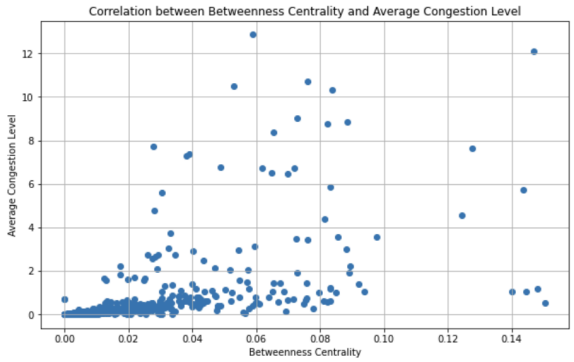
By implementing these changes, the model can provide more nuanced insights into how congestion affects travel times and vehicle flow. The results from such a model could inform traffic management decisions, like adjusting traffic signal timings, implementing congestion charges, or planning road works.

a.

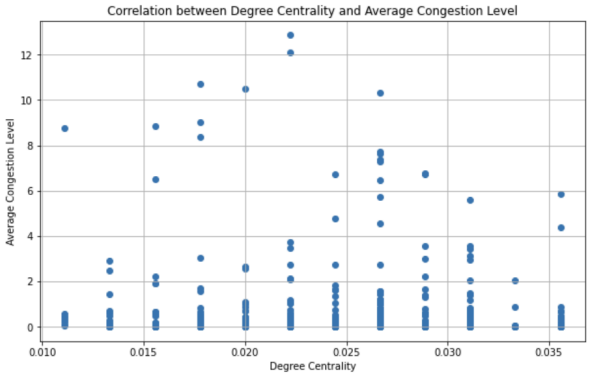


The red highlighted roads get more congested than the other roads on average after running the simulation 100 times. I used a loop to run the simulation multiple times and stored the congestion data. Then from the data, I found the edges with the highest average congestions. However, I did only a basic analysis. To go deeper, we could consider more statistical methods. You can see more of the codes in the Python notebook I provided.

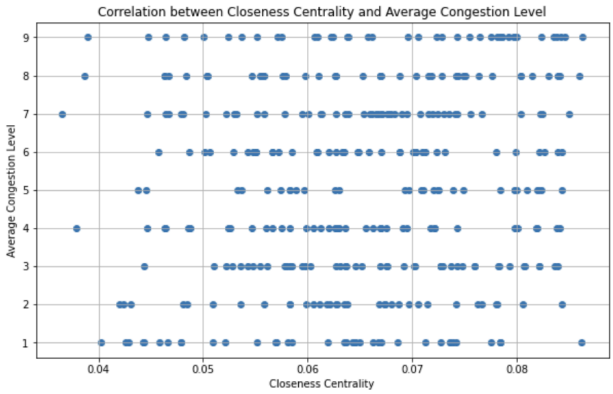
b. The network metric that had the highest correlation with the average congestion was the Betweenness centrality metric. However, the correlation coefficient was only 0.57. The lack of strong correlation could be because of several reasons such as the network characteristics of our network, the parameters, or the behavioral factors we talked about previously. The way our simulation is built can be a reason behind the low correlation. We could improve it by modifying our simulation to include more features that we can find in real life.



Correlation coefficient: 0.5670916358754646

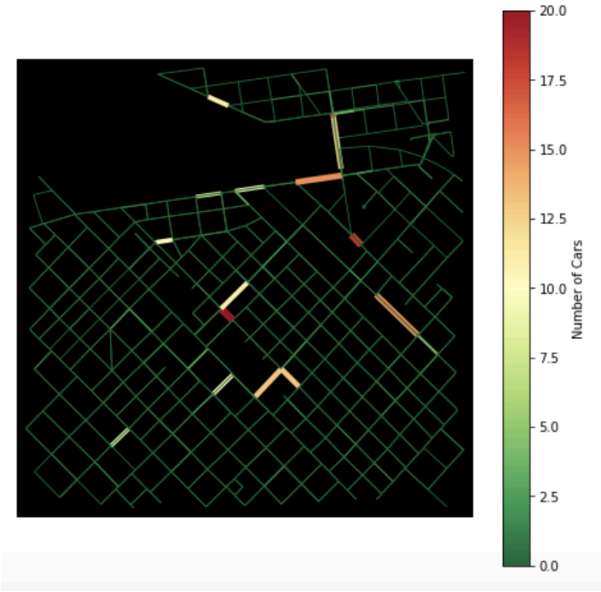


Correlation coefficient: -0.016697344915595654



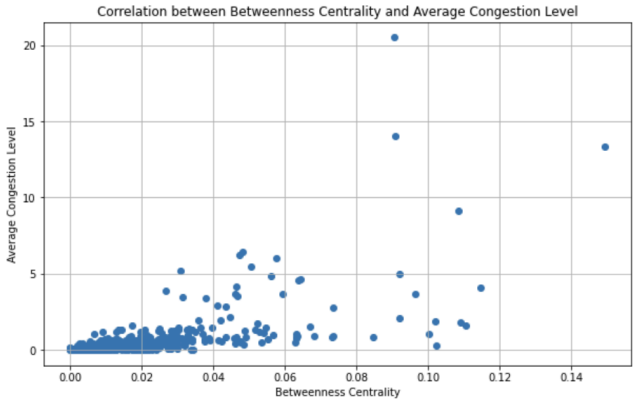
Correlation coefficient: 0.10101768480743763

c.



The graph above shows the simulation result for Esmeralda 920, Buenos Aires, Argentina, which is the place we live in.

The graph below shows the correlation between Betweenness Centrality and Average Congestion Level of 100 simulation results in Esmeralda 920, Buenos Aires, Argentina.




```
Correlation coefficient: 0.648928124716542
```

The Average Congestion level of the simulation in Buenos Aires shows more correlation with respect to the Betweenness Centrality. Although the correlation coefficient is higher this time, it is still lower than what we need to predict where congestion can occur. If the coefficient is at least 0.9 or close, we can say that the metric is a good predictor of congestion.