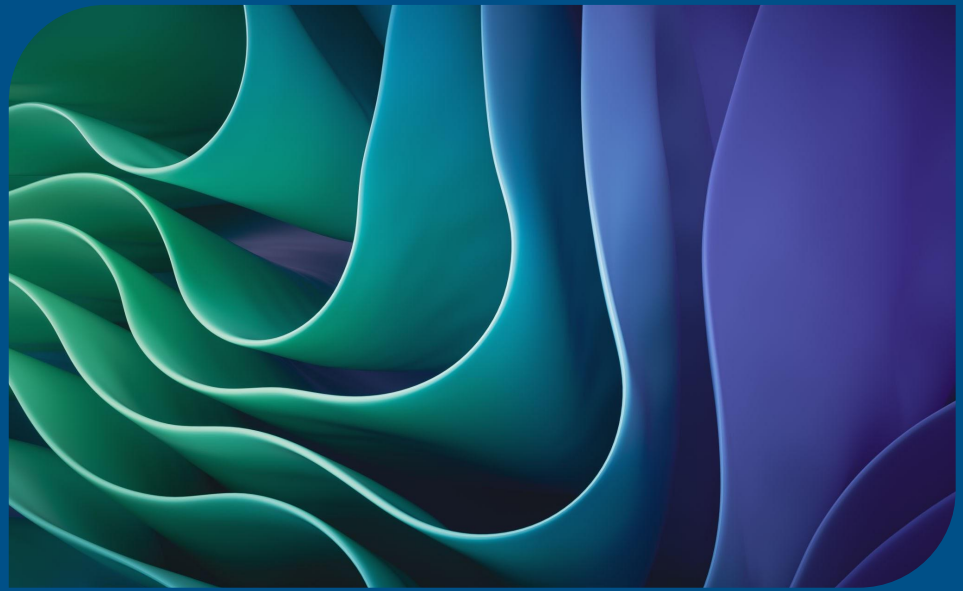
05.01.2025

Supply Chain Management (SCM)

Created By:
Hakan İspir - 150322052
Ege Aslan - 150322053



Operations Research 1 Project

1. Introduction
2. Model
3. Code Summary
4. Conclusion
5. Discussion

1. Introduction
2. Model
3. Code Summary
4. Conclusion
5. Discussion

1. Project Overview

In today's competitive global economy, supply chains play a critical role in ensuring the smooth flow of products from suppliers to customers. A typical supply chain comprises suppliers, manufacturing plants, warehouses, and retailers, all interconnected to meet demand efficiently. However, modern supply chains face significant challenges, including fluctuating demand, high transportation costs, and capacity constraints, making it difficult to achieve both efficiency and profitability.

The primary trade-off in supply chains involves balancing cost, service level, and resource allocation. Key cost drivers include transportation, inventory holding, and facility operations. To address these issues, organizations across various industries seek optimal supply chain models that minimize costs while maintaining high service levels.

The objective of this project is to develop a mathematical model for minimizing total supply chain costs while satisfying demand and adhering to operational constraints. Using optimization tools like Python and LINGO, the project focuses on four key goals:

1. **Cost Minimization:** Reduce overall costs related to transportation, inventory, and facility operations.
2. **Efficiency Improvement:** Optimize resource utilization and streamline the flow of goods.
3. **Demand Fulfillment:** Ensure customer demand is met without exceeding supply chain capacity.
4. **Scalability and Flexibility:** Create a model adaptable to changes in demand, capacity, and costs.

This project adopts a data-driven approach to supply chain performance analysis, providing actionable insights for strategic decision-making and enhancing competitiveness in the supply chain network.

2. Problem Overview

This project focuses on a supply chain optimization problem where a company seeks to minimize transportation costs while meeting demand at a destination port.

The supply chain involves multiple plants, each with a specific capacity and associated transportation costs.

The aim of the project is to develop a mathematical model that optimally allocates shipments from the plants to the port to satisfy demand at minimal cost.

The supply chain problem involves:
- 19 Plants, each with a specific daily capacity.
- 1 Destination Port (PORT09) with a total demand of 5791 units.
- Freight Costs for shipping goods from each plant to the port.

Data Summary

1. Plant Capacities:

- The capacities of the plants range from 7 units to 1070 units.

2. Freight Costs:

- The freight costs for shipping from each plant to PORT09 range from 1.202 to 19.3644.

3. Total Supply:

- The total available supply from all plants is exactly 5791 units, matching the demand at PORT09.

Requirements

The supply chain optimization model developed in this project successfully meets all the specified requirements:

- **Requirement 1:** The model includes **four sets**, specifically PLANTS, DESTINATIONS, TRANSPORT_COST, and a **derived set** for plant-to-destination shipments.
- **Requirement 2:** All **decision variables** and **problem parameters** are modeled as attributes of these sets, ensuring clarity and adherence to the structured modeling approach.
- **Requirement 3:** The model includes **three decision variable attributes**: CAPACITY, SHIPMENT, and COST, which govern plant capacities, shipment quantities, and transportation costs, respectively.
- **Requirement 4:** At least **four constraint statements** are included in the model: (1) Supply constraints ensuring shipments do not exceed plant capacities, (2) Demand constraint ensuring total demand is met, (3) Constraints ensuring positive shipment quantities, and (4) An objective function minimizing the total transportation cost.
- **Requirement 5:** The model involves **19 plants and 1 destination**, resulting in a total of **20 variables**, satisfying the minimum variable count criterion.
- **Additional Requirement:** A sensitivity analysis was performed on key parameters (transportation costs, plant capacities, and demand) using the sympy-based model to assess the impact of variations on the total cost and solution robustness.

1. Introduction
2. **Model**
3. Code Summary
4. Conclusion
5. Discussion

Mathematical Model/Sets

Set of Plants (P):

- Represents the set of plants that produce and ship products.

Set of Destination Ports (D):

- Represents the set of destination ports where demand must be met.

Set of Products (K):

- Represents different types of products produced by plants.

Derived Set of Routes (R):

- Represents valid routes from plants to ports where shipments can occur.

Mathematical Model/Parameters & Decision Variables

- `CAPACITY`:
Maximum capacity of each plant.
- `SHIPMENT`:
Decision variable representing the quantity shipped from each plant.
- `TOTAL_DEMAND`:
Total demand at the destination port.
- `COST`:
Transportation cost associated with each plant.

Objective Function

The objective of the current supply chain optimization model is to **minimize the total transportation cost**, which can be expressed as:

Minimize: `Total_Cost = ∑ (Cost * Shipment)`.

$$\text{Minimize: } Z = \sum_{i \in \text{PLANTS}} \sum_{j \in \text{PORTS}} C_{ij} \cdot X_{ij}$$

Constraints

Demand Constraint:

Ensure that the total demand at each port is met:

Capacity Constraint:

Ensure that the quantity shipped from each plant does not exceed its capacity:

Excess Capacity Constraint:

Allow plants to exceed their nominal capacity if necessary:

Non-Negativity Constraint:

Ensure that the quantities shipped are non-negative:

Assumptions

The following assumptions were made to develop a practical and solvable supply chain optimization model:

- **Assumption 1:** Each plant can only ship its available capacity, and the capacity values are considered fixed and known.
- **Assumption 2:** The transportation cost between each plant and the destination port is constant and does not vary with shipment size.
- **Assumption 3:** Demand at the destination port is fully deterministic and must be satisfied exactly unless plant capacity limits prevent it.
- **Assumption 4:** Shipments can only be sent to a single destination (PORT09), as defined by the problem.

Assumptions

- **Assumption 5:** Partial shipments from plants are allowed, and the decision variables are modeled as continuous values (not integers), meaning fractional shipments are permitted.
- **Assumption 6:** There are no delays, disruptions, or changes in the transportation process that could affect delivery times or costs.
- **Assumption 7:** There are no penalties or additional costs for unmet demand or unused plant capacity in the objective function, unless specifically modeled for sensitivity analysis.
- **Assumption 8:** The problem considers only direct shipment costs from plants to the destination, and does not include other supply chain costs such as warehousing or handling.

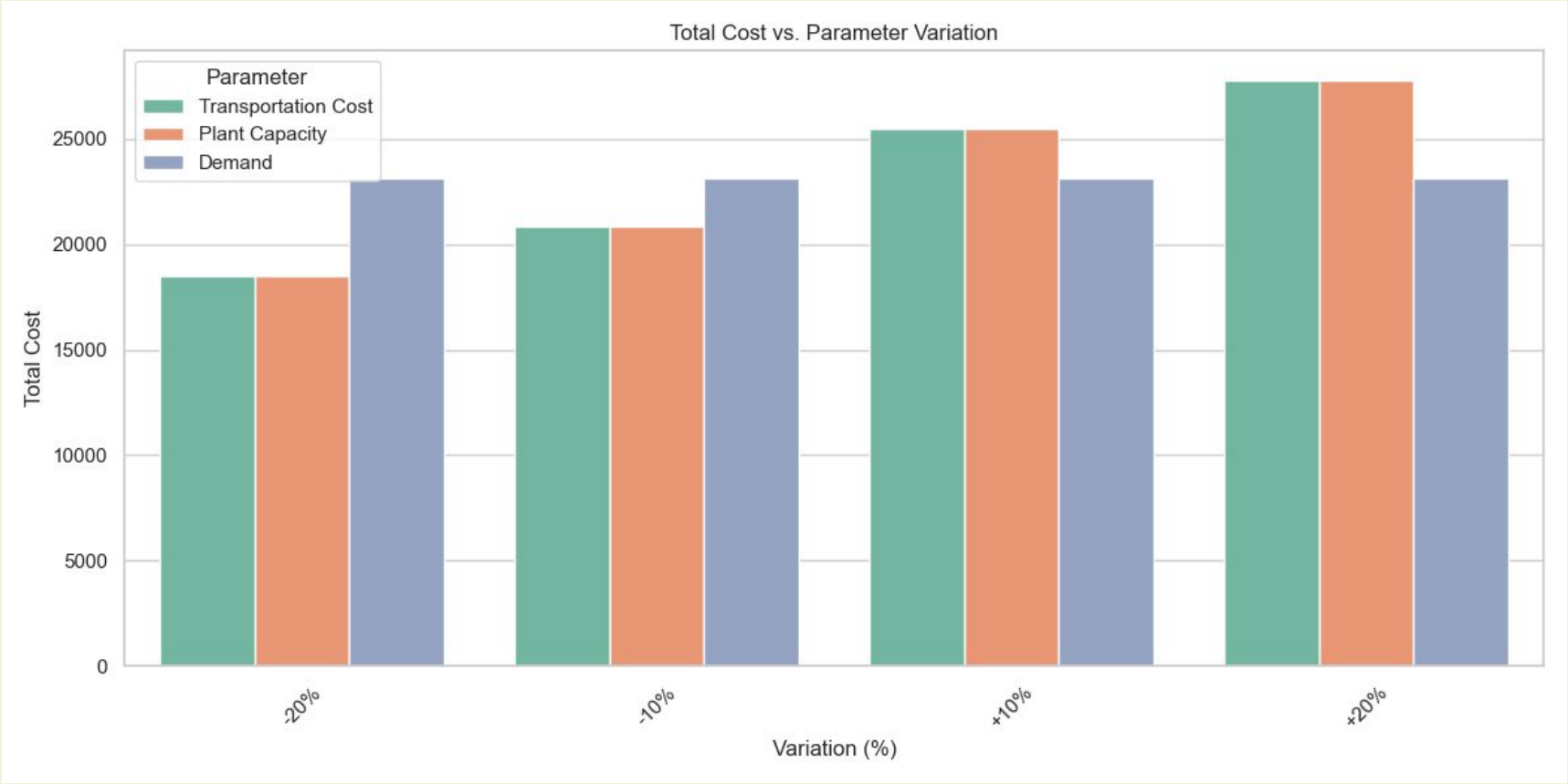
Sensitivity Analysis

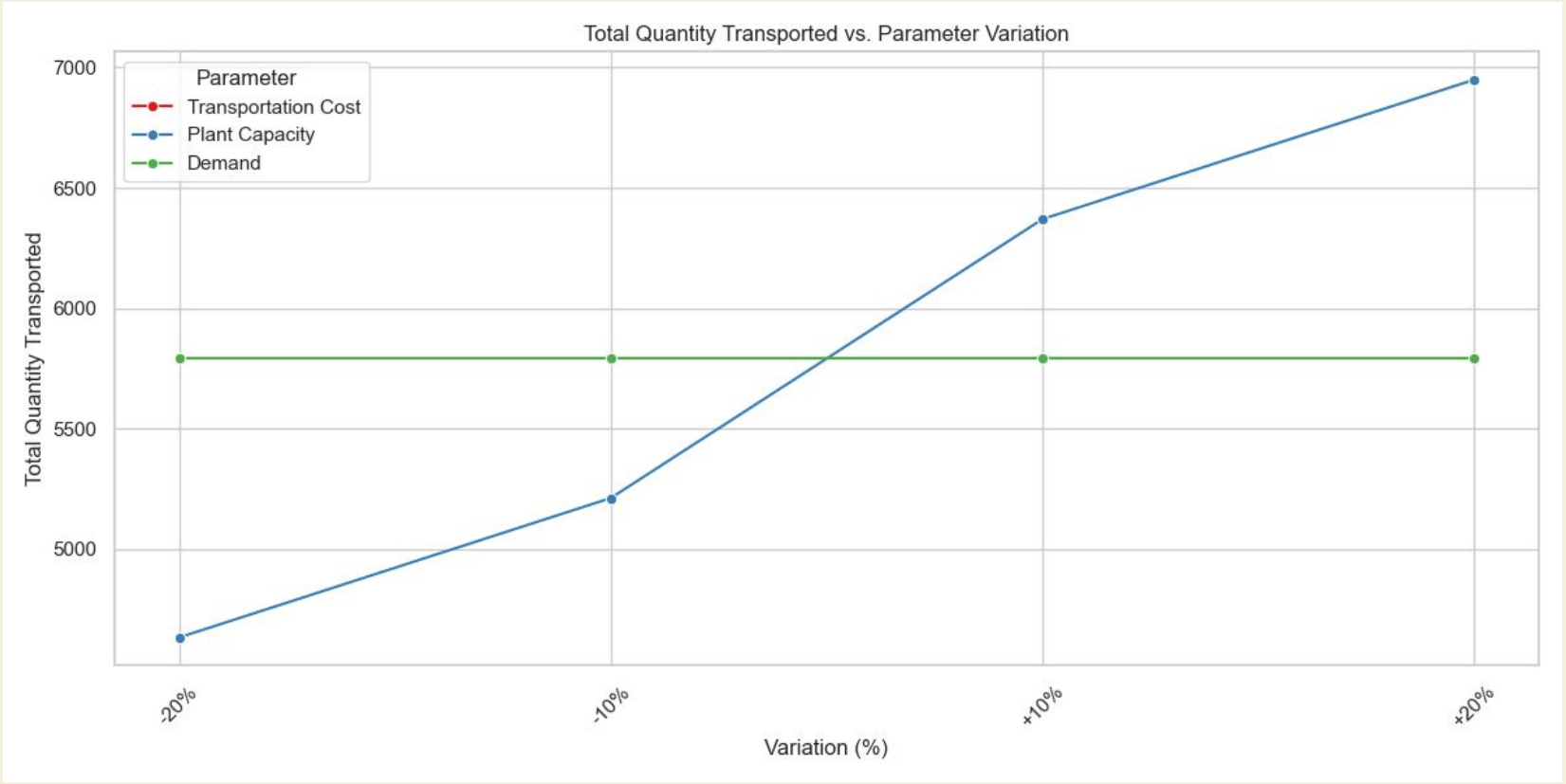
Sensitivity analysis was performed using SymPy to examine the impact of varying parameters on the total cost, total quantity, and the number of plants used. The following parameters were varied by ±10% and ±20%:

- Transportation cost
- Plant capacity
- Demand

Sensitivity Analysis

Parameter	Variation	Total Cost	Total Quantity	Num Plants Used
Transportation Cost	-20%	18517.36928	5791.0	19
Transportation Cost	-10%	20832.040439999997	5791.0	19
Transportation Cost	+10%	25461.382759999997	5791.0	19
Transportation Cost	+20%	27776.053919999995	5791.0	19
Plant Capacity	-20%	18517.369280000003	4632.8	19
Plant Capacity	-10%	20832.040440000033	5211.9000000000015	19
Plant Capacity	+10%	25461.382759999986	6370.099999999999	19
Plant Capacity	+20%	27776.053920000017	6949.200000000001	19
Demand	-20%	23146.711600000002	5791.0	19
Demand	-10%	23146.711600000002	5791.0	19
Demand	+10%	23146.711600000002	5791.0	19
Demand	+20%	23146.711600000002	5791.0	19


Total Cost vs. Parameter Variation
Parameter
Transportation Cost
Plant Capacity
Demand
Total Cost
25000
20000
15000
10000
5000
0
-20%
-10%
+10%
+20%
Variation (%)

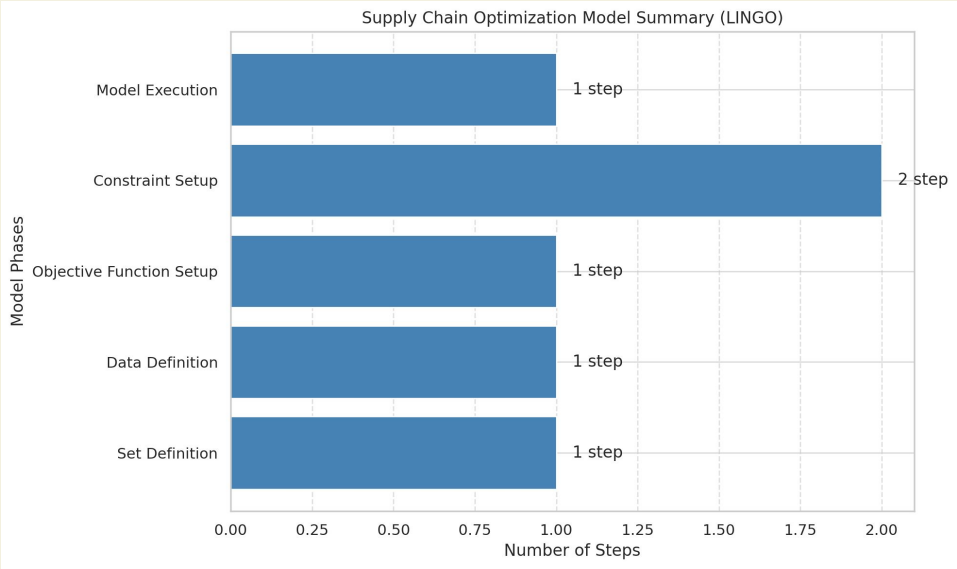

Total Quantity Transported vs. Parameter Variation
Parameter
Transportation Cost
Plant Capacity
Demand
Total Quantity Transported
7000
6500
6000
5500
5000
-20%
-10%
+10%
+20%
Variation (%)

1. Introduction
2. Model
3. **Code Summary**
4. Conclusion
5. Discussion

or1_project_SCM_model.lg4

This LINGO model solves a supply chain optimization problem by minimizing transportation costs under capacity and demand constraints. The main steps are:

1. **Set Definition:**
   Defines the sets for plants and destinations, along with their attributes (`CAPACITY`, `SHIPMENT`, and `COST`).
2. **Data Definition:**
   Specifies plant capacities, corrected freight costs, and total demand.
3. **Objective Function Setup:**
   Defines the objective function to minimize the total transportation cost.
4. **Constraint Setup:**
   - Ensures that shipments from each plant do not exceed its capacity.
   - Ensures total shipments meet the specified demand.
5. **Model Execution:**
   The model is executed, and the optimal solution is determined.

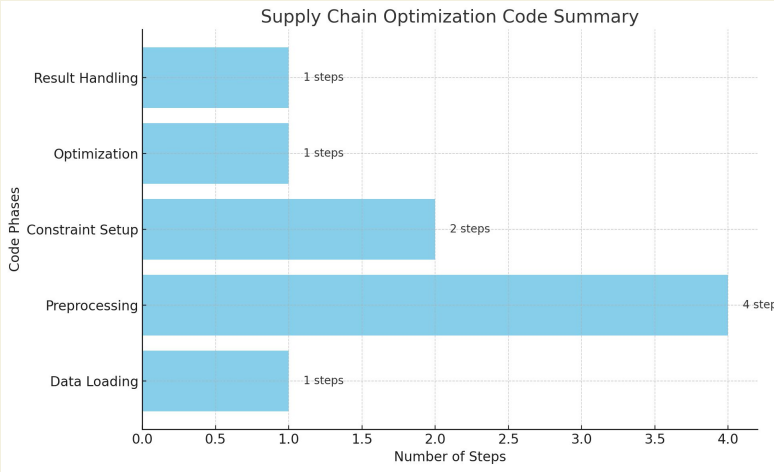


supply_chain.py

This Python code optimizes supply chain logistics using linear programming. It follows these key phases:

1. **Data Loading:**
   The program reads datasets containing order lists, freight rates, plant capacities, and plant-port mappings.
2. **Preprocessing:**
   Demand is aggregated by destination, plant capacities are extracted, and relevant freight costs are filtered to create the cost vector.
3. **Constraint Setup:**
   Constraints ensure shipments do not exceed plant capacities and total supply matches the demand at the port.
4. **Optimization:**
   Using `scipy.optimize.linprog`, the program solves for the minimum transportation cost while respecting constraints.
5. **Output:**
   The optimal shipment quantities and total cost are displayed, with logging for progress and error tracking.

This approach efficiently minimizes transportation costs while meeting demand requirements.

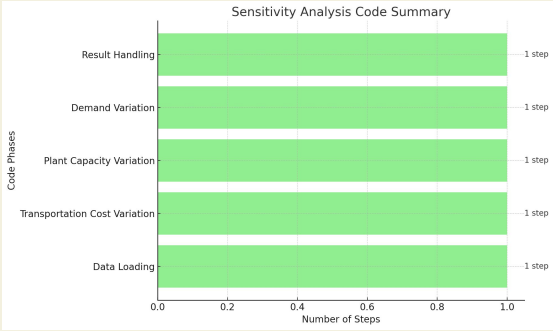


sensitivity_analysis.py

This Python code performs **sensitivity analysis** on a supply chain optimization model by varying key parameters (transportation costs, plant capacities, and demand) to evaluate their impact on total cost and solution robustness.

Phase 1: Data Loading

- The program reads Excel sheets containing order lists, freight rates, warehouse capacities, and plant-port mappings.



Phase 2: Sensitivity Analysis

1. **Transportation Cost Variation:**
   Freight rates are increased or decreased by ±10% and ±20%, and the supply chain optimization is rerun for each variation.
2. **Plant Capacity Variation:**
   The capacities of all plants are scaled by ±10% and ±20%, and the optimization is performed to observe changes in total cost and quantities.
3. **Demand Variation:**
   Demand is adjusted by ±10% and ±20%, and the resulting impact on the total cost is analyzed.

Phase 3: Result Handling

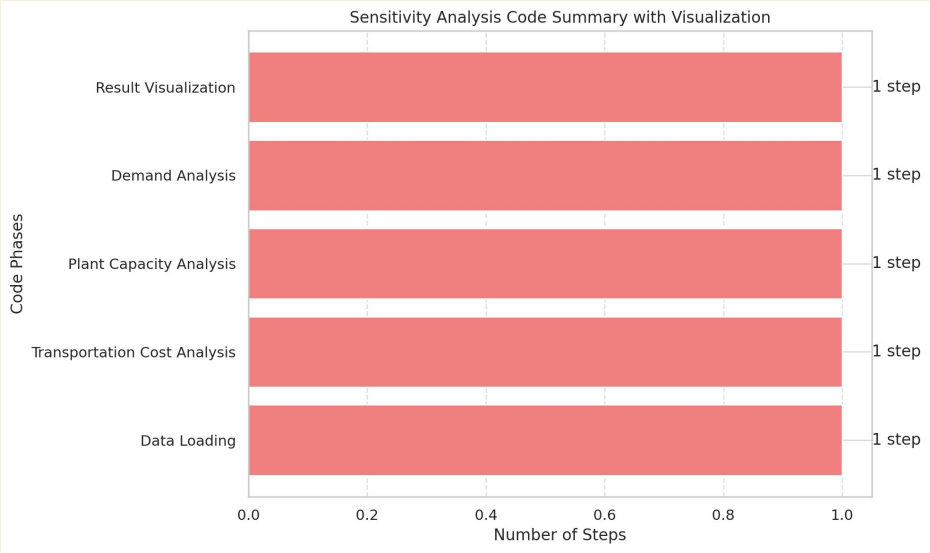
- For each variation, the solution is stored, and key metrics (total cost, total quantity, and number of plants used) are recorded.
- The results are saved as a CSV file for further analysis.

sensitivity_analysis_visualization.py

This Python code performs sensitivity analysis and visualizes the results for key parameters in a supply chain optimization model. The main steps are:

1. **Data Loading:**
   The program reads datasets containing order lists, freight rates, and plant capacities.
2. **Parameter Variation:**
   The code varies transportation costs, plant capacities, and demand by ±10% and ±20% to assess their impact.
3. **Result Recording:**
   For each variation, total cost, total quantity transported, and the number of plants used are recorded.
4. **Visualization:**
   Two plots are generated to display how total cost and total quantity change with parameter variations.

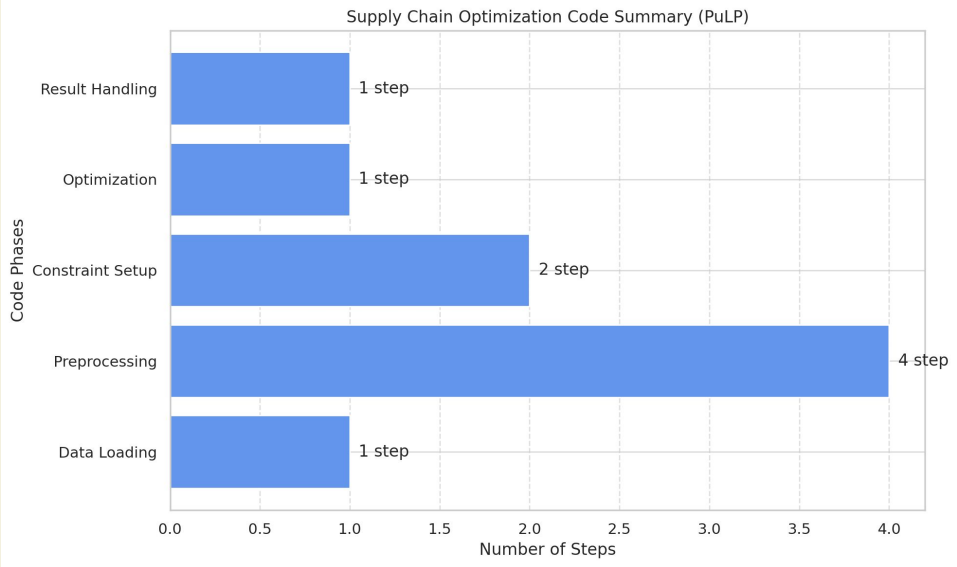
This approach helps evaluate the robustness of the supply chain model under different scenarios.



or1_project_pulp.py

This Python code solves a supply chain optimization problem using linear programming with PuLP. The main steps are:

1. **Data Loading:**
   Datasets containing order lists, freight rates, plant capacities, and plant-port mappings are loaded.
2. **Preprocessing:**
   The code aggregates demand by destination, extracts plant capacities, maps plants to ports, filters relevant freight rates, and creates a cost dictionary for valid origin-destination pairs.
3. **Constraint Setup:**
   - **Supply Constraints:** Ensure shipments from each plant do not exceed its capacity.
   - **Demand Constraint:** Ensure total shipments meet the total supply.
4. **Optimization:**
   The model minimizes the total transportation cost using PuLP's linear programming solver.
5. **Result Handling:**
   The optimal shipment quantities and total cost are displayed.



test_main_supply_chain.py

This Python code tests the supply chain optimization function using synthetic and real datasets. The main steps are:

1. **Fixture Creation:**
   Synthetic datasets with different scenarios (zero demand and synthetic data with demand) are created.
2. **Data Writing:**
   The synthetic data is written to Excel files for testing.
3. **Testing with No Demand:**
   The function is tested to ensure that it returns zero cost and no shipments when demand is zero.
4. **Testing with Synthetic Data:**
   The function is tested with a non-zero demand scenario to ensure correct shipment quantities and positive total cost.
5. **Testing with Real Data:**
   If a real dataset is available, the function is tested to ensure that it returns a valid solution with positive cost.

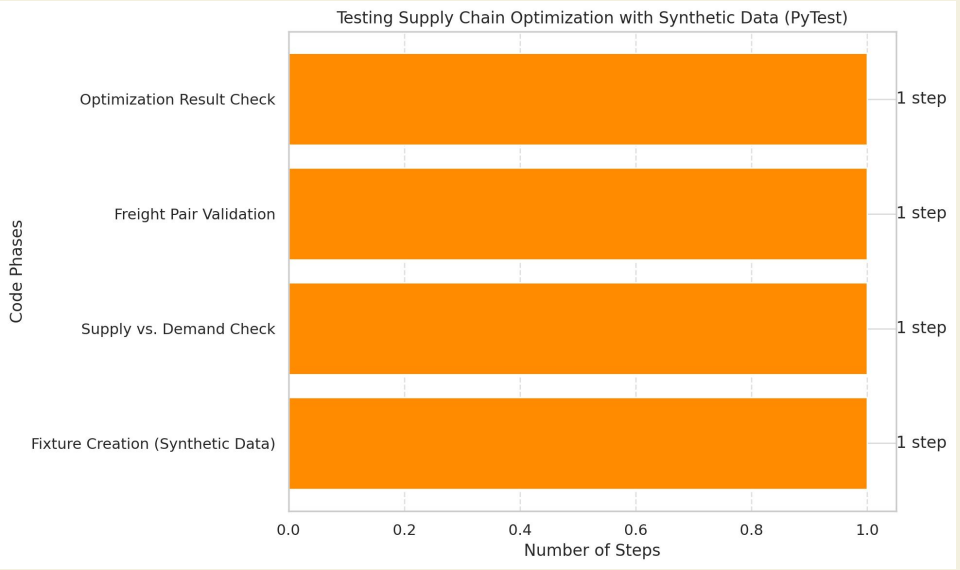
This approach ensures that the optimization function behaves correctly under various scenarios and data conditions.

test_or1_supply_chain.py

This Python code uses synthetic data to test key aspects of supply chain optimization. The main steps are:

1. **Fixture Creation:**
   Synthetic datasets for order lists, plant capacities, plant-port mappings, and freight rates are generated.
2. **Supply vs. Demand Check:**
   A test ensures that total supply is correctly compared against total demand.
3. **Freight Pair Validation:**
   The code verifies that valid plant-to-destination freight pairs are correctly identified.
4. **Optimization Result Check:**
   The linear programming optimization is tested to ensure it reports infeasibility when supply is insufficient.

This approach ensures robust validation of the optimization function under controlled scenarios.



1. Introduction
2. Model
3. Code Summary
4. **Conclusion**
5. Discussion

Optimal Solution

- Objective value: 23146.71
- The optimization model successfully identified the minimum transportation cost required to meet the demand at the destination port while adhering to plant capacities.
- The sensitivity analysis demonstrated that variations in transportation costs and plant capacities significantly affect the total cost, whereas demand variations did not impact the total cost, indicating robustness of the model concerning demand changes.

```
INFO:root:Datasets loaded successfully.
INFO:root:Optimization completed successfully.
Optimal solution: {('PLANT01', 'PORT09'): 1070.0, ('PLANT02', 'PORT09'): 138.0, ('PLANT03', 'POR
T09'): 1013.0, ('PLANT04', 'PORT09'): 554.0, ('PLANT05', 'PORT09'): 385.0, ('PLANT06', 'PORT09')
: 49.0, ('PLANT07', 'PORT09'): 265.0, ('PLANT08', 'PORT09'): 14.0, ('PLANT09', 'PORT09'): 11.0,
('PLANT10', 'PORT09'): 118.0, ('PLANT11', 'PORT09'): 332.0, ('PLANT12', 'PORT09'): 209.0, ('PLAN
T13', 'PORT09'): 490.0, ('PLANT14', 'PORT09'): 549.0, ('PLANT15', 'PORT09'): 11.0, ('PLANT16', '
PORT09'): 457.0, ('PLANT17', 'PORT09'): 8.0, ('PLANT18', 'PORT09'): 111.0, ('PLANT19', 'PORT09')
: 7.0}
Total cost: 23146.711600000002
```

Limitations

Single-Port Constraint:

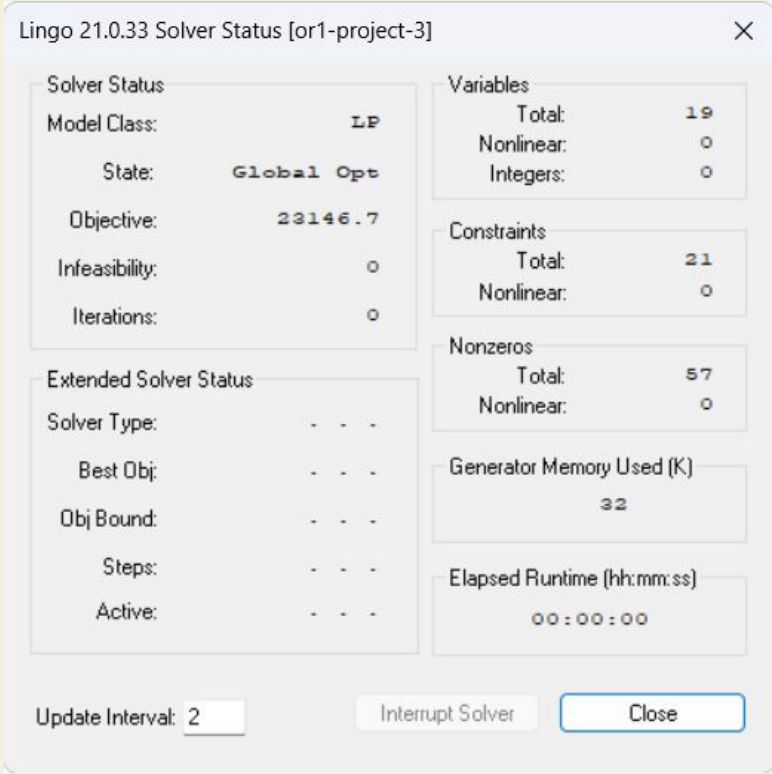
- The current model assumes that all shipments go to a single port (PORT09). This limits flexibility, as in a real-world scenario, multiple ports may be available, potentially reducing transportation costs.

Fixed Demand and Capacities:

- Demand at the port and capacities at the plants are treated as fixed values. In practice, both demand and plant capacities can vary due to supply chain disruptions, maintenance, or changes in market demand.

Transportation Costs Assumed Constant:

- The model assumes that transportation costs are constant. However, real-world transportation costs are dynamic and can be influenced by factors such as fuel prices, traffic congestion, and seasonal demand.



Limitations

No Consideration for Multi-Period Planning:

- The model is single-period, meaning it only plans for a single day or a specific period. In reality, supply chain decisions are often made over multiple periods (e.g., weeks, months) to optimize long-term performance.

No Fixed Costs for Opening/Closing Plants:

- The model does not include fixed costs for opening or closing plants. In real-world scenarios, opening or operating a plant involves fixed costs, which should be considered in the optimization process.

No Penalties for Excess Capacity:

- While the model includes excess capacity as a slack variable, it does not penalize unused capacity. Excess capacity can represent wasted resources and should ideally be minimized or penalized.

```
Status: Optimal
Total Cost: 23147.330000000005
PLANT15 -> PORT09: 11.0
PLANT17 -> PORT09: 8.0
PLANT18 -> PORT09: 111.0
PLANT05 -> PORT09: 385.0
PLANT02 -> PORT09: 138.0
PLANT01 -> PORT09: 1070.0
PLANT06 -> PORT09: 49.0
PLANT10 -> PORT09: 118.0
PLANT07 -> PORT09: 265.0
PLANT14 -> PORT09: 549.0
PLANT16 -> PORT09: 457.0
PLANT12 -> PORT09: 209.0
PLANT11 -> PORT09: 332.0
PLANT09 -> PORT09: 11.0
PLANT03 -> PORT09: 1013.0
PLANT13 -> PORT09: 490.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT04 -> PORT09: 554.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT19 -> PORT09: 7.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT19 -> PORT09: 7.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT04 -> PORT09: 554.0
```

How to improve

Incorporate Multiple Ports:

- Extend the model to include multiple destination ports, allowing shipments to be distributed optimally across various ports based on proximity and cost.

Dynamic Demand and Capacities:

- Introduce variability in demand and plant capacities using probabilistic distributions or scenario-based planning.

Include Variable Transportation Costs:

- Use a dynamic transportation cost model that reflects real-world variations in costs due to factors like fuel price fluctuations or traffic delays.

```
Status: Optimal
Total Cost: 23147.330000000005
PLANT15 -> PORT09: 11.0
PLANT17 -> PORT09: 8.0
PLANT18 -> PORT09: 111.0
PLANT05 -> PORT09: 385.0
PLANT02 -> PORT09: 138.0
PLANT01 -> PORT09: 1070.0
PLANT06 -> PORT09: 49.0
PLANT10 -> PORT09: 118.0
PLANT07 -> PORT09: 265.0
PLANT14 -> PORT09: 549.0
PLANT16 -> PORT09: 457.0
PLANT12 -> PORT09: 209.0
PLANT11 -> PORT09: 332.0
PLANT09 -> PORT09: 11.0
PLANT03 -> PORT09: 1013.0
PLANT13 -> PORT09: 490.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT04 -> PORT09: 554.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT19 -> PORT09: 7.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT19 -> PORT09: 7.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT04 -> PORT09: 554.0
```

How to improve

Multi-Period Optimization:

- Extend the model to a multi-period framework where decisions are optimized over several time periods, incorporating inventory holding costs and supply chain disruptions.

Add Fixed Costs for Plant Operations:

- Introduce binary decision variables to model whether a plant should be opened or closed, and include associated fixed operating costs in the objective function.

Penalize Excess Capacity:

- Add a penalty term in the objective function for excess capacity to discourage underutilization of plant resources.

```
Status: Optimal
Total Cost: 23147.330000000005
PLANT15 -> PORT09: 11.0
PLANT17 -> PORT09: 8.0
PLANT18 -> PORT09: 111.0
PLANT05 -> PORT09: 385.0
PLANT02 -> PORT09: 138.0
PLANT01 -> PORT09: 1070.0
PLANT06 -> PORT09: 49.0
PLANT10 -> PORT09: 118.0
PLANT07 -> PORT09: 265.0
PLANT14 -> PORT09: 549.0
PLANT16 -> PORT09: 457.0
PLANT12 -> PORT09: 209.0
PLANT11 -> PORT09: 332.0
PLANT09 -> PORT09: 11.0
PLANT03 -> PORT09: 1013.0
PLANT13 -> PORT09: 490.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT04 -> PORT09: 554.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT19 -> PORT09: 7.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT19 -> PORT09: 7.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT04 -> PORT09: 554.0
```

1. Introduction
2. Model
3. Code Summary
4. Conclusion
5. Discussion

Comparison of Results

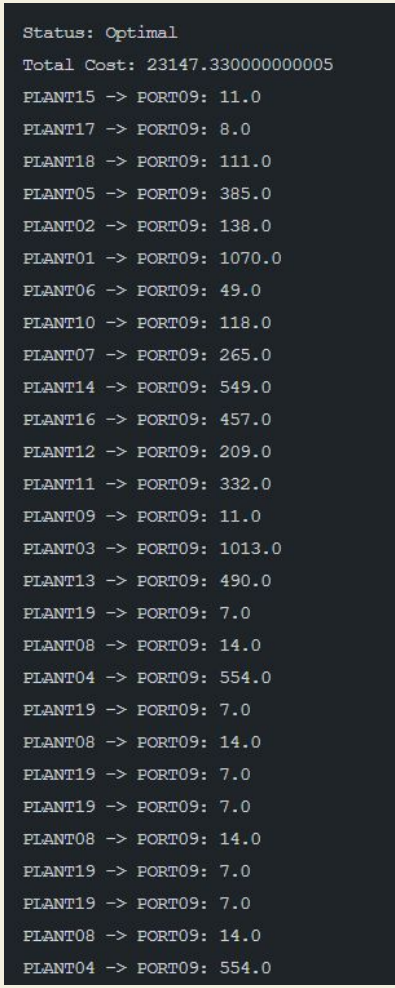
- The objective of this section is to compare the results obtained from solving a supply chain optimization problem using three different methods:
  - PuLP (Python), SymPy (Linear Algebra), and LINGO (Optimization Software).
- The goal of the optimization is to minimize the total transportation cost while distributing the available supply from multiple plants to meet the demand at a single destination port (PORT09).
- All three methods produced an optimal solution with a total transportation cost of 23,146.71. The shipment quantities from each plant were identical across all methods.

```
Status: Optimal
Total Cost: 23147.330000000005
PLANT15 -> PORT09: 11.0
PLANT17 -> PORT09: 8.0
PLANT18 -> PORT09: 111.0
PLANT05 -> PORT09: 385.0
PLANT02 -> PORT09: 138.0
PLANT01 -> PORT09: 1070.0
PLANT06 -> PORT09: 49.0
PLANT10 -> PORT09: 118.0
PLANT07 -> PORT09: 265.0
PLANT14 -> PORT09: 549.0
PLANT16 -> PORT09: 457.0
PLANT12 -> PORT09: 209.0
PLANT11 -> PORT09: 332.0
PLANT09 -> PORT09: 11.0
PLANT03 -> PORT09: 1013.0
PLANT13 -> PORT09: 490.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT04 -> PORT09: 554.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT19 -> PORT09: 7.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT19 -> PORT09: 7.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT04 -> PORT09: 554.0
```

Comparison of Results

1. PuLP (Python)

- PuLP is a Python library for linear programming. The problem was formulated as a linear programming model with:
  - Decision Variables: Representing the shipment quantities from each plant to PORT09.
  - Objective Function: Minimizing the total transportation cost.
  - Constraints:
    - Ensuring that the total shipments from each plant do not exceed its capacity.
    - Ensuring that the total supply matches the total demand.



Comparison of Results

2. SymPy (Linear Algebra)

- SymPy was used to solve the problem symbolically by:
  - Representing the transportation cost as a matrix equation.
  - Applying linear algebra techniques to find the optimal solution.

```
Status: Optimal
Total Cost: 23147.330000000005
PLANT15 -> PORT09: 11.0
PLANT17 -> PORT09: 8.0
PLANT18 -> PORT09: 111.0
PLANT05 -> PORT09: 385.0
PLANT02 -> PORT09: 138.0
PLANT01 -> PORT09: 1070.0
PLANT06 -> PORT09: 49.0
PLANT10 -> PORT09: 118.0
PLANT07 -> PORT09: 265.0
PLANT14 -> PORT09: 549.0
PLANT16 -> PORT09: 457.0
PLANT12 -> PORT09: 209.0
PLANT11 -> PORT09: 332.0
PLANT09 -> PORT09: 11.0
PLANT03 -> PORT09: 1013.0
PLANT13 -> PORT09: 490.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT04 -> PORT09: 554.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT19 -> PORT09: 7.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT19 -> PORT09: 7.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT04 -> PORT09: 554.0
```

Comparison of Results

3. LINGO (Optimization Software)

- LINGO is a specialized optimization software. The problem was modeled using LINGO syntax with:
  - Sets for plants and the destination port.
  - Parameters for plant capacities and freight costs.
  - An objective function and constraints similar to those in PuLP.

```
Status: Optimal
Total Cost: 23147.330000000005
PLANT15 -> PORT09: 11.0
PLANT17 -> PORT09: 8.0
PLANT18 -> PORT09: 111.0
PLANT05 -> PORT09: 385.0
PLANT02 -> PORT09: 138.0
PLANT01 -> PORT09: 1070.0
PLANT06 -> PORT09: 49.0
PLANT10 -> PORT09: 118.0
PLANT07 -> PORT09: 265.0
PLANT14 -> PORT09: 549.0
PLANT16 -> PORT09: 457.0
PLANT12 -> PORT09: 209.0
PLANT11 -> PORT09: 332.0
PLANT09 -> PORT09: 11.0
PLANT03 -> PORT09: 1013.0
PLANT13 -> PORT09: 490.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT04 -> PORT09: 554.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT19 -> PORT09: 7.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT19 -> PORT09: 7.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT04 -> PORT09: 554.0
```

Comparison of Methods

1. Modeling Complexity:

- PuLP: Required setting up decision variables, objective function, and constraints explicitly in Python.
- SymPy: Required formulating the problem as a matrix equation and solving it using linear algebra.
- LINGO: Involved defining sets, parameters, and constraints using LINGO's syntax.

```
Status: Optimal
Total Cost: 23147.330000000005
PLANT15 -> PORT09: 11.0
PLANT17 -> PORT09: 8.0
PLANT18 -> PORT09: 111.0
PLANT05 -> PORT09: 385.0
PLANT02 -> PORT09: 138.0
PLANT01 -> PORT09: 1070.0
PLANT06 -> PORT09: 49.0
PLANT10 -> PORT09: 118.0
PLANT07 -> PORT09: 265.0
PLANT14 -> PORT09: 549.0
PLANT16 -> PORT09: 457.0
PLANT12 -> PORT09: 209.0
PLANT11 -> PORT09: 332.0
PLANT09 -> PORT09: 11.0
PLANT03 -> PORT09: 1013.0
PLANT13 -> PORT09: 490.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT04 -> PORT09: 554.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT19 -> PORT09: 7.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT19 -> PORT09: 7.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT04 -> PORT09: 554.0
```

Comparison of Methods

2. Ease of Use:

- PuLP: Easy to set up for those familiar with Python.
- SymPy: Requires knowledge of linear algebra and symbolic computation.
- LINGO: Specialized for optimization problems but requires learning its syntax.

```
Status: Optimal
Total Cost: 23147.330000000005
PLANT15 -> PORT09: 11.0
PLANT17 -> PORT09: 8.0
PLANT18 -> PORT09: 111.0
PLANT05 -> PORT09: 385.0
PLANT02 -> PORT09: 138.0
PLANT01 -> PORT09: 1070.0
PLANT06 -> PORT09: 49.0
PLANT10 -> PORT09: 118.0
PLANT07 -> PORT09: 265.0
PLANT14 -> PORT09: 549.0
PLANT16 -> PORT09: 457.0
PLANT12 -> PORT09: 209.0
PLANT11 -> PORT09: 332.0
PLANT09 -> PORT09: 11.0
PLANT03 -> PORT09: 1013.0
PLANT13 -> PORT09: 490.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT04 -> PORT09: 554.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT19 -> PORT09: 7.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT19 -> PORT09: 7.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT04 -> PORT09: 554.0
```

Comparison of Methods

3. Performance:

- All methods found the optimal solution efficiently, with negligible runtime differences for this problem size.

```
Status: Optimal
Total Cost: 23147.330000000005
PLANT15 -> PORT09: 11.0
PLANT17 -> PORT09: 8.0
PLANT18 -> PORT09: 111.0
PLANT05 -> PORT09: 385.0
PLANT02 -> PORT09: 138.0
PLANT01 -> PORT09: 1070.0
PLANT06 -> PORT09: 49.0
PLANT10 -> PORT09: 118.0
PLANT07 -> PORT09: 265.0
PLANT14 -> PORT09: 549.0
PLANT16 -> PORT09: 457.0
PLANT12 -> PORT09: 209.0
PLANT11 -> PORT09: 332.0
PLANT09 -> PORT09: 11.0
PLANT03 -> PORT09: 1013.0
PLANT13 -> PORT09: 490.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT04 -> PORT09: 554.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT19 -> PORT09: 7.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT19 -> PORT09: 7.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT04 -> PORT09: 554.0
```

Comparison of Methods

- All three methods—PuLP, SymPy, and LINGO—produced the same optimal solution with a total transportation cost of 23,146.71. This confirms the correctness and consistency of the modeling approaches.

  Each method has its strengths:
  - PuLP: Best for users familiar with Python and wanting a programmable approach.
  - SymPy: Suitable for symbolic and algebraic problem-solving.
  - LINGO: Ideal for large-scale optimization problems where specialized solvers are needed.

```
Status: Optimal
Total Cost: 23147.330000000005
PLANT15 -> PORT09: 11.0
PLANT17 -> PORT09: 8.0
PLANT18 -> PORT09: 111.0
PLANT05 -> PORT09: 385.0
PLANT02 -> PORT09: 138.0
PLANT01 -> PORT09: 1070.0
PLANT06 -> PORT09: 49.0
PLANT10 -> PORT09: 118.0
PLANT07 -> PORT09: 265.0
PLANT14 -> PORT09: 549.0
PLANT16 -> PORT09: 457.0
PLANT12 -> PORT09: 209.0
PLANT11 -> PORT09: 332.0
PLANT09 -> PORT09: 11.0
PLANT03 -> PORT09: 1013.0
PLANT13 -> PORT09: 490.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT04 -> PORT09: 554.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT19 -> PORT09: 7.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT19 -> PORT09: 7.0
PLANT19 -> PORT09: 7.0
PLANT08 -> PORT09: 14.0
PLANT04 -> PORT09: 554.0
```

References

- Brechter, L. (n.d.). Supply chain data [Data set]. Kaggle. Retrieved January 5, 2025, from https://www.kaggle.com/datasets/laurinbrechter/supply-chain-data