Data Analysis of the Indian Railway Network

Supervisor: Dr. Abhijnan Chakraborty

Authors: Burouj Armgaan, Manthan Dalmia, Mohit Sharma, Harshit Bhardwaj

We highly recommend reading our blog instead of the report as some images are rendered out of place in the PDF format. Find it here: https://glamorous-snowdrop-fc2.notion.site/Data-Analysis-of-the-Indian-Railway-Network-1c06d44fe4944bd19ca4ea9b10aede15?pvs=4

1. Introduction

The Indian Railway network is one of the most extensive and complex railway systems in the world. It plays a crucial role in the transportation of passengers and goods across India. It is important to study the Indian Railway network as IRN is one of the largest employers in the world and the backbone of the Indian Economy. Indian Railway Network spans the entire country, connecting remote villages, bustling cities, and diverse landscapes. It provides a crucial mode of transportation, particularly in a country as geographically varied as India. Over the years, IRN has embraced technological advancements. The introduction of high-speed trains and electrification of tracks, but are there enough tracks/ trains running between the stations to meet the current requirements? Despite a lot of advancements, IRN still faces challenges such as congestion, punctuality, and the need for modernization. Many economists argue that India's current transportation network is insufficient for the rapidly growing economy. Challenges like congestion, traffic surpassing planned capacities between major cities, and overused railway tracks lead to reduced train speeds and lower freight capacity, elevating transportation costs and time. By conducting a comprehensive analysis of the railway network, we show how the network has changed over time and what additional contributions are essential to be made, such as the introduction of new routes to mitigate congestion and punctuality issues, facilitating the development of effective expansion strategies and improved railway budget planning in the future.

Our project is divided into three parts:

- Section 2: Data collection
- Section 3: Data analysis
- Section 4: Delay prediction

You'll find the code at this repository: https://github.com/Armagaan/irn-data-analysis

2. Data Collection

Data collection for this project was done in two phases. In the first phase, we need to get the complete latest Indian railway network. In the second phase, we collected the delay history for every train on each day of the last 1 year.

2.1. Train Info and Formation of Indian Railway network graph

1. A function `getTrainInfo` outputs the various information for a given train number in JSON format. The output looks like the format attached below.

2. Now, to collect information for every available train, we need a complete list of all train numbers, upon which `getTrainInfo` will be called iteratively, and the output for each call will be stored in `train_info.json`
3. The list of all train numbers was collected by scrapping the train list from https://www.prokerala.com/travel/indian-railway/trains/, and info was scraped from irctc.co.in

2.2. Train Delay History

1. A function `getTrainDelayOnDate` scraps https://runningstatus.in/status for a given train number and a given date and outputs the delay in every station on that day that train halts in a dictionary.
2. A function `getDelayForAllDates` calls the above function for every day in the past year, collects the whole train delay info, and outputs it in JSON format data.
3. The above function is called in all trains in our train numbers list.

3. Data Analysis

The Indian railway system is one of the largest railway networks in the world, with over 67,000 km of tracks and more than 7,000 stations. In Data analysis of IRN, we study the network structure and various network properties of IRN. We also study how IRN has changed over time by conducting a comparative analysis with respect to the [1], 2017 IRN data [4] (collected through data.gov.in) and 2023 dataset (collected through scraping the website [5] for every train in the list from irctc.co.in). The objective of the data analysis part is to study the various aspects of the Indian Railway Network to identify patterns, trends, and insights that can help improve the efficiency and safety of the railway system. In the following subsections, first, we construct the graph/network from the available dataset and thereafter calculate the various network properties, parameters and centrality measures.

3.1. Network Construction

In order to analyze and conduct network-related experiments, the first step is to create a graph of the Indian railway network based on the data available. We create the station-station graph as mentioned in [1], where each node of the graph (undirected) corresponds to a station, and an edge between two nodes represents a train existing between those two nodes (stations). Edge weights in the graph represent the number of trains running between the stations.

Note: Suppose train X starts from station A following routes B, C, D, and E. Then we connect station A with stations B, C, and D, i.e. all stations falling along its route and obtain a complete graph.

Once we have collected the dataset by scraping the data in JSON format, we convert this JSON data into a data-frame. If a train stops at 10 different stations, then for this train, we have 10 rows in our data-frame. In another case, if we have a dataset readily available (in the form of a data-frame), for example, IRN 2017 collected from data.gov.in, we do not need to have the conversion from JSON to data-frame. After conversion, the data frame looks like the below:

	trainNumber	trainName	stationFrom	stationTo	station_code	station_name	arrivalTime	departureTime	routeNumber	haltTime	distance	dayCount	stnSerialNumber	boardingDisabled
0	12303	POORVA EXPRESS	HWH	NDLS	HWH	HOWRAH JN	--	08:00	1.0	--	0.0	1.0	1.0	False
1	12303	POORVA EXPRESS	HWH	NDLS	BWN	BARDDHAMAN JN	09:05	09:08	1.0	03:00	95.0	1.0	2.0	False
2	12303	POORVA EXPRESS	HWH	NDLS	DGR	DURGAPUR	09:57	09:59	1.0	02:00	158.0	1.0	3.0	False
3	12303	POORVA EXPRESS	HWH	NDLS	ASN	ASANSOL JN	10:32	10:37	1.0	05:00	200.0	1.0	4.0	False
4	12303	POORVA EXPRESS	HWH	NDLS	CRJ	CHITTARANJAN	11:00	11:02	1.0	02:00	225.0	1.0	5.0	False

For each unique train in the data frame, we construct the network by connecting all those stations with each other lying along its route indicated by the station_name column in the above dataset. We followed [3] for the construction of a graph using the data frame. In all our subsequent subsections, we have used the results produced by [1] with respect to the IRN 2010 dataset.

3.2. Network Information

The network structure, such as the number of nodes, edges and edge weights, varies across the datasets obtained at different timestamps. In [1], the authors mentioned that the website (www.indianrail.gov.in) hosts information on 1072 express train routes and 3041 stations which are scheduled stops on at least one such train route. Whereas the dataset [4] has 7580 unique trains and 8147 stations, and our latest scraped dataset [5] has 3800 stations and 3370 unique trains, which is significantly less than the dataset published by [4] because of the unavailability of the schedules of many trains.

3.3. Comparative Analysis | Network Structure

This section emphasizes the network properties such as nodes, edges, average degree and average shortest path length. The average degree of the network indicates the average number of stations reachable from an arbitrary station via a single train. Indian Railway Network also exhibits small world properties as we can see the average shortest path length for each dataset variation (avg. number of edges separating two nodes in a network) is very small as compared to the network size. Also, our analysis shows that the average shortest path length reduced further in 2017, and the latest scraped data as well.

Properties	IRN 2010	IRN 2017	IRN Scraped (2023)
Nodes, N	3041	8147	3800
Edges, E	181,208	500,412	311,674
Avg. Degree	119.177	122.845	164.038
Avg. Shortest Path Length	2.53	7	3
Connected Components	NA	1.379	1.488

3.4. Comparative Analysis | Degree and Strength Distribution

The degree of a node/station is a measure of connectivity which represents the number of stations reachable from a given station via direct single train. The strength of a node is the weighted degree of a node, i.e., the sum of weights of edges adjacent to the particular node. In IRN, the strength of a station represents the total number of different journeys that can be taken from this station. In other words, the strength of a station is the measure of available transportation from this station which indicates the connectivity and amount of traffic flow.

The degree distribution p(k) of a network is the fraction of nodes with degree k. For a total of N nodes, if p nodes have degree k, then p(k) will be p/N, in this case. However, as shown below, degree distribution is often noisy; therefore, we move for cumulative degree and strength distributions.

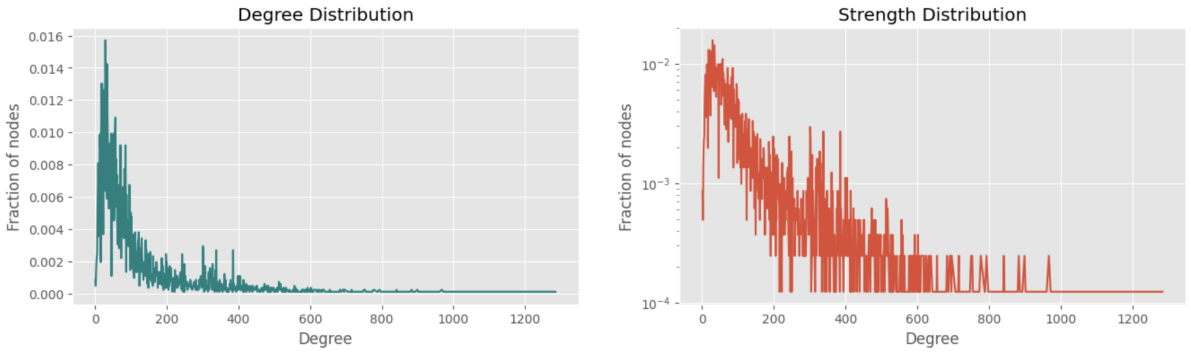

Fig.1. Degree and Strength Distribution

The cumulative degree distribution is given by:

$$P(k) = \sum_{i=k}^{\infty} p(i)$$

IRN's node, strength distribution and degree distribution both exhibit an exponential decaying nature. An exponentially decaying distribution means that there are many nodes with a low degree (few connections) and progressively fewer nodes with higher degrees (followed by most real-world networks). However, the degree distribution deviates from the exponential decaying nature for larger k (degrees), indicating that adding a link between stations involves introducing new train routes or new stations into existing routes. This process becomes more challenging and costly as the degree of a station increases.

3.4.1. IRN 2010

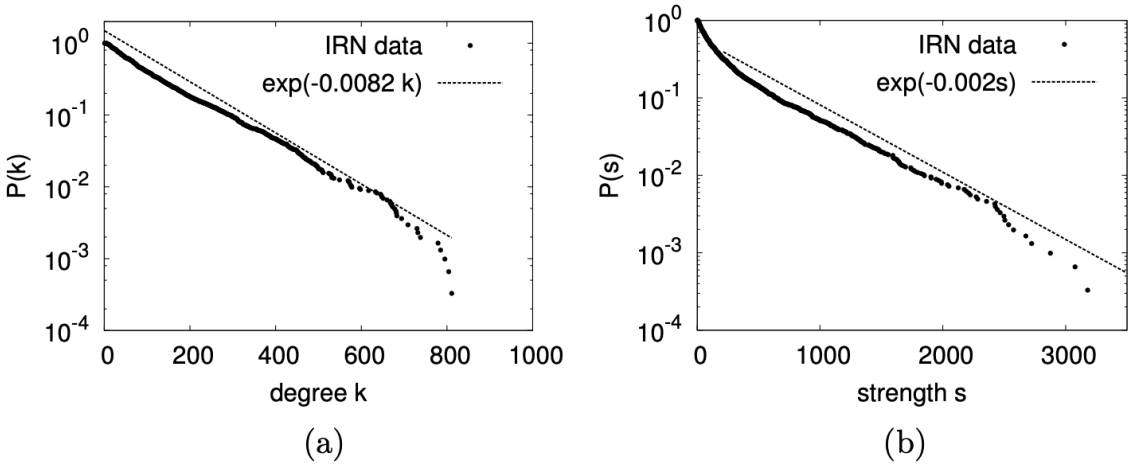

Fig.2. (a) Cumulative degree distribution of IRN. (b) Cumulative strength distribution (both in semi-log scale, along with exponential fits).

3.4.2. IRN 2017

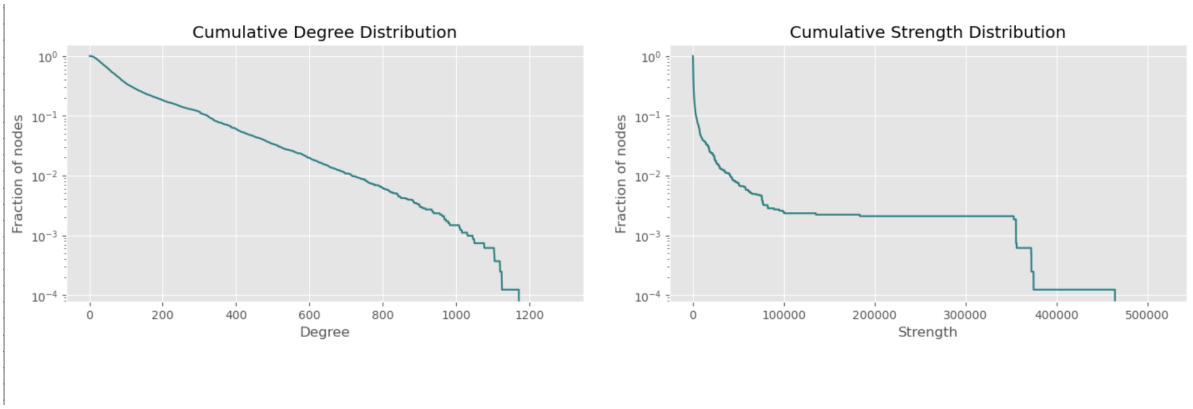

Fig.3. Cumulative Degree and Strength Distribution for IRN 2017

3.4.3. IRN 2023 (Scraped Dataset)

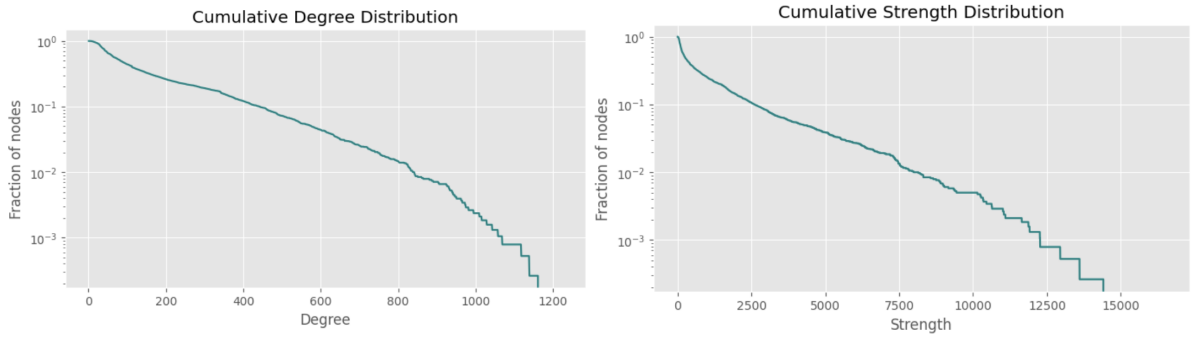

Fig.4. Cumulative Degree and Strength Distribution for IRN 2023 (Scraped Data)

We observe an improvement in deviation from the exponentially decaying nature of degree distribution, which implies that trains have been added more in number, which was a more challenging and expensive business earlier.

3.5. Distribution of Edge-Weights

The weights assigned to edges in the station–station network characterize the traffic flow in the railway network. The weight (wij) of an edge connecting two station nodes i and j reflects the count of train routes directly linking those stations. Consequently, areas with higher edge weights experience more frequent movement of passengers and freight. Examination of these edge weights reveals a considerable diversity in traffic flow within the Indian Railway Network (IRN). The cumulative distribution of edge weights follows an exponential pattern, represented as $P(w) \sim \exp(-\alpha w)$.

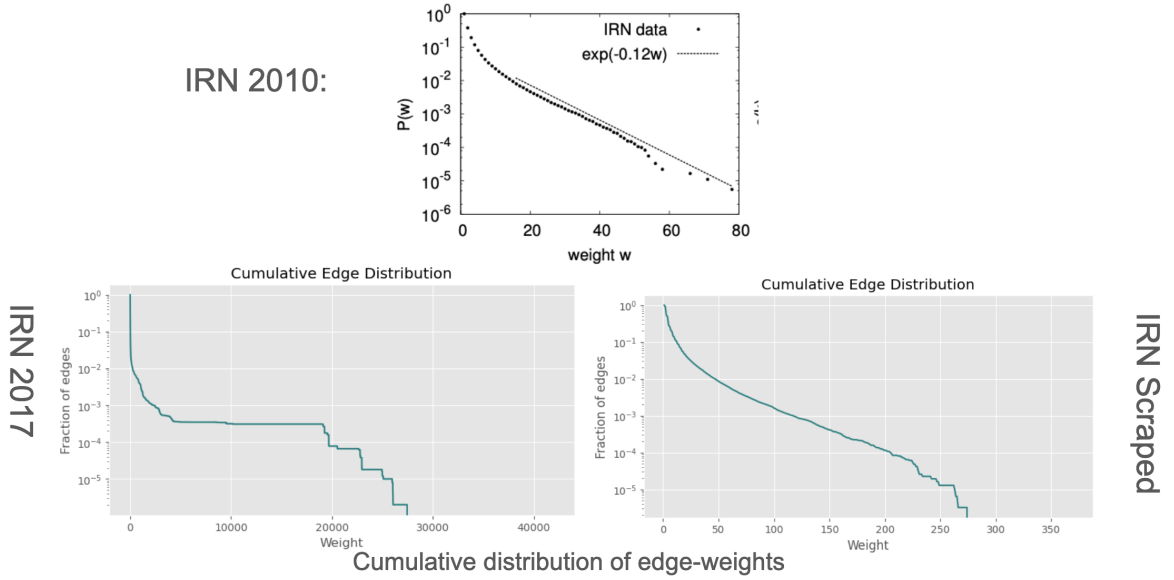

Fig.5. Comparison of Distribution of Edge Weights among IRN 2010, IRN 2017 and IRN Scraped (2023)

3.6. Strength-Degree Correlation

When we plot the correlation between degree k and average strength s(k) of nodes with degree k, we observe that s(k) increases rapidly with k and follows a power law behaviour. $s(k) \sim k^{\beta}$. The higher value of β, implies node-strength is strongly connected with node-degree and strength grows faster as compared to degrees. Note that $\beta = 1$ implies strength is proportional to degree.

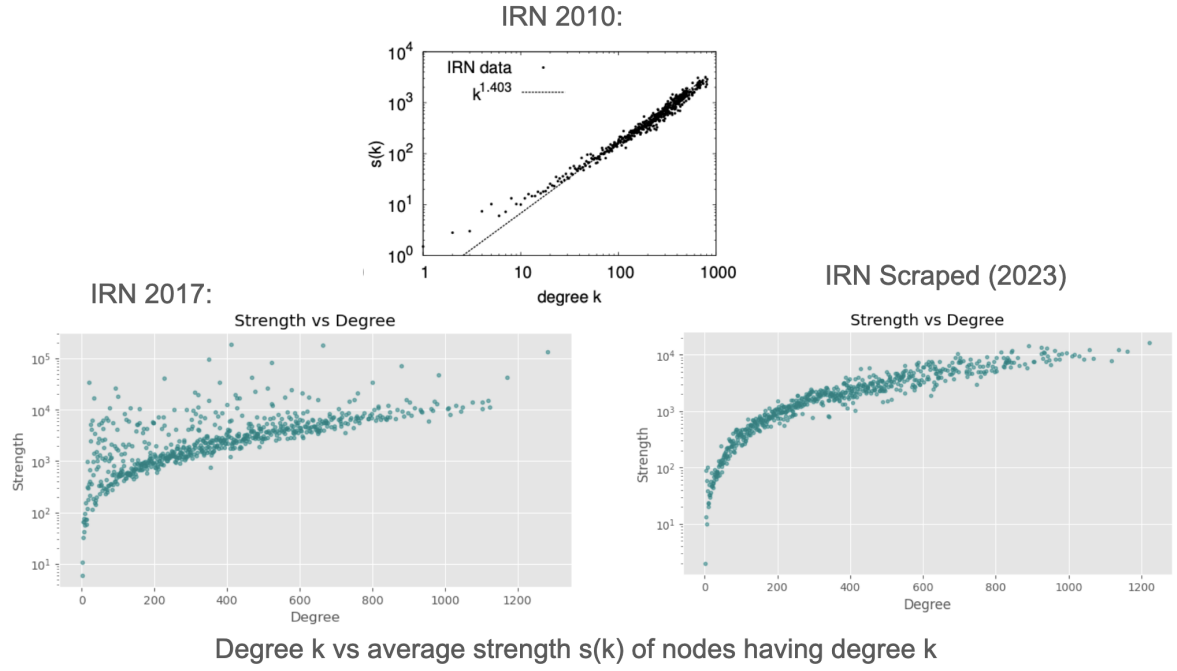

Fig.6. Comparison of Degree vs strength of nodes having degree k among IRN 2010, IRN 2017 and IRN Scraped (2023)

The power law behavior of nodes indicates that the strength (connectivity) of stations grows more rapidly than their degrees. Specifically, the introduction of new trains, i.e. increasing weights on edges (strengthening nodes), is more common as compared to the construction of new routes (increasing the degree of nodes).

The comparison in figure (Fig.5.) indicates that gradually with time, the degree of stations has also improved along with the strengths, and the relation is no longer linear. This implies that with the passage of time from 2010 till date, the introduction of new routes has also been more frequent.

3.7. Weight-Degree Correlation

Another way to characterize the strength and degree relationship is to examine the correlation of edge weight w_{ij}, between nodes i and j having degrees k_i and k_j. We observe that connections between nodes with high degrees exhibit elevated traffic values. Such routes in [1] are referred to as trunk routes.

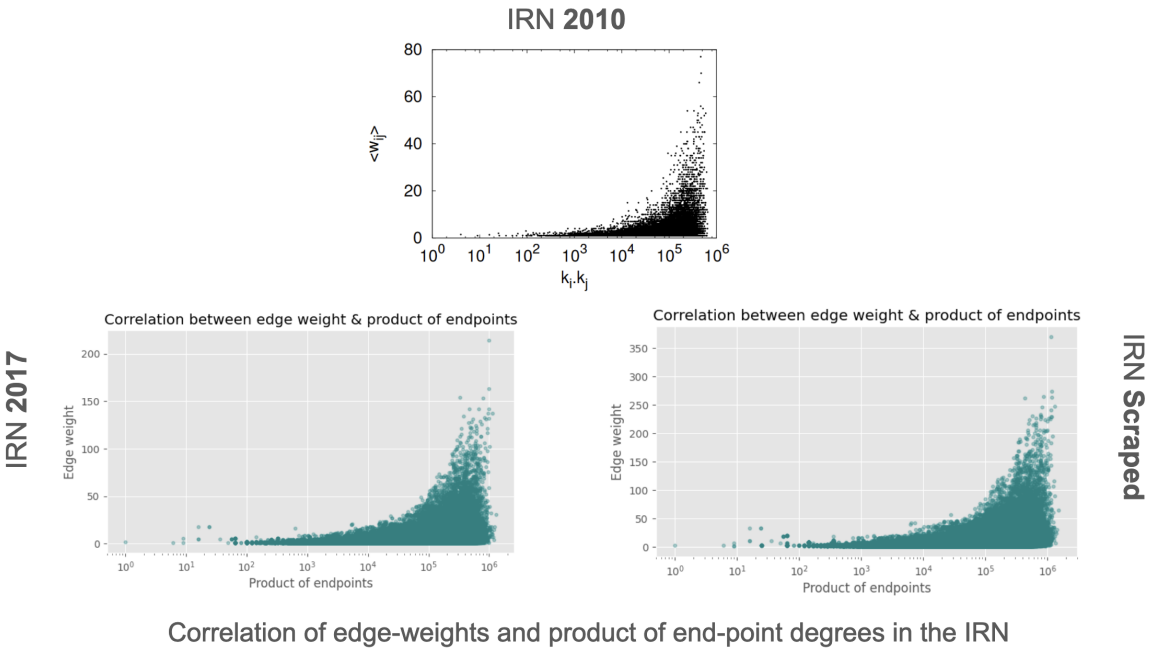

Fig.7. Comparison of Weight-Degree Correlation among IRN 2010, IRN 2017 and IRN Scraped (2023)

Also, one can observe that the traffic for the high-degree nodes, i.e. stations having high connectivity, has high values of traffic, which is even more increased significantly for the IRN 2017 and IRN Scraped dataset as compared to the 2010 dataset.

3.8. Degree-Degree Correlation

To examine the network architecture further, we consider the correlation among the degrees of neighboring nodes. This correlation is explored using the average nearest-neighbor degree, denoted as $k_{nn}(k)$, for nodes with a degree of k. The analysis of the 2010 dataset suggested that $k_{nn}(k)$ on average remains almost stable over varying degrees, which implies the “absence of major correlations among the nodes of different degrees”, meaning stations of different degrees tend to be not directly connected or in other words, bigger stations tend to be connected to bigger stations and smaller stations with smaller ones.

Factoring in the weights of edges, we obtain $k^{w}_{nn}(k)$[6], the average weighted nearest neighbor degree of nodes having degree k. This term represents the assortative nature of the graph while considering the edge weights, i.e. no. of trains between stations.

$$k^{w}_{nn,i} = \frac{1}{s_i}\sum_{j=1}^{N} a_{ij} w_{ij} k_j$$

This equation represents a local weighted average of the nearest neighbor degree according to the normalized weight of connecting edges, w_{ij}/s_i for node i where:

a_{ij} is the adjacency matrix representation of the graph, k_j is the degree of vertex j, w_{ij} is the weight of edge ij , and s_i is the strength of node i.

Plotting these variables with degree produced the following result on the 2010 network.

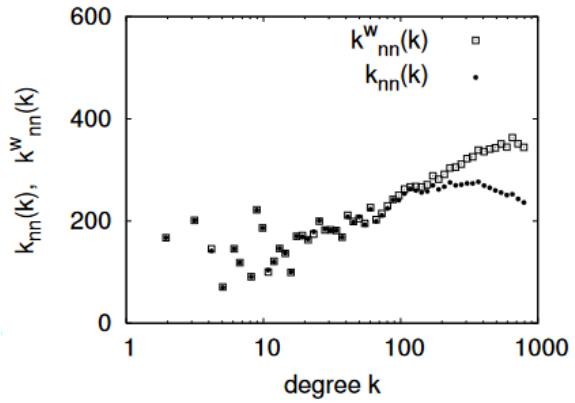

Fig.8. Avg. Degree of nearest neighbors knn (k) and average weighted degree of nearest neighbors kw(k) of nodes with degree k using logarithmic binning of edges (semi-log scale)

The experiment was repeated in 2017, and the latest scraped 2023 data gave the following results.

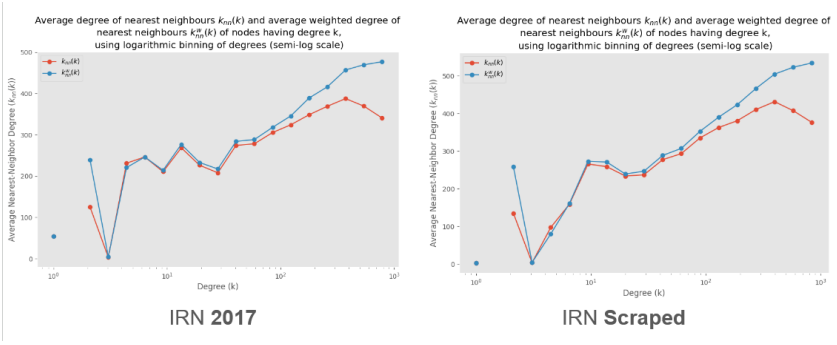

Fig.9. Avg. Degree of nearest neighbors knn (k) and average weighted degree of nearest neighbors kw(k) of nodes with degree k using logarithmic binning of edges (semi-log scale) for IRN 2017 and Scraped dataset.

The plots show that the average nearest neighbor degree, $k_{nn}(k)$, does not vary much, with degrees agreeing with the 2010 result. The weighted average nearest neighbor degree, $k^w_{nn}(k)$, shows a more pronounced effect, however. What this implies is that degrees themselves have gradually increased from 2010 to 2017 to 2023, but higher degree nodes are still only connected to higher degree nodes, and the number of trains between these nodes has also increased. We must also consider that the average degree in the 2010 dataset was 119.177, while in the 2017 dataset was 122.845, and in the 2023 dataset, it was 164.038

3.9. Centrality Measures

In this section, we will have a brief introduction about three centrality measures (degree centrality, betweenness centrality and closeness centrality)[8]. We also list the top 10 stations corresponding to different centrality measures (for illustration, the table below indicates the top stations based on different centrality measures for the scraped dataset 2023):

Stations	Degree Centrality		Stations	Betweenness Centrality		Stations	Closeness Centrality
ITARSI JN	0.32192682		VISAKHAPAT.	0.01816805		ITARSI JN	0.5923922
VADODARA JN	0.30560673		HOWRAH JN	0.01732744		VADODARA JN	0.5815552
AHMEDABAD	0.29955251		SEALDAH	0.01370336		AHMEDABAD	0.5805734
GHAZIABAD	0.29402474		VIJAYAWADA	0.01251222		NEW DELHI	0.5737059
MATHURA JN	0.28112661		MATHURA JN	0.01174923		HOWRAH JN	0.5731843
HOWRAH JN	0.278231113		GHAZIABAD	0.01168142		AGRA CANTT	0.5724904
AGRA CANTT	0.27428270		GUNTAKAL JN	0.01071396		MATHURA JN	0.5716253
ANAND JN	0.27059752		SURAT	0.01069807		GHAZIABAD	0.5714526
NEW DELHI	0.267175572		BHOPAL JN	0.01050095		ANAND JN	0.5704187
KALYAN JN	0.26559620		JHS	0.01028257		BHOPAL JN	0.5699889

Stations	Degree Centrality		Stations	Betweenness Centrality		Stations	Closeness Centrality

3.9.1. Degree Centrality

Degree centrality is a measure in network analysis that quantifies the importance of a node in a graph based on the number of edges (connections) it has. In simpler terms, it reflects how well-connected a node is to other nodes in a network. These nodes have the most one-hop connections to other nodes or Stations. These nodes can connect to the wider network in less number of hops.

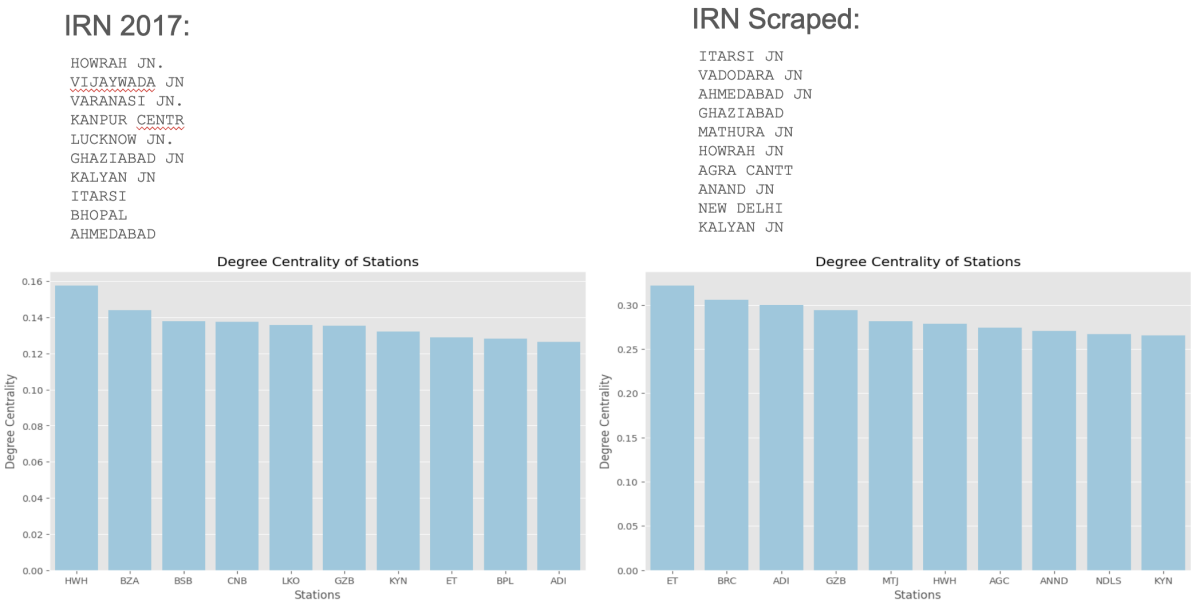

Fig.10. Top 10 stations according to highest values of Degree Centrality for IRN 2017 and Scraped dataset

3.9.2. Betweenness Centrality

Betweenness centrality is a measure in network analysis that identifies nodes that act as crucial bridges or intermediaries in the flow of information, traffic, or interactions within a network. It quantifies the extent to which a node lies on the shortest paths between other nodes. In simpler terms, it reflects how often a node serves as a critical connector or bottleneck in the network. It represents the number of times a station node lies on the shortest path between other stations. These nodes have a high influence on the network.

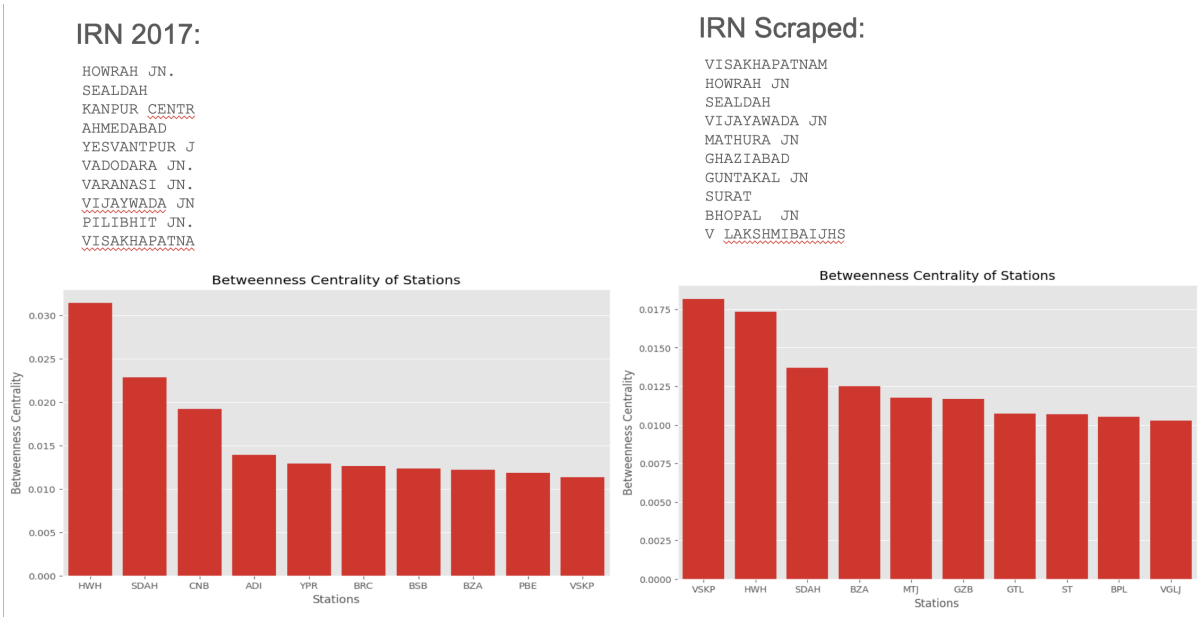

Fig.11. Top 10 stations according to highest values of Betweenness Centrality for IRN 2017 and Scraped dataset

3.9.3. Closeness Centrality

Closeness centrality is a measure used in network analysis to assess the importance or centrality of a node within a network. It quantifies how close a node is to all other nodes in the network, emphasizing nodes that are more central and have shorter average distances to other nodes. Identifies the good broadcasters in the network.

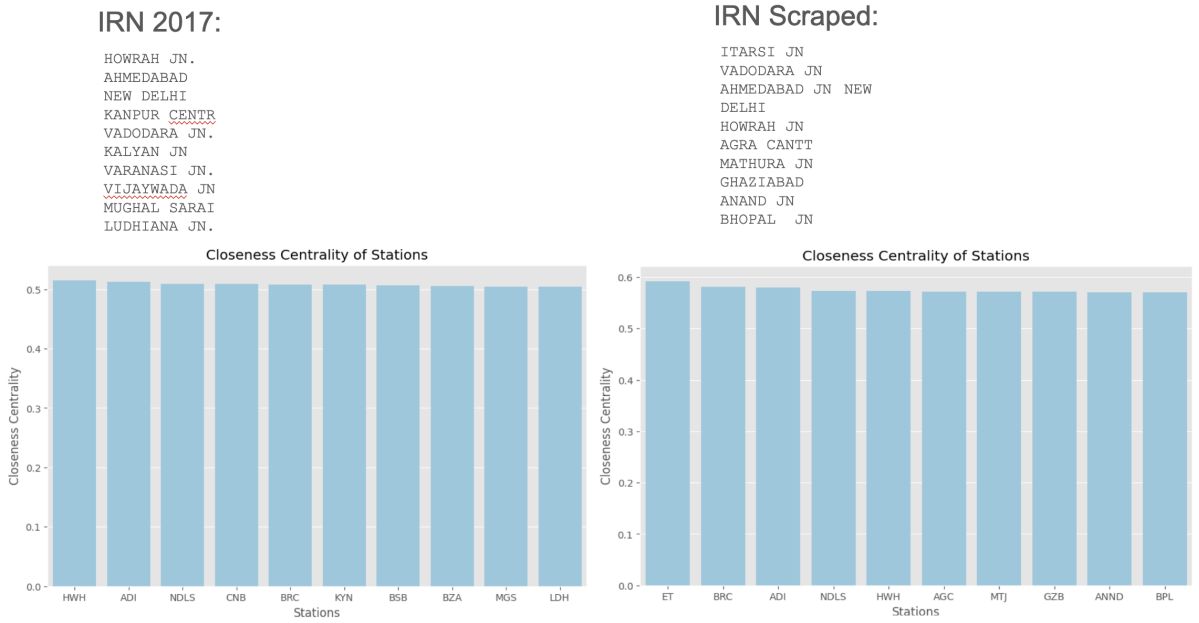

Fig.12. Top 10 stations according to highest values of Closeness Centrality for IRN 2017 and Scraped dataset

3.10. Clustering Coefficient

The clustering coefficient in a graph measures how connected a node's neighbors are to each other. For the IRN, we plot the average clustering coefficient $cc(k)$ of nodes having degree k as a function of k for the 2010, 2017 and 2023 datasets.

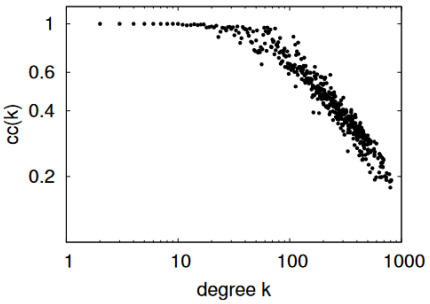

Fig.13. Clustering Coefficient 2010 IRN

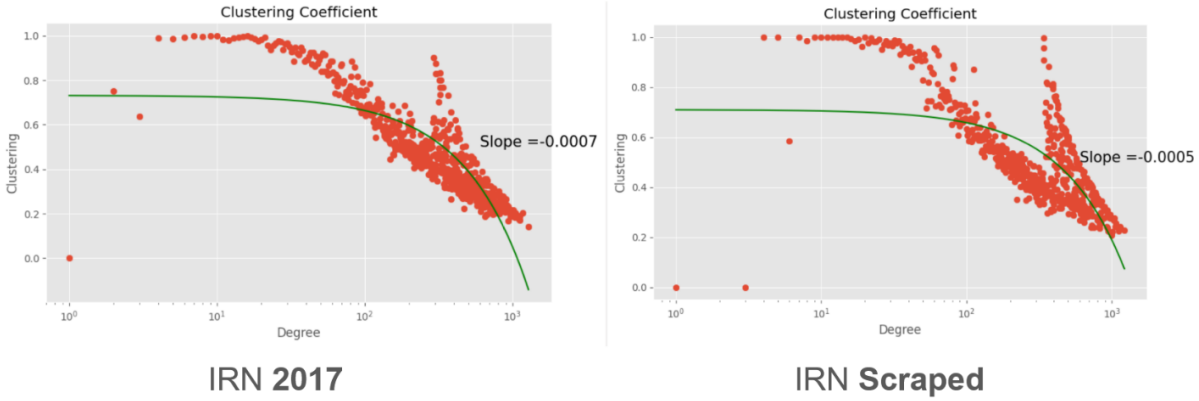

Fig.14. Clustering Coefficient for IRN 2017 and IRN Scraped 2023 Datasets

We observe that lower degree nodes have a clustering coefficient close to 1, and it decays following power law as the degree increases. What this means is that lower-degree nodes tend to form close inter-knit communities called cliques[1]. In the context of IRN, this means that lower degree stations which are only serviced by a few trains, connect to each other, forming a clique. In contrast, larger stations are connected to many stations that are geographically distant and thus are not connected to each other, making their clustering coefficient small.

A quick comparison of these plots shows that not much has changed in IRN if we only consider the clustering coefficient. A new noticeable feature, however, is the appearance of a new arrow-tail-like shape in the 2017 and 2023 graphs, which implies the existence of new high-degree nodes having clustering coefficients touching unity. This also cannot be dismissed since this feature emerges in both 2017 and the latest 2023 dataset, both of which have different sources. In the IRN context, this means that there are now stations that have a high degree (e.g. junctions and interchanges) and also have closely interconnected neighbors, forming cliques.

3.11. Other Network Inferences

3.11.1 Top 10 Stations with the highest average delay

To analyze the architecture of the IRN, we found the stations that had the highest average delay in the 2023 dataset. These were as follows.

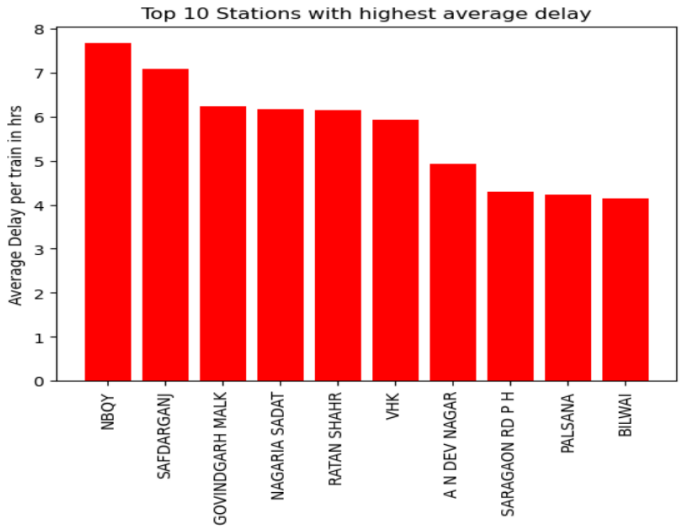

Fig.15. Top 10 stations with the highest average delay

State	Stations
Uttar Pradesh	1. Safdarganj 2. Nagaria Sadat 3. VHK(Jeonathpur) 4. A N Dev Nagar 5. Bilwai
Rajasthan	1. Govindgarh Malk 2. Ratan Shahr 3. Palsana
Assam	NBQY
Chhattisgarh	Saragaon RD PH

A quick reading shows that out of the 3800 stations in the graph, in the top 10 stations with maximum delay, 5 are in Uttar Pradesh, and 3 are in Rajasthan, highlighting the need to work on railway infrastructure in these regions.

3.11.2 Top 10 Trains with Highest Average Total Delay

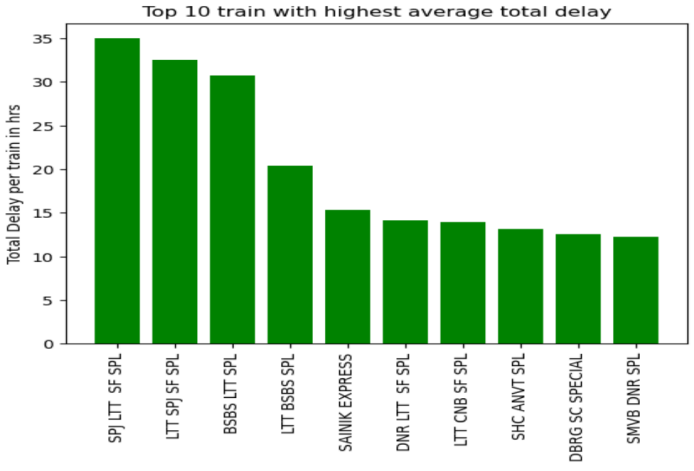

Fig.16. Top 10 trains with the highest average delay

In the latest 2023 scraped dataset, the Samastipur, Bihar, to Lokmanya Tilak Terminus, Mumbai, is the most delayed train. LTT station alone 6 trains in top delayed. This might be because of temporal effects on the dataset, for example, because of fog or rainy season.

3.11.3 Top 10 Stations based on number of journeys

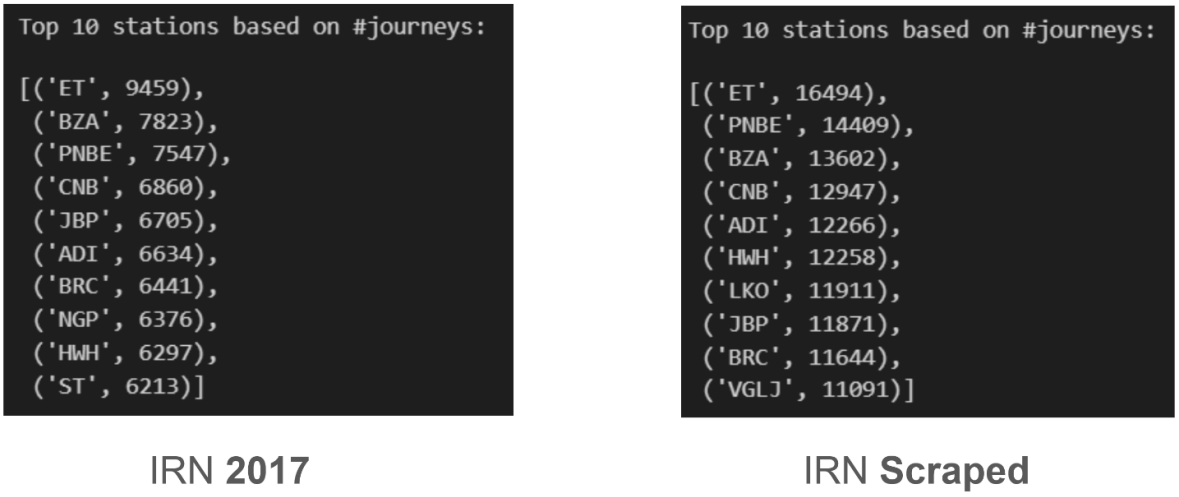

Fig.17. Top 10 stations based on number of journeys for IRN 2017 and IRN Scraped 2023 datasets

The top 10 stations based on the number of journeys that can be undertaken have also varied slightly from 2017 to 2023

2017		**2023**	
Station Code	Location	Station Code	Location
ET	Itarsi, MP	ET	Itarsi, MP
BZA	Vijaywada, AP	PNBE	Patna, Bihar
PNBE	Patna, Bihar	BZA	Vijaywada, AP
CNB	Kanpur, UP	CNB	Kanpur, UP
JBP	Jabalpur, MP	ADI	Ahemdabad, Guj
ADI	Ahemdabad, Guj	HWH	Howrah, WB
BRC	Vadodara, Guj	**LKO**	**Lucknow, UP**
NGP	**Nagpur, Maha**	JBP	Jabalpur, MP
HWH	Howrah, WB	BRC	Vadodara, Guj
ST	**Surat, Guj**	**VGLJ**	**Jhansi, UP**

From the 2017 to 2023 dataset, there have been two deletions in the top stations from which a maximum number of journeys can be undertaken and two additions. The ET, Itarsi Junction in Madhya Pradesh, retains its position as the number one station from which the maximum number of journeys can be undertaken. This station is located in the centre of India.

4. Delay Prediction

Train delay prediction can enhance the effectiveness of train dispatching, enabling dispatchers to make more informed decisions based on accurate estimations of train operations. Train delays are influenced by a multitude of factors, including passenger volume, technical malfunctions, extreme weather conditions, and dispatching strategies. Train departure times are typically determined by dispatchers, who are constrained by their knowledge and strategies. Existing train delay prediction methods often fail to fully account for the temporal and spatial dependencies between multiple trains and routes.

4.1 Station delay instead of train delay

Following [7], instead of predicting an individual train's delay, we focus on predicting the cumulative impact of train delays over a specified period, represented by the total number of delays at a particular station, rather than predicting the exact delay time for individual trains. Why? At every station, the specific dispatching decision is made by the train dispatching department. An individual's train's delay is of little significance. Predicting the number of delayed trains at a station in a certain period is more valuable for the dispatcher's decision-making. Let us understand this with an example presented in [7].

TRAIN DEPARTURE INFORMATION

number	expected departure time	train status	terminal	current time	passengers	departure station	actual departure time
t1	10:00	delay	Shanghai	10:32	900	yes	10:35
t2	10:15	delay	Taiyuan	10:32	252	yes	10:45
t3	10:20	delay	Wuhan	10:32	286	no	10:50
t4	10:30	delay	Shanghai	10:32	980	yes	10:40

Train departure information of four trains at Beijingnan station [7].

In Beijingnan station (Beijing), there are four trains (t1, t2, t3, t4) to Shanghai, Taiyuan and Wuhan, respectively. The table above shows the departure information for these four trains. Due to extreme weather, the trains are delayed. The station dispatcher may give priority to trains t1 and t4 to Shanghai based on the station environment (such as passenger flow).

4.2 Limitation of prior work

Prior work on delay prediction employs the historical delay data of each train for training. This is a flawed approach, as trains don't exist in isolation. A train's delay "spreads" in its network. Delays at a station are very likely to cause delays at the neighboring stations. Hence, employing this structural dependency along with the historical data is vital. This is where graphs come into play.

4.3 Graphs

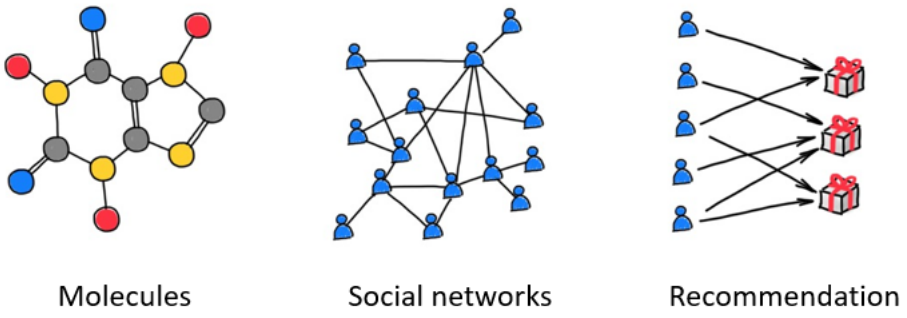

Figure 18: Graphs can represent a variety of real-life data.
Source: https://towardsdatascience.com/graph-neural-networks-beyond-weisfeiler-lehman-and-vanilla-message-passing-bc8605fa59a

A graph is a collection of nodes connected by edges. The nodes and edges can have attributes as well. Powerful models can be trained by combining the structural information with the node/edge features. A graph G is stored as a tuple (V, E, X_N, X_E) where V is the node set, E is the edges between the nodes, X_N are the node features, and X_E are the edge features.

4.4 Graph Neural Networks

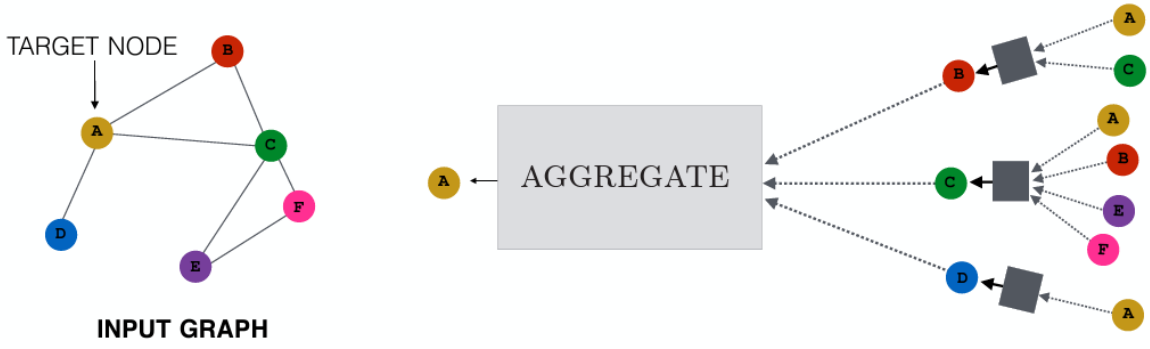

Figure 19: Representation of a Graph Neural Network.
Source: Graph Representation Learning Book, William L. Hamilton, McGill University

Graph neural networks (GNNs) are a type of deep learning architecture specifically designed to operate on graph-structured data. Unlike traditional neural networks, which are primarily designed for processing grid-like data such as images, GNNs can effectively capture and

utilize the inherent relationships between entities represented as nodes connected by edges in a graph. This makes them well-suited for a wide range of applications involving relational data, such as social networks, knowledge graphs, and molecular structures.

GNNs achieve their graph-learning capabilities by iteratively aggregating information from neighboring nodes, allowing them to learn complex patterns and representations of the graph structure. This aggregation process is typically implemented using a message-passing mechanism, where each node updates its state based on the information received from its neighbors. Through multiple rounds of message passing, GNNs can effectively propagate information across the graph, enabling them to capture global patterns and relationships. Figure 19 shows a block diagram of the same.

Temporal Graphs

Temporal graphs, also known as dynamic graphs, are a type of graph-structured data that represents relationships between entities and their evolution over time. Unlike static graphs, which capture relationships at a single point in time, temporal graphs capture how these relationships change and evolve over a sequence of time steps.

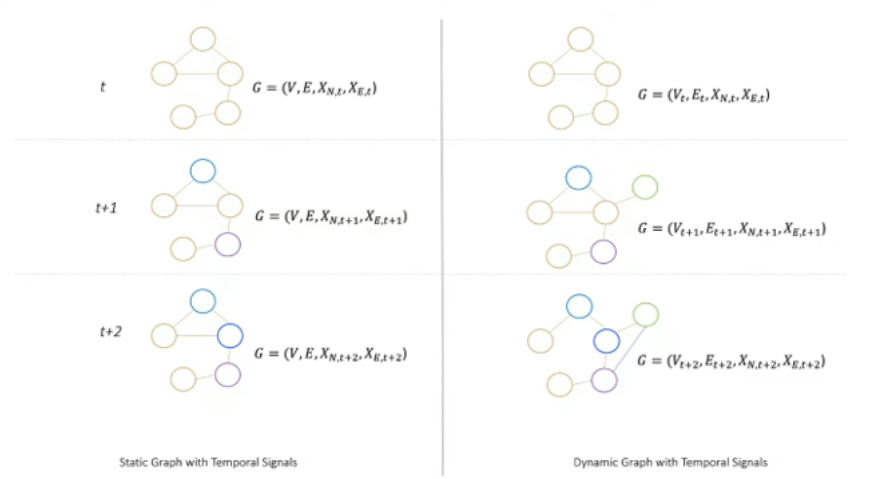

Figure 20: Types of temporal graphs.
Source: https://youtu.be/WEWq93tioC4

Temporal graphs can have a static topology with dynamic features, or the topology can also vary over time (Figure 20). The former is called a static graph with temporal signals; the latter is called a dynamic graph with temporal signals.

Temporal Graph Neural Networks

Temporal Graph Neural Networks (T-GNNs) are an extension of Graph Neural Networks (GNNs) specifically designed to handle temporal graphs. T-GNNs incorporate temporal information into the message-passing mechanism, enabling them to capture the dynamic nature of the graph and learn temporal patterns and representations.

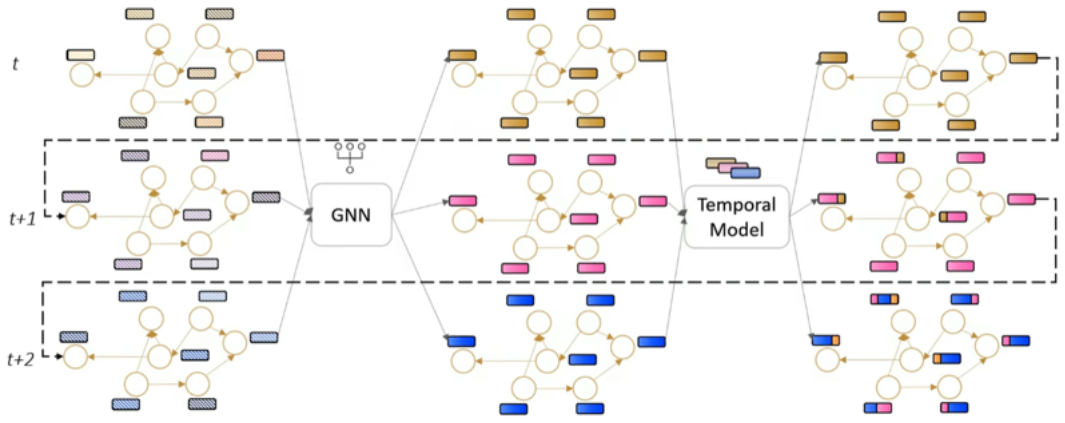

Figure 21: T-GNN layer
Source: https://youtu.be/WEWq93tioC4

T-GNNs achieve this by introducing a time dimension to the message-passing process. At each time step, nodes not only receive information from their immediate neighbors but also consider the temporal context of these interactions. This allows T-GNNs to learn how the graph structure and relationships change over time, enabling them to model temporal dependencies and dynamics. Figure 21 shows a single block layer of a T-GNN. Multiple such blocks are stacked to form a T-GNN. We recommend watching this video for a better understanding.

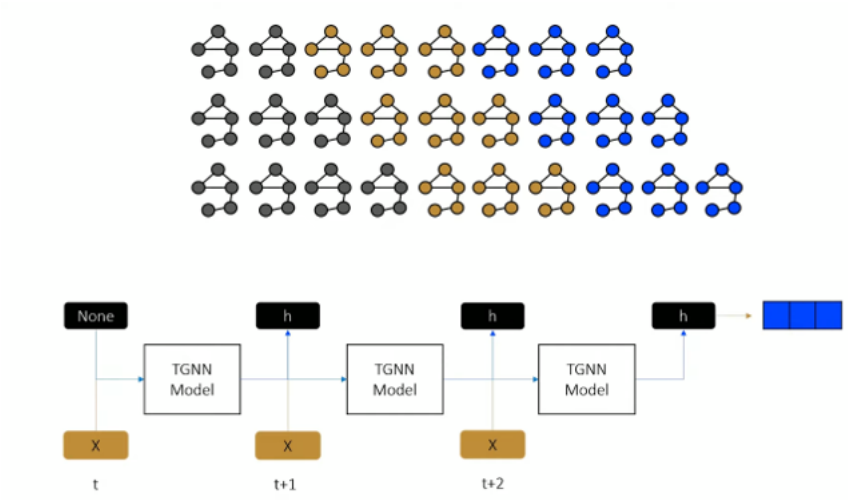

Figure 22: Training a T-GNN involves running a sliding window over the graph's time dimension.
Source: https://youtu.be/Rws9mf1aWUs

Training a T-GNN involves running a sliding window over the graph's time dimension. In Figure 22, three rows show three training iterations. In each row, the yellow graphs are the training timestamps, and the blue ones are the target timestamps. Notice that a target timestamp (blue) becomes a training timestamp (yellow) in the next iteration. It is up to the developer to decide how many timestamps he'll use for training and how many timestamps in the future he will predict per iteration. In this figure, three timestamps are used for training, and the model predicts three timestamps into the future at once.

Results

We design the Indian Railway Network as a static graph with dynamic features. The stations represent the nodes, and tracks form the edges. Every node, on every timestamp, has two features: the number of delayed trains on that day; and the average delay on that day. Every edge has two attributes: the number of trains that run on that track; and the distance between the connecting nodes. We conduct two experiments: at each station, predict 1) the number of delayed trains on the next day, and 2) predict the average delay on the next day. The target feature is chosen based on the experiment. For experiment 1, the number of delayed trains becomes the target. For experiment 2, the average delay becomes the target.

We use Torch Geometric Temporal [9] to create the temporal datasets. We use the past 7 timestamps (each timestamp is a day) for training and predict one timestamp into the future. We use A3TGCN [10] as the T-GNN. You can find the details in our repository. One thing to note is that A3TGCN does not utilize the edge attributes. Instead, it uses edge weights. An edge's attribute is a vector, while a weight is just a number. To overcome this, we pass the edge attributes through a fully connected layer with an output dimension set to one. We train this layer alongside A3TGCN so that it learns to combine the edge attributes into an edge weight meaningfully.

Here is the Mean Absolute Error over the test set:

Experiment	MEA
# Delayed trains	0.2340
Average train delay	2013.7595

The model learning the average train delay did not converge. We did not have time to finetune the model. We had some ideas but did not have the time to try them out.

5. Unfinished Experiments

Dynamic graph dynamic data

We are aware of the days of the week that each train operates. We can create dynamic edges by incorporating this data.

T-GNN architecture

Usually, T-GNNs are run in parallel over different time granularities: monthly, weekly, and daily. We only used the weekly window for training.

Periodicity in delay

Is the train delay cyclic across the weeks and months?

T-GNN explainer

We planned to apply T-GNNExplainer [11] on the trained model to highlight the reasons behind the delay. T-GNNExplainer [11] is an XAI model that identifies which subgraphs across time the trained model utilizes to make its prediction. Sadly, the authors have not made their code public.

6. References

1. Ghosh, Saptarshi, et al. "Statistical analysis of the Indian railway network: a complex network approach." *Acta Physica Polonica B Proceedings Supplement* 4.2 (2011): 123-138.
2. Sen, Parongama, et al. "Small-world properties of the Indian railway network." *Physical Review E* 67.3 (2003): 036106.
3. medium.com/codex/analysis-of-the-indian-railway-network-4fe0c501779e
4. data.gov.in/resource/indian-railways-time-table-trains-available-reservation-01112017#api
5. https://www.prokerala.com/travel/indian-railway/trains/
6. A. Barrat, M. Barthélemy, R. Pastor-Satorras, A. Vespignani, Proc. Natl. Acad. Sci. USA 101, 3747 (2004).
7. Zhang, Dalin, et al. "Train time delay prediction for high-speed train dispatching based on spatio-temporal graph convolutional network." IEEE Transactions on Intelligent Transportation Systems 23.3 (2021): 2434-2444.
8. Newman, M.(2018), Networks (2nd Edition).
9. Rozemberczki, Benedek, et al. "Pytorch geometric temporal: Spatiotemporal signal processing with neural machine learning models." *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2021.
10. Bai, Jiandong, et al. "A3t-gcn: Attention temporal graph convolutional network for traffic forecasting." *ISPRS International Journal of Geo-Information* 10.7 (2021): 485.
11. Xia, Wenwen, et al. "Explaining temporal graph models through an explorer-navigator framework." *The Eleventh International Conference on Learning Representations*. 2022.