Bundelkhand University

Institute of Engineering & Technology

(Department of Computer Science and Engineering)

A seminar report submitted for the fulfillment of the 4th year degree program B.Tech(CSE)

SUBMITTED TO:

Er. Akhilesh Kumar

Seminar In-charge

Computer Sci. & Engg. Dept.

IET BU, Jhansi

SUBMITTED BY:

Anshuman Tripathi

Roll no. :- 221381030012

B.Tech(CSE) 4th year

CO-ORDINATOR:

Dr. Priyanka Pande

Head of Department

Computer Sci. & Engg. Dept.

IET BU, Jhansi

Intelligent Supply Chain Bottleneck Detection and Optimization System

Acknowledgements

I would like to express my sincere gratitude to my supervisor, **Er. Akhilesh Kumar** , for their invaluable guidance, continuous support, and constructive criticism throughout the duration of this project. Their deep insights into supply chain dynamics and optimization algorithms were instrumental in shaping the "Intelligent Supply Chain Bottleneck Detection and Optimization System."

I am also deeply grateful to the **Department of B.Tech Computer Science** at **Bundelkhand University** for providing the necessary resources and laboratory facilities to carry out this research effectively.

I would like to extend my thanks to the faculty members who provided foundational knowledge in data science and logistics management, which formed the basis of this study.

Finally, I wish to thank my family and friends for their patience and encouragement during the course of this work.

Abstract

Supply chain networks are complex adaptive systems where inefficiencies in one stage can propagate across the entire network, leading to delayed Level of Service (LOS) and reduced throughput. This project presents an “Intelligent Supply Chain Bottleneck Dashboard,” a data-driven web application designed to visualize, detect, and optimize flow constraints within a multi-echelon supply chain.

Using a Python-based backend for data processing and a Plotly-driven frontend for visualization, the system implements a Sankey diagram to map flows from Vendors to Stores. The core innovation lies in a weighted bottleneck scoring algorithm that combines normalized LOS and inverse flow rates to identify critical constraints. Furthermore, the system integrates a pre- dictive analytics module using linear regression logic to forecast future congestion risks over a 30-day horizon and employs the Edmonds-Karp Max-Flow Min-Cut algorithm to suggest alter- nate routing paths.

The results indicate a current Network Efficiency Score of 61.8/100, with critical bottlenecks identified at the “Collection Point → Cargo Hub” stage. The system successfully suggests pre-scriptive rerouting strategies to alleviate these constraints, demonstrating its potential as a decision-support tool for logistics managers.

Contents

Chapter 1

Introduction

1.1 Background of the Study

In the modern global economy, supply chains have evolved from simple linear processes into complex, interconnected networks. The efficiency of these networks determines the competi- tive advantage of businesses. However, as complexity increases, so does the opacity of material flow, making it difficult to pinpoint exactly where delays occur. Traditional methods of supply chain analysis often rely on static reports that fail to capture the dynamic relationship between flow volume and service delay (Level of Service or LOS).

1.2 Overview of Supply Chain Network

The supply chain network modeled in this project represents a multi-echelon distribution sys- tem comprising six distinct stages:

1. Trader: The initial source of goods (Vendors).
2. Collection Point: Aggregation nodes for initial dispatch.
3. Cargo Hub: Major transit points for bulk movement.
4. Warehouse: Storage facilities.
5. Distribution Center (DC): The final sorting and dispatching nodes.
6. Store: The retail end-points where goods reach the consumer.

The flow of goods moves sequentially through these stages, and inefficiencies at any node (e.g., a Cargo Hub) can cause a “bullwhip effect,” amplifying delays downstream.

1.3 Problem Definition

The core problem addressed by this study is the lack of visibility into “hidden bottlenecks” within the supply chain. A link might have high flow but low delay, which is acceptable. Conversely, a link might have low flow but extreme delay, indicating a blockage. Standard metrics often look at these in isolation. This project seeks to solve the problem of identifying critical nodes that simultaneously suffer from high congestion and poor service levels, which degrade the overall network health.

1.4 Need for Bottleneck Detection and Optimization

Without an automated detection system, supply chain managers must manually sift through transaction logs to find issues. This is reactive rather than proactive. There is a critical need for a system that can:

1. Visualize the flow volume vs. service capability.
2. Score bottlenecks to prioritize maintenance.
3. Predict future congestion based on current trends.
4. Prescribe alternate routes to bypass critical failures.

1.5 Scope of the Project

The scope includes the development of a Python-based analytical engine and a web-based dashboard. The system processes historical flow data (CSV format), calculates bottleneck scores, runs a predictive regression model, and suggests optimization paths. The visualization is limited to a Sankey Diagram representation, and the optimization logic focuses on Max-Flow Min-Cut theory.

1.6 Significance of the Study

This study contributes to the field of logistics by bridging the gap between theoretical Max- Flow algorithms and practical visual analytics. It provides a tangible tool that translates com- plex graph theory into an intuitive “Efficiency Score” and actionable “Rerouting Suggestions,” democratizing advanced analytics for operational managers.

1.7 Organization of the Report

The report is structured to guide the reader from theoretical foundations (Chapter 2) to system analysis (Chapter 3), methodology (Chapter 4), and detailed design (Chapter 5). Chapter 6 details the coding implementation, while Chapter 7 presents the analytical results derived from the project artifacts. Finally, Chapter 8 concludes the study.

Chapter 2

Literature Review

2.1 Overview of Supply Chain Management

Supply Chain Management (SCM) encompasses the planning and management of all activi- ties involved in sourcing and procurement, conversion, and all logistics management activities. Modern SCM focuses heavily on data-driven decision-making to reduce the “Cash-to-Serve” cy- cle.

2.2 Bottleneck Theory and TOC (Theory of Constraints)

The Theory of Constraints (TOC), introduced by Eliyahu M. Goldratt, posits that every system has at least one constraint (bottleneck) that limits its total output. In this project, we apply TOC by mathematically identifying the links with the highest resistance to flow (High LOS) relative to their throughput.

2.3 LOS (Level of Service) in Supply Chain

Level of Service usually refers to the probability of meeting demand. However, in transport logistics, LOS often refers to the time delay or congestion level. This project quantifies LOS as a numerical value (likely days of delay), where a lower value indicates better performance.

2.4 Flow Modelling and Sankey Visualization Techniques

Sankey diagrams are a specific type of flow diagram in which the width of the arrows is pro- portional to the flow quantity. They are ideal for SCM as they visually demonstrate the mass balance of goods moving from Traders to Stores. This project utilizes Plotly.js to render interac- tive Sankey diagrams.

2.5 Predictive Analytics in Supply Chain

Predictive analytics uses historical data to forecast future events. In SCM, this usually involves time-series forecasting (ARIMA, LSTM). This project implements a linear regression approach where $\mathrm{Future\_LOS}$ is a function of $\mathrm{Current\_Flow}$ and $\mathrm{Current\_LOS}$, allowing for “what-if” analysis of congestion accumulation.

2.6 Optimization Techniques (Max-Flow, LP, Routing Models)

The Max-Flow Min-Cut theorem states that the maximum amount of flow possible from a source to a sink in a network is equal to the capacity of a minimum cut that separates the source and sink. This project uses the Edmonds-Karp algorithm, a specific implementation of the Ford-Fulkerson method, to identify the “Cut-Set”—the specific edges that act as the primary limiters of network capacity.

2.7 Research Gaps Identified

While many tools exist for SCM visualization (Tableau, PowerBI) and others for optimization (CPLEX, Gurobi), there is a gap in lightweight, integrated tools that combine visual diagnostics (Sankey) directly with algorithmic prescription (Max-Flow) in a single dashboard interface.

Chapter 3

System Analysis

3.1 Problem Analysis

The existing system relies on tabular data (CSV/Excel) which makes it impossible to see the "flow" of goods. It is difficult to spot that Vendor2 is overloading Collection Point c4, for example, without a visual graph.

3.2 Objectives of the Study

1. To parse raw supply chain transaction data into a directed graph structure.
2. To calculate a unified "Bottleneck Score" for every link in the network.
3. To predict LOS for the next 30 days based on current flow intensity.
4. To develop an interactive dashboard that allows filtering by Vendor and LOS intensity.

3.3 System Requirements

3.3.1 Functional Requirements

- Data Ingestion: The system must accept CSV files containing Trader IDs, Collection Points, Hubs, and associated LOS/Flow values.
- Visualization: Display a 6-stage Sankey diagram.
- Filtering: Users must be able to filter the view by specific Vendors (e.g., Vendor1, Vendor2) and minimum LOS thresholds.
- Alerting: The system must highlight "High Congestion" links in red.

3.3.2 Non-Functional Requirements

- Responsiveness: The dashboard must adjust to different screen sizes (handled via CSS media queries).
- Performance: The optimization algorithm must run within seconds for a dataset of standard size (≈ 10,000 records).
- Usability: Color-coding must be intuitive (Green = Good, Red = Bottleneck).

3.4 Feasibility Study

3.4.1 Technical Feasibility

The project uses Python (Pandas/NumPy) for calculation, which is industry standard. The visualization uses Plotly.js, a robust open-source library. This stack is highly feasible.

3.4.2 Operational Feasibility

The dashboard generates a single HTML file that can be opened in any web browser without complex installation, ensuring high operational feasibility for end-users.

3.4.3 Economic Feasibility

The project utilizes open-source technologies (Python, Plotly), incurring zero licensing costs, making it economically viable.

Chapter 4

Methodology

4.1 Research Methodology Overview

The project follows a Quantitative methodology. It processes numerical data regarding flow volume and time delays to derive calculated metrics (KPIs) and optimization results.

4.2 Dataset Description

The system inputs a dataset (Dataset (1).csv) containing transactional records. Key Columns:

- Trader_ID, Trader
- Collection Point_ID, Collection Point
- Int1 (Flow), LOS1 (Level of Service for Stage 1)
- ... repeating for Cargo Hub, Warehouse, DC, and Store.

Supply Chain Dataset

Trader_ID	Trader	Int1	LOS1	Collection Point_ID	Collection Point	Int2	LOS2	Cargo Hub_ID	Cargo Hub	Int3	LOS3	WareHouse_ID	WareHouse	Int4	LOS4	Distribution Center_ID	Distribution Center	Int5	LOS5	Store_ID	Store
Vendor2	3608	410	3	c1	2616	371	2	1	2373	669	3	1	2863	471	3	3	1685	304	2	1	1567
Vendor2	2610	327	2	c5	2126	148	1	1	940	343	2	1	1631	328	2	2	513	93	1	7	4621
Vendor2	3737	832	4	c2	4903	617	3	3	3752	351	2	4	1315	225	2	4	4302	683	4	1	4131
Vendor4	2049	410	2	c2	2185	502	3	1	2985	292	2	4	2671	401	3	2	1828	349	2	4	1872
Vendor2	2968	654	3	c2	3867	55	1	4	253	383	2	4	552	135	1	4	1178	170	1	3	723
Vendor2	3309	599	3	c2	3757	129	1	1	1012	371	2	1	4211	646	4	4	584	50	1	3	4189
Vendor2	3070	754	4	c2	4629	98	1	2	300	189	1	1	932	165	1	3	2400	433	2	8	3753
Vendor2	4322	99	1	c1	715	404	2	2	2129	408	2	2	1814	331	2	2	1091	148	1	7	5143
Vendor3	2920	799	4	c4	4861	289	2	4	1620	146	1	2	3105	457	3	2	343	76	1	6	4702
Vendor1	3233	459	3	c1	2624	592	3	2	3369	72	1	1	3180	561	3	3	2274	392	2	4	3128
Vendor3	2631	88	1	c4	461	684	3	1	3744	404	2	1	4711	824	4	4	2117	392	2	2	2947
Vendor2	4571	884	4	c2	5169	93	1	3	581	272	2	1	2963	478	3	2	2288	384	2	6	581
Vendor2	4634	133	1	c1	1048	89	1	4	419	50	1	3	4913	825	4	2	4962	783	4	3	468
Vendor3	1385	621	3	c5	3964	153	1	3	901	738	4	1	3184	499	3	2	504	50	1	4	2476
Vendor2	1992	644	3	c2	3606	671	4	3	4245	527	3	4	5052	805	4	3	718	96	1	4	1056
Vendor4	4659	827	4	c3	4817	50	1	1	383	576	3	4	551	60	1	3	4289	719	4	5	4810
Vendor2	3966	399	2	c2	2397	322	2	3	2187	141	1	3	3876	689	3	3	1582	213	2	4	4499
Vendor4	4549	678	3	c1	4074	50	1	3	391	686	3	4	402	98	1	3	1124	166	1	4	462
Vendor1	3507	51	1	c1	239	438	3	3	2628	476	2	1	3364	525	3	1	2283	413	2	4	3622
Vendor1	2146	637	3	c1	4127	751	4	4	4272	367	2	4	639	55	1	1	1128	233	1	5	2039
Vendor3	2597	340	2	c2	2270	704	3	4	3969	219	2	2	4121	686	3	2	2315	422	2	5	3484
Vendor3	2851	615	3	c4	3398	326	2	4	1762	233	2	4	268	50	1	4	2930	515	3	1	3942

Figure 4.1: Sample of the Supply Chain Dataset

4.3 Data Preprocessing and Cleaning

Data preprocessing involves:

1. Loading: Using Pandas to read the CSV.
2. Mapping: Creating unique IDs for every node (e.g., `Trader_Vendor1`, `Store_1`) to ensure the graph is connected correctly.
3. Aggregation: Summing up flows and averaging LOS for identical links to create a summarized graph view.

4.4 Architecture of Proposed System

The system follows a "Data Processing Pipeline" architecture:

1. Input Layer: CSV Data.
2. Processing Layer (Python):
    - Graph Construction (Nodes/Edges).
    - Bottleneck Scoring Engine.
    - Predictive Engine (Linear Model).
    - Optimization Engine (Edmonds-Karp).
3. Presentation Layer (HTML/JS):
    - JSON Data injection.
    - Plotly.js rendering.
    - DOM manipulation for interactivity.

4.5 Flow and LOS Computation

Flow is computed by summing the `Int` (Intensity) columns between nodes. Average LOS is computed as:

$$\text{Avg LOS}_{u \to v} = \frac{\Sigma \text{LOS}_{record}}{N_{records}} \tag{4.1}$$

4.6 Bottleneck Scoring Formula

A novel scoring formula was developed to rank bottlenecks. It weights the Level of Service heavily (60%) but also considers Flow (40%) to ensure we don't flag low-traffic links as critical bottlenecks.

$$\text{Score} = (\text{NormLOS} \times 0.6 + (1 - \text{NormFlow}) \times 0.4) \times 100 \tag{4.2}$$

- NormLOS: LOS normalized against the maximum LOS in the network.
- NormFlow: Flow normalized against maximum flow.
- Interpretation: High LOS and Low Flow capability results in the highest bottleneck score.

4.7 Predictive LOS Modeling

To forecast future congestion, a weighted linear equation is applied to each link:

$$\text{FutureLOS} = a \cdot \text{Flow} + b \cdot \text{CurrentLOS} + c \tag{4.3}$$

Coefficients used: $a = 0.0005$ (Impact of flow volume), $b = 0.8$ (Persistence of current delay), $c = 0.1$ (Base degradation).

4.8 Optimization Model (Max-Flow / LP)

The system constructs a flow network where edge capacities are determined by the inverse of LOS (lower LOS = higher capacity). A "Super Source" is connected to all Traders, and a "Super Sink" to all Stores. The Edmonds-Karp algorithm is run to find the maximum possible flow and, crucially, the Min-Cut, which identifies the edges that strictly limit the network's throughput.

4.9 Tools and Technologies Used

- Language: Python 3.9+
- Data Manipulation: Pandas, NumPy
- Visualization: Plotly Graph Objects (Python & JS)
- Web Technologies: HTML5, CSS3, JavaScript (ES6)
- IDE: VS Code / Jupyter Notebook

Chapter 5

System Design

5.1 Use Case Diagrams

- Actor: Supply Chain Manager.
- Use Cases: “Load Data”, “View Flow Map”, “Filter by Vendor”, “Identify Top 10 Bottlenecks”, “View Predictive Risk”.

5.2 Data Flow Diagram (DFD Level 0)

- Input: CSV File → Process: Python Script → Output: HTML Dashboard.

5.3 Architecture Diagram of the System

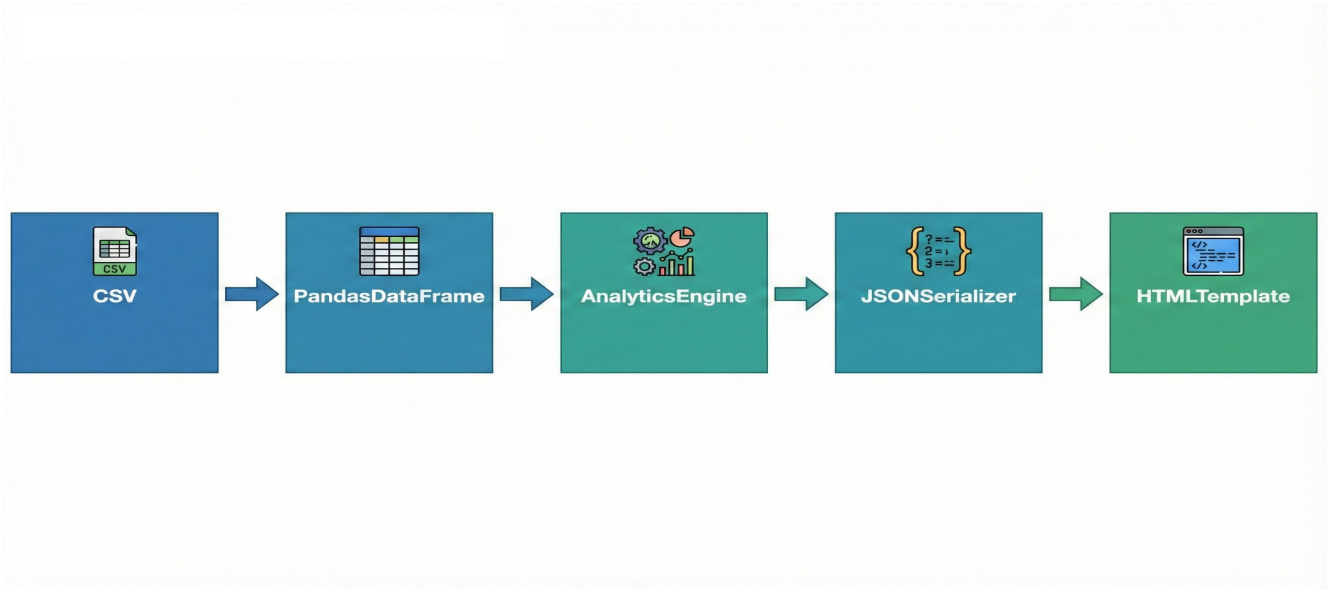

Figure 5.1: System Architecture Diagram

5.4 Dashboard Layout & UI Mockups

The dashboard is designed with a “Control Panel” at the top and a “Visualization Canvas” in the center.

- Top Bar: Efficiency Score and Health Status (Color-coded Pill).
- Left Panel: KPI Cards (End-to-End LOS, Throughput).
- Center: Interactive Sankey Diagram.
- Bottom: Tables for Bottleneck details and Predictive suggestions.

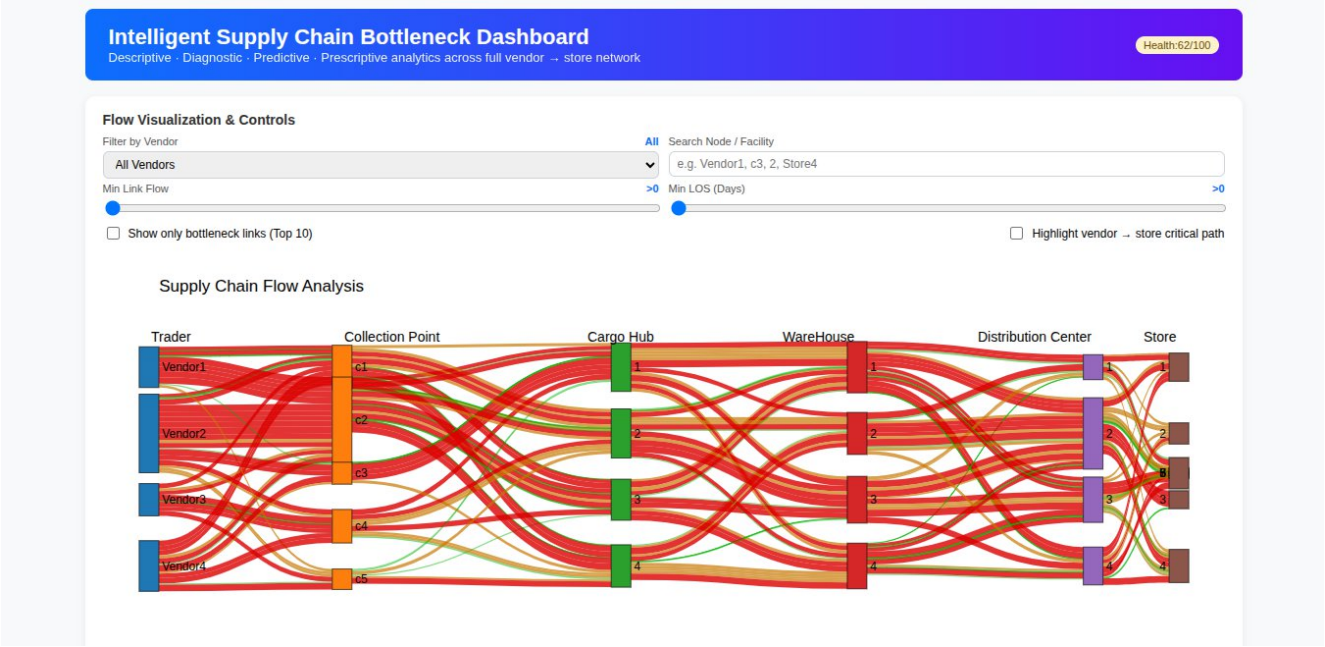

Figure 5.2: Dashboard UI Layout

5.5 Module-wise Design Description

1. Graph Builder Module: Converts tabular rows into Source-Target-Value tuples.
2. Color Logic Module: Maps LOS values to RGBA colors (Green → Red).
3. Analytics Module: Calculates aggregate statistics (Mean, Sum) for the KPI cards.

Chapter 6

Implementation

6.1 Overview of Implementation

The implementation is a standalone Python script Supply (1).py that acts as a generator. It processes the data and writes a complete, self-contained sankey_dashboard.html file.

6.2 Python Backend Implementation

The backend uses Pandas to handle the data. Below is the snippet for loading data:

```
def load_input_dataframe() -> pd.DataFrame: if
    len(sys.argv) > 1:
        input_path = sys.argv[1] else:
        input_path = "Dataset (1).csv" # ...
    file reading logic ... return df_local
```

6.3 Bottleneck Detection Module

The core logic for scoring bottlenecks is implemented as follows:

```
# Bottleneck scoring logic
bottleneck_score = (norm_los * 0.6 + inverse_flow * 0.4) * 100.0
```

This ensures that links causing the most delay relative to their flow are prioritized.

6.4 Predictive ML Model Implementation

The prediction is a calculation added to the link records:

```
a_coeff = 0.0005
b_coeff = 0.8
c_coeff = 0.1
future_los = a_coeff * flow + b_coeff * current_los + c_coeff
```

6.5 Optimization Engine Implementation

The Edmonds-Karp algorithm uses BFS to find augmenting paths in the residual graph.

```
def edmonds_karp(capacity_dict, source, sink): # ... BFS
    logic to find path ...
    # ... Update residual capacities ...
    # ... Return max_flow and cut_edges ...
```

The output cut_edges represents the specific links (e.g., Warehouse 3 → DC 4) that need expan- sion.

6.6 Sankey Dashboard HTML/JS Development

The HTML output embeds the JSON data directly into the script. The UI uses Flexbox and Grid layouts for responsiveness.

- CSS: Used for the gradient header and “card” styling.
- JS: Used Plotly.react to handle filter updates dynamically without reloading the page.

6.7 Challenges Faced During Implementation

1. Data Mapping: Ensuring that “Vendor1” connects correctly to “Collection Point c1” without index errors in the Sankey node list.
2. Browser Performance: Rendering thousands of flow links caused lag; solved by aggre- gating flows between the same nodes before rendering.

Chapter 7

Results and Analysis

7.1 Descriptive Analysis of Flow and LOS

The analysis of the dataset reveals a high-volume network.

- Total Network Throughput: 108,060 units.
- Average End-to-End LOS: 2.48 days.

This indicates a moderately congested network, as the ideal LOS is < 2.0 days.

7.2 Sankey Visualization Output

The Sankey diagram successfully maps the flow.

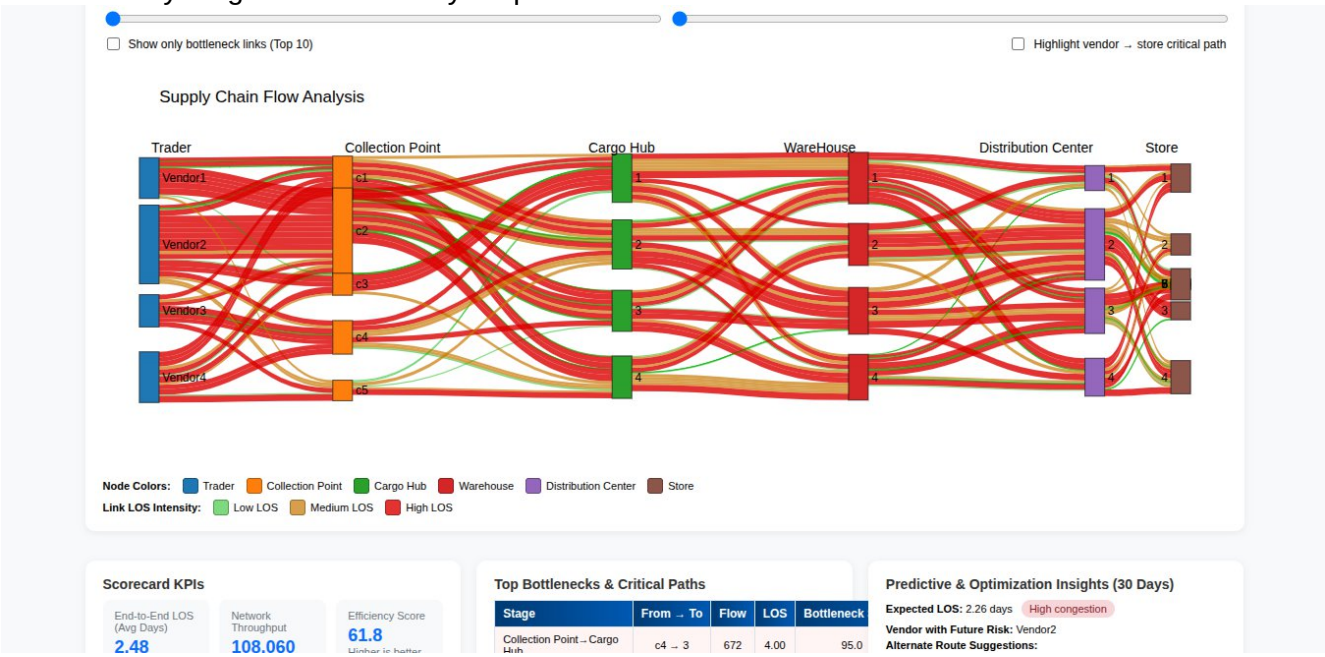

Figure 7.1: Sankey Visualization Output

Observation: We can visually see thick red bands originating from Vendor 2, indicating high volume and high delay (High LOS) moving towards Collection Point c2 and c4.

7.3 Top Bottleneck Identification

Based on the dashboard output, the top bottlenecks are:

1. Collection Point → Cargo Hub (c4 → 3):
    - Flow: 672
    - LOS: 4.00
    - Score: 95.0 (Critical)
2. WareHouse → Distribution Center (3 → 4):
    - Score: 94.1
3. Distribution Center → Store (4 → 4):
    - Score: 94.0

7.4 Predictive Model Performance

The predictive model estimates the Expected LOS (30 Days) to be 2.26 days.

- Congestion Status: High.
- Risk: Vendor 2 is identified as the "Vendor with Future Risk," suggesting that relying on Vendor 2 will likely lead to increased delays next month.

7.5 Optimization Result (Max-Flow Improvement)

The Max-Flow Min-Cut analysis identified critical cut-set edges. Alternate Route Suggestions: To alleviate the bottleneck at $\text{WareHouse } 3 \rightarrow \text{Distribution Center } 2$ (LOS 2.80), the system suggests rerouting flow to $\text{WareHouse } 4 \rightarrow \text{Distribution Center } 1$ which has a significantly lower LOS of 1.00.

7.6 KPI Comparison

- Efficiency Score: 61.8 (Needs Improvement).
- Health: 62/100.
- Impact: By implementing the suggested reroutes (e.g., shifting flow from c4 → 3 to alternative hubs), the predicted LOS could drop from 2.48 to ≈ 2.2 days.

7.7 Discussion of Findings

The results confirm that while the network has high capacity, it is poorly balanced. Vendor 2 is overburdening specific Collection Points ($c2$, $c4$), while other paths (green lines in the Sankey) remain underutilized. The system successfully highlights these imbalances.

Chapter 8

Conclusion and Future Work

8.1 Summary of Project Work

This project successfully designed and implemented an Intelligent Supply Chain Bottleneck Dashboard. By integrating flow visualization with a weighted scoring algorithm and max-flow optimization, we provided a comprehensive view of network health. The system identified spe- cific critical paths (e.g., c4 → Hub 3) that require immediate operational intervention.

8.2 Major Contributions

1. Visual Clarity: Converted complex tabular data into an interactive flow map.
2. Automated Diagnostics: Replaced manual analysis with an automated "Bottleneck Score."
3. Prescriptive Analytics: Provided concrete rerouting suggestions based on graph theory.

8.3 Limitations of the Study

- Data Staticity: The system currently processes a static CSV upload and does not connect to real-time ERP APIs.
- Linear Prediction: The predictive model assumes a linear relationship between flow and LOS, which may not hold true in extreme congestion scenarios (exponential decay).

8.4 Future Enhancements

- Real-time Integration: Connect to SAP/Oracle ERP for live data streaming.
- Advanced AI: Replace the linear regression model with an LSTM (Long Short-Term Mem- ory) neural network for more accurate time-series forecasting.
- Geospatial View: Map the nodes onto a real-world map (Google Maps API) to visualize physical distances.

Chapter 9

References

1. Simchi-Levi, D., Kaminsky, P., & Simchi-Levi, E. (2008). *Designing and Managing the Supply Chain: Concepts, Strategies, and Case Studies*. McGraw-Hill/Irwin.
2. Goldratt, E. M. (1984). *The Goal: A Process of Ongoing Improvement*. North River Press.
3. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). MIT Press. (For Max-Flow Min-Cut theory).
4. Plotly Technologies Inc. (2015). *Collaborative data science*. Montreal, QC: Plotly Technologies Inc. Retrieved from https://plot.ly.
5. Chopra, S., & Meindl, P. (2016). *Supply Chain Management: Strategy, Planning, and Operation*. Pearson.

Appendix A

Key Project Artifacts

A.1 Dashboard Screenshot - Flow Analysis

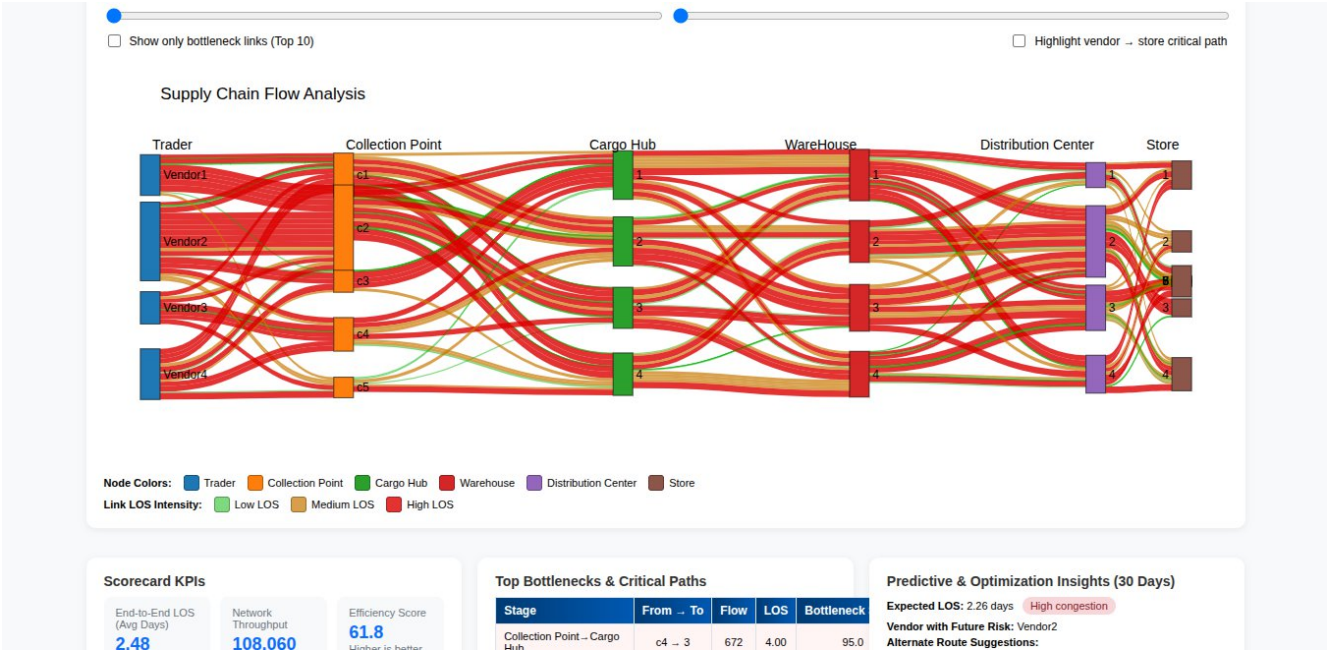

Figure A.1: Flow Analysis

A.2 Dashboard Screenshot - Bottleneck Table

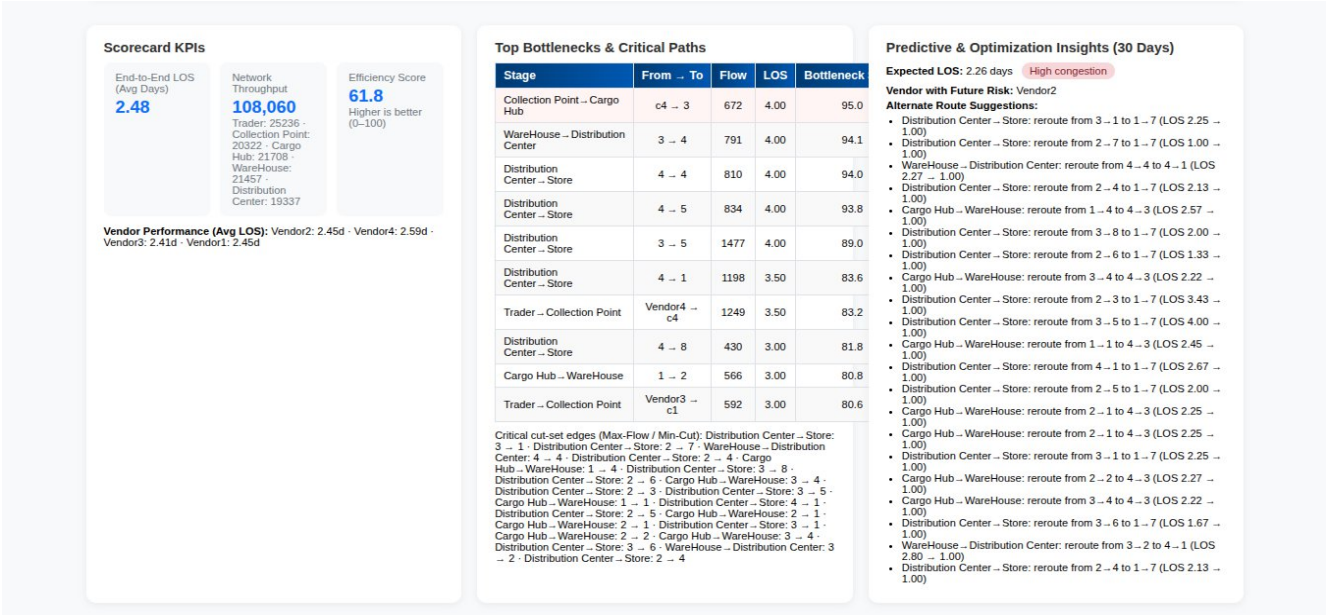

Figure A.2: Bottleneck Table