Analysis of Order Delivery Route using Graph Theory

Indhu Sri Krishnaraj
MS in Artificial Intelligence

1. Introduction

In recent years, we witnessed a phenomenal rise in the e-commerce sector due to technological improvements. People are more interested in shopping online these days since technology has made it so much simpler. Enterprises operating in the electronic commerce sector disperse their merchandise to an extensive global clientele. After orders are received, goods are picked up from warehouses and delivered to customer locations. The items are stored in warehouses owned by the companies. But the process of order pickup from warehouse to delivery to customer must be optimized [3], [4]. To study and gain some important insights and solutions to this problem, we analyze the routing network using graph theory concepts.

2. Objectives

The objective of this project is to determine the most efficient path for delivering products for a delivery vehicle by analyzing the order delivery network. The following is a list of the project's primary goals:

1. Using visualization on the network of cities for viewing their connectedness.
2. Implementing and evaluating important graph theory concepts such as degree measure, centrality measure of cities, etc.
3. Finding the optimal or the shortest route for deliveries in different cities.
4. To apply additional graph algorithms, like community detection, to the network of cities.

In doing so, we utilize the data to apply some of the most practical graph theory algorithms and offer some insightful findings on the order delivery network and also find the optimal route of delivering products between cities.

3. Methodology

To analyze this problem and carry out the project, a dataset is needed. Gathering a city network data will be the first step. The logical steps that for the implementation process are depicted in Fig. 1 and the next section shows the implementation.

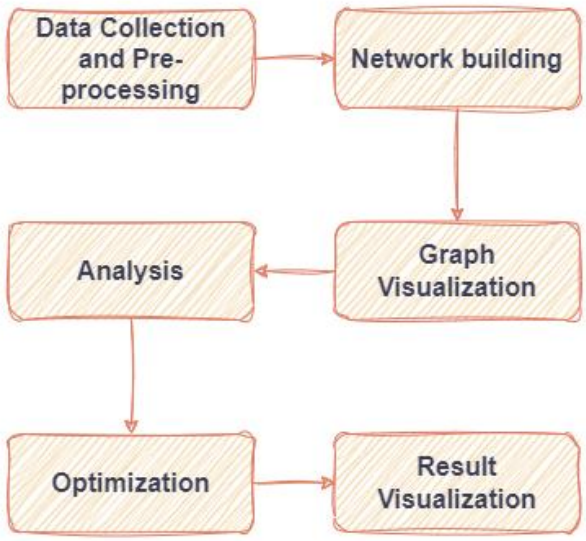

Fig 1: Implementation Design

The initial step of the project is to collect the order data and city network data and then to pre-process the data for further analysis. Subsequently, we create a network out of the data by depicting the cities as nodes and the paths connecting them as edges. The weights of the edges will be represented by the distance between the cities. The network will then be prepared for visualization. The visualization, will offer a clear view of every node and the connections or edges between them. The visualization can be carried out using Python Libraries [1]. Next, we perform various analyses on the graph to gain insights into the delivery network. Some possible analyses include degree analysis [4] for identifying the cities where multiple routes originate from, centrality measure [4] for identifying the most important nodes in the network and the most important – finding the shortest path to deliver all the orders. Dijkstra's Algorithm will find the shortest distance route between a given starting city to the destination city [2]. The community detection problem can be used to determine the cities in the network that are closely related. Lastly, a visualization of the analysis' findings is possible. Python programming language will be used for implementing the project by utilizing the most robust libraries of Python for graph network such as Networkx [1][8] and Pyvis [5].

4. Implementation

4.1 Dataset

The dataset used in this project consists of the network of cities in Turkey [10]. This data shows the distances between the neighboring cities. The distance is calculated by taking the Euclidean distances between the centers of the cities. The dataset includes 81 cities and 398 total links among the cities. This city network will be used in our study of route optimization for delivery network.

4.2 Network Visualization

The graph shown in Fig. 2 gives the outline of the network of Turkey where the nodes represent the cities while the edges represent the connections between them.

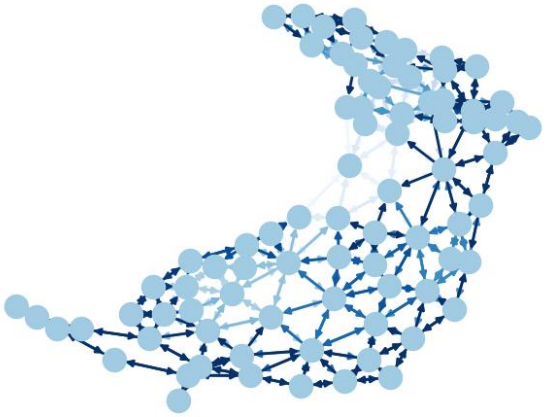

Fig 2: Network Outline

The graph shown in Fig. 3 gives the names of all the cities along with the connections.

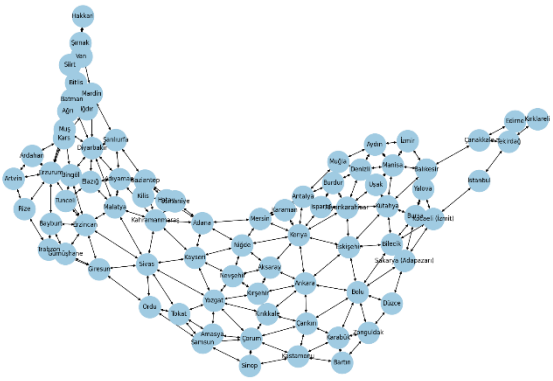

Fig 3: Network Visualization with city names

This graph is also visualized outside the python code environment where it can accessible to anyone with the link. The tool used for this implementation is Pyvis [5] which is a network visualization tool for interactive graph visualization. This network exports itself into an HTML page that can be rendered in any browsers. The visualization of the network using Pyvis is shown in Fig. 4 and Fig. 5.

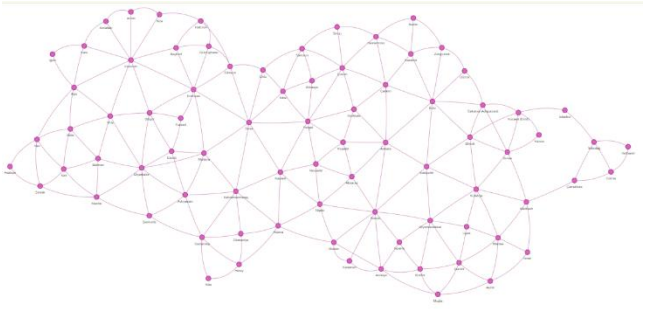

Fig 4: PyVis Network Visualization

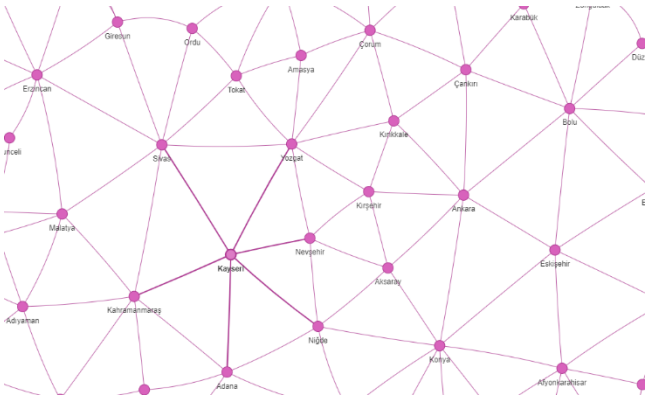

Fig 5: Clear View of PyVis graph by zooming in browser

4.3 Network Analysis

A. Degree Analysis:

The degree of a graph is the number of neighbors that a city has. In the problem of order delivery route optimization, the cities that have higher degrees indicate more interconnected connections. Routing through these cities will lead to more efficient routes for delivery. This is because these cities may have highest customers and the time for delivery might get reduced due to better infrastructure and roads. The degree analysis of the Turkey network is shown below:

```
Top 10 highest degree nodes:
1: ('Konya', 18)
2: ('Erzurum', 18)
3: ('Diyarbakır', 16)
4: ('Yozgat', 16)
5: ('Bolu', 16)
6: ('Erzincan', 16)
7: ('Sivas', 16)
8: ('Kahramanmaraş', 14)
9: ('Afyonkarahisar', 14)
10: ('Kütahya', 14)
Average degree: 9.82716049382716
```

```
Highest degree: 18

Lowest degree: 4
```

It can be seen that Konya and Erzurum are the cities which have highest degree of 18 neighboring cities.

B. In-Degree Centrality Analysis:

In-degree centrality measures the number of incoming connections [6] a city has. Cities that have high in-degree centrality could represent major distribution hubs. These cities may represent central locations where orders are delivered frequently. The In-Degree Centrality analysis of the Turkey network is shown below:

```
Top 10 highest in-degree nodes:
1: ('Konya', 0.1125)
2: ('Erzurum', 0.1125)
3: ('Diyarbakır', 0.1)
4: ('Yozgat', 0.1)
5: ('Bolu', 0.1)
6: ('Erzincan', 0.1)
7: ('Sivas', 0.1)
8: ('Kahramanmaraş',
0.08750000000000001)
9: ('Afyonkarahisar',
0.08750000000000001)
10: ('Kütahya', 0.08750000000000001)
Highest in-degree: 0.1125

Lowest in-degree: 0.025

Average in-degree: 0.06141975308641975
```

It is apparent that the cities with the greatest degrees of centrality are Erzurum and Konya.

C. Out-Degree Centrality Analysis:

Out-degree centrality measures the number of outgoing connections a city has. Cities that have high out-degree centrality could represent major transit points. These cities may represent important departure locations of goods. The Out-Degree Centrality analysis of the Turkey network is shown below:

```
Top 10 highest out-degree nodes:
1: ('Konya', 0.1125)
2: ('Erzurum', 0.1125)
3: ('Diyarbakır', 0.1)
4: ('Yozgat', 0.1)
5: ('Bolu', 0.1)
6: ('Erzincan', 0.1)
7: ('Sivas', 0.1)
8: ('Kahramanmaraş',
0.08750000000000001)
9: ('Afyonkarahisar',
0.08750000000000001)
10: ('Kütahya', 0.08750000000000001)
Highest out-degree: 0.1125

Lowest out-degree: 0.025

Average out-degree: 0.06141975308641975
```

As it can be noticed that the cities with the highest out-degree centrality are Konya and Erzurum.
By comparing the in-degree and out-degree centralities, it can be said that all the connections between the cities are bi-directional.

D. Betweenness Centrality Analysis:

An important measure of the network's connection is betweenness centrality. In a network, the cities with high betweenness centrality serve as critical intermediaries. The Betweenness Centrality analysis of the Turkey network is shown below:

```
Top 10 highest betweenness centrality
nodes:
1: ('Sivas', 0.3200772034282754)
2: ('Yozgat', 0.21153523221829842)
3: ('Konya', 0.20880683476884276)
4: ('Erzincan', 0.20643753289436134)
5: ('Erzurum', 0.15170575626809113)
6: ('Niğde', 0.1463273150664438)
7: ('Adana', 0.14309772978244725)
8: ('Eskişehir', 0.1418447551920609)
9: ('Kayseri', 0.13628096047700905)
10:                  ('Kahramanmaraş',
0.13334254490365474)
Highest betweenness centrality:
0.3200772034282754

Lowest betweenness centrality: 0.0

Average betweenness centrality:
0.04896858884200656
```

It can be seen that the city Sivas has highest betweenness centrality and is an important city in the network followed by Yozgat, Konya and Erzincan.

E. Closeness Centrality Analysis:

Closeness centrality is a measure of how close a city is to all other cities in the network. The cities with high closeness centrality are geographically central. These cities are well-connected to other cities. Including these cities in delivery routes can reduce travel distances and overall delivery times, and thus lead to more efficient routes. The Closeness Centrality analysis of the Turkey network is shown below:

```
Top 10 highest closeness centrality
nodes:
1: ('Kayseri', 0.2826855123674912)
2: ('Sivas', 0.2807017543859649)
3: ('Yozgat', 0.2787456445993031)
```

```
4: ('Niğde', 0.2739726027397260)
5: ('Kırıkkale', 0.2702702702702703)
6: ('Konya', 0.26755852842809363)
7: ('Kırşehir', 0.26666666666666666)
8: ('Adana', 0.264026402640264)
9: ('Ankara', 0.264026402640264)
10: ('Kahramanmaraş',
0.2631578947368421)
```

```
Highest closeness centrality:
0.2826855123674912

Lowest closeness centrality:
0.12558869701726844

Average closeness centrality:
0.21216705707475667
```

It can be seen that the cities Kayseri, Sivas, Yozgat and Nigde form important cities in closeness measure.

4.4 Community Detection

Community Detection is used to identify groups or clusters [7] within the Turkey network. The clusters are formed of cities that are densely connected to each other than to the nodes in other groups. This is helpful in identifying the cities with similar delivery requirements. By treating each community as a single unit, the delivery routes can be optimized within each community, thereby minimizing the travel distance and time. This technique also reduces the complexity of the problem by the means of breaking down the network based on communities and plan delivery strategies for each community effectively. 5 communities have been identified for the Turkey Delivery network which are shown in the figure Fig. 6.

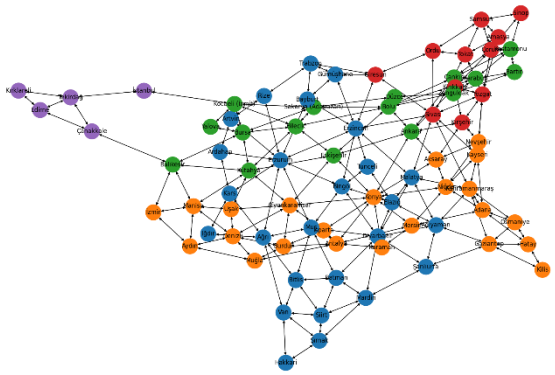

Fig 6: Community Detection

4.5 Routing Optimization

Routing Algorithms are used to find the shortest path routes between one city to another. The most fundamental routing algorithm is Dijkstra's algorithm. This algorithm iteratively explores all the nodes by maintaining temporary shortest distances. As it iterates through all the nodes, the distance values keep updating until minimum distance to the destination node is found or all the nodes have been explored. Let the source city of orders collection be Kilis and the destination city for orders delivery be Yalova. Dijkstra's algorithm is run on the network to find the shortest path and the corresponding distance between the source and destination cities. The shortest path between these cities is shown in figure Fig. 7.

```
Shortest path: ['Kilis', 'Gaziantep',
'Kahramanmaraş', 'Kayseri', 'Nevşehir',
'Kırşehir', 'Ankara', 'Bolu', 'Sakarya
(Adapazarı)', 'Kocaeli (İzmit)',
'Yalova']
```

```
Shortest path distance between Kilis and
Yalova: 1168
```

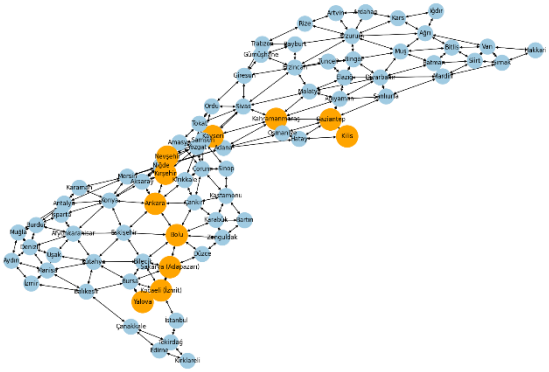

Fig 7: Dijkstra's Algorithm

5. Evaluation

In the evaluation process, we compare our routing algorithm implementation with a normal path finding algorithm. This algorithm is a typical Breadth First Search algorithm. The algorithm explores all the nodes that are neighbours to a node [9] and continues this process until the destination node is found. Let us consider the same example by having the source city of orders collection as Kilis and the destination city for orders delivery as Yalova. The BFS algorithm is run on this network and the result produced is shown below.

```
Path from Kilis to Yalova: Kilis -> Hatay
-> Adana -> Mersin -> Konya ->
Afyonkarahisar -> Kütahya -> Bursa ->
Yalova
```

```
Total distance of the path: 1322
```

The following observations can be made:

1. The number of cities in the path is higher in Dijkstra's path as compared with the typical BFS path. This shows that the BFS algorithm terminates as and when the destination city is found in the path without considering the weights whereas the

Dijkstra's algorithm explores each city to find the final path.

2. The distance of the path calculated by the BFS algorithm is 1322 which is higher than that calculated by Dijkstra's, which is 1168. Therefore, it can be concluded that the implemented algorithm is more efficient that the Brute force algorithm.

6. Risks

Certain parts of the data are not considered. A few features like delivery deadline, vehicle capacity, weight of products and number of vehicles available for delivery are crucial factors for a delivery problem. But these factors are not considered. The only agenda here is to find the optimal or the shortest route to deliver all the orders to the customers by optimizing the distance travelled. This might only be optimal in case of fuel efficiency.

7. Conclusion

This project was about analysing the network of Turkey for efficient route planning in an order delivery network. The network of Turkey cities was visualized along with analysing various network parameters such as degree and centrality measures which are very crucial for efficient delivery. Community detection algorithm was also implemented for analysing the network. Next, Dijkstra's algorithm was implemented to find the shortest path between a pair cities and thus optimize the delivery route. Future work can focus on integrating Dijkstra's algorithm with other network analysis techniques, such as community detection or centrality measures, to further optimize the delivery routes.

References

[1] A. Hagberg and D. Conway, 'Networkx: Network analysis with python', URL: https://networkx. github. io, 2020.

[2] P. S. Rani and S. M. Owais, 'Application of Graph Theory In Air-Transportation Network', *Strad Research*, no. 8, p. 9, 2021.

[3] A. D. Handayani, 'Dijkstra Algorithm in Transportation Systems at Kediri's Central Post Office'.

[4] A. Majeed and I. Rauf, 'Graph theory: A comprehensive survey about graph theory applications in computer science and social networks', *Inventions*, vol. 5, no. 1, p. 10, 2020.

[5] G. Perrone, J. Unpingco, and H.-M. Lu, 'Network visualizations with Pyvis and VisJS', arXiv [cs.SI]

[6] Z. A. Rachman, W. Maharani, and Adiwijaya, 'The analysis and implementation of degree centrality in weighted graph in Social Network Analysis', in 2013 International Conference of Information and Communication Technology (ICoICT), 2013, pp. 72–76.

[7] S. Fortunato and D. Hric, 'Community detection in networks: A user guide', Physics reports, vol. 659, pp. 1–44, 2016.

[8] Aric A. Hagberg, Daniel A. Schult and Pieter J. Swart, "Exploring network structure, dynamics, and function using NetworkX", in Proceedings of the 7th Python in Science Conference (SciPy2008), Gäel Varoquaux, Travis Vaught, and Jarrod Millman (Eds), (Pasadena, CA USA), pp. 11–15, Aug 2008

[9] R. Zhou and E. A. Hansen, 'Breadth-first heuristic search', Artificial Intelligence, vol. 170, no. 4–5, pp. 385–408, 2006.

[10] https://www.kaggle.com/datasets/ahmetomer/turkey-city-distances