OR LAB PROJECT

Group -3

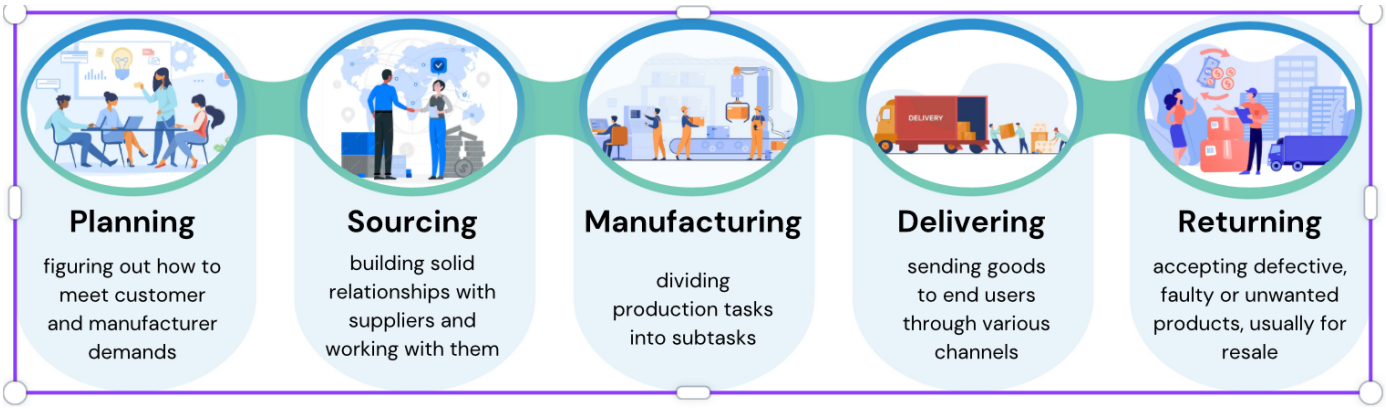
Team Members

SAMSRHITHA BURA (22IM10012)
INALA JANNAVI MEHAR SAI (22IM10016)
SEMANTI GHOSH (22IM10036)
VANKUDOTH SAI PRIYA (22IM10043)

Table of Contents

Introduction

Supply chain optimization plays a pivotal role in the global industry by enhancing operational efficiency, reducing costs, and improving customer satisfaction. In today's interconnected world, where businesses operate across borders and continents, supply chains have become increasingly complex and dynamic. Optimization strategies leverage advanced technologies, data analytics, and strategic planning to streamline the flow of goods, information, and capital across the entire supply chain network. From procurement and manufacturing to distribution and logistics, optimization efforts aim to synchronize processes, minimize lead times, optimize inventory levels, and mitigate risks. By optimizing supply chains, companies can respond more effectively to market demand fluctuations, reduce wastage, improve sustainability practices, and gain a competitive edge in the global marketplace.



Business Problems

Supply chain optimization has wide range of applications in different industrial domains and a few examples:

Retail Supply Chain Optimization: A retail chain with multiple warehouses and stores needs to efficiently allocate products from warehouses to stores while minimizing transportation costs and ensuring that each store meets its demand. This optimization problem helps in determining the most cost-effective way to distribute products.

Manufacturing Inventory Management: A manufacturing company with multiple production facilities needs to decide how much inventory to keep at each location to meet future demand while minimizing inventory holding costs. This problem helps in optimizing inventory levels across different locations in the supply chain.

Logistics and Distribution: A logistics company needs to optimize its transportation routes and warehouse assignments to deliver goods to customers with minimal cost and time. This optimization problem helps in determining the most efficient distribution network and routing plan.

E-commerce Fulfillment: An e-commerce company needs to manage inventory across warehouses and fulfill customer orders efficiently. By solving this optimization problem, the company can minimize storage and transportation costs while ensuring timely delivery to customers.

Healthcare Supply Chain: Hospitals and healthcare facilities need to manage their supply chain effectively to ensure the availability of medical supplies while minimizing costs. This optimization problem helps in optimizing inventory levels and distribution routes for medical supplies.

Literature Review

The concept of supply chain optimization discussed in multiple recent papers. Supply chain optimization makes the best use of data analytics to find an optimal combination of factories and distribution centers to match supply and demand.
Because of the current surge in shipping costs, companies start to challenge their current footprint to adapt to the post-covid "New Normal".
In this report, a simple methodology using Linear Programming for Supply Chain Optimization considering Fixed production costs of facilities ($/Month), Variable production costs per unit produced ($/Unit), Shipping costs ($), Customer's demand (Units).
The current project takes reference from following papers/ articles:

- Kostikov E et. al. provides a model that allows for the central warehouse's optimal location and minimizes transportation costs from the central warehouse to sub-warehouses/branches located in individual EU countries for varying demands. [1]
- Optimization Toolbox of MATLAB provides algorithmic steps for Mixed Integer Linear Programming (MILP) [2]
- Igor Shvab provides a comprehensive approach for optimization modeling using SciPy, PuLP, Pyomo [3]
- Collins, E.J. provides the optimization technique using Markov Decision Process (MDP) [4]

Problem Statement

In an international manufacturing company, there are markets in 5 countries. The market can have a high-capacity or low-capacity production factory. The local market's demand (market in the same country) is always met first. Then,
If there is excess, it is shipped to another country where there is a requirement.
The objective - The total cost consisting of production cost and freight cost must be minimized.

Model

The model is based on following assumptions:

- Fixed and Variable costs are independent of the period of production.
- 1 shipping container can contain 1000 units.
- Demand is assumed to be constant every month.

- **Manufacturing Facility Fixed Costs**

    - Capital Expenditure for the equipment (Machines, Storage etc.)
    - Utilities (Electricity, Water etc.)
    - Factory management, administrative staff
    - Space Rental

- **Production Variable Costs**

    - Production lines operators
    - Raw materials

Notations and Model Parameters

i = 1 to 5, representing 5 markets {USA, Germany, Japan, Brazil, India}.

s = 1, representing the high-capacity production factories.

2, representing the low-capacity production factories.

d_i – No of units demanded in market i per month.

C_{is} – The number of units produced in s^{th} production factory in i^{th} market.

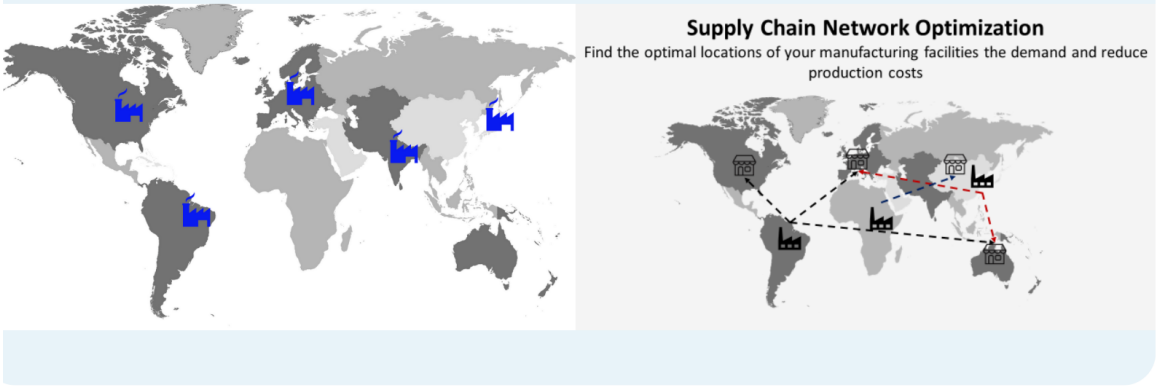
f_{is} – Fixed cost for the s^{th} production factory in i^{th} market per month.

vp_{is} – Variable production cost per unit produced in market i.

t_{ij} – Variable shipping cost (freight cost) from market i to market j per container.

x_{ij} – Number of units shifts from market i to market j.

y_{is} – Binary variables with states 0 or 1, indicating the establishment of production factory s in market i.



Objective Function

Minimize

Total variable cost of production + Total fixed cost of production + Total freight (transportation) cost.

Constraints

- Fulfilling the demand of customers in each market.
- Total no. of units shipped from market i to other markets is less than or equal to the capacity of units produced in i.

Mathematical Formulation

$$\text{Minimise: } Z = \sum_{i=1}^{5}\sum_{s=1}^{2} f_{ij} * y_{is} * 1000 + \sum_{i=1}^{5}\sum_{j=1}^{5} x_{ij} * (vp_i + t_{ij} * 0.001)$$

$$\text{Subject to: } \sum_{i=1}^{5} x_{ij} = d_j \quad \forall \quad j = 1, 2, \ldots, 5$$

$$\sum_{j=1}^{5} x_{ij} = \sum_{s=1}^{2} C_{is} * y_{is} \quad \forall \quad i = 1, 2, \ldots 5$$

$$x_{ij} \geq 0 \quad \forall \quad i = 1, 2, \ldots 5, \; j = 1, 2, \ldots 5$$

$$y_{is} \in \{0, 1\}$$

Methodologies - Algorithms

This problem is solved using optimization technique. There are two modelling approaches:
Linear Model – It can be formulated using Multi Integer Linear Model (MILP). This is suitable for the above problem because:

⇒ Decision variables are not restricted to integers (e.g. fixed cost, variable cost) – not Integer programming.

⇒ All constraints and objective function are linear.

Non-linear Model - It is formulated using nonlinear program for complex problem. Markov Decision Process (MDP) with an effective convex reward function can be considered.

Linear model is applied in solving the problem.

Algorithmic Steps:
The following steps are followed in solving MILP problem:
Initially

- Reduce the size of the problem using Linear Program Processing.
- Solve the Initial relaxed (non-integer) problem using Linear Programming.
- Perform Mixed Integer Program Processing to tighten the LP relaxation of the mixed-integer problem.

Next Level

- Try Cut Generation to further tighten the LP relaxation of the mixed-integer problem.
- Try to find integer feasible solution using heuristics.

- Use a branch & bound algorithm to search iteratively for the optimal solution.

Methodologies – Tools

Key tools considered are given below:

- IBM ILog CPlex
- Python Packages
    - Scipy.optimize
    - PuLP
    - Pyomo
- Gurobi

For solving this problem, python based open-source library is used.
Key Considerations:

- IBM CPLEX or Gurobi can be used in python. But we use open-source optimization solver.
- Scipy is most supported one and the standard recursive approach can be implemented here.
- PuLP and Pyomo both follow same type of structure. They are easy to learn. PuLP supports nonlinear optimization as well.

In this implementation, PuLP package is used.

Results/Discussion

Data Used:

(Units/month)	Demand
USA	2 800 000
Germany	90 000
Japan	1 700 000
Brazil	145 000
India	160 000

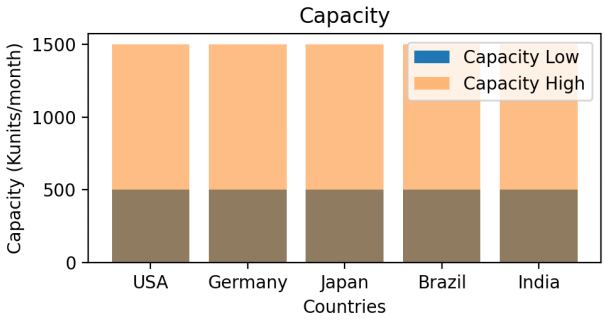
K$/month	Low	High
USA	6500	9500
Germany	4980	7270
Japan	6230	9100
Brazil	3230	4730
India	2110	6160

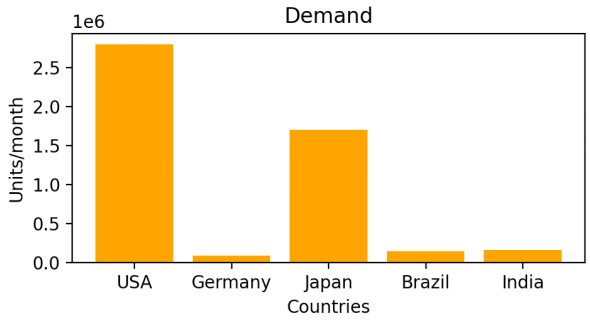
Capacity (kUnits/month)	Low	High
USA	500	1500
Germany	500	1500
Japan	500	1500
Brazil	500	1500
India	500	1500

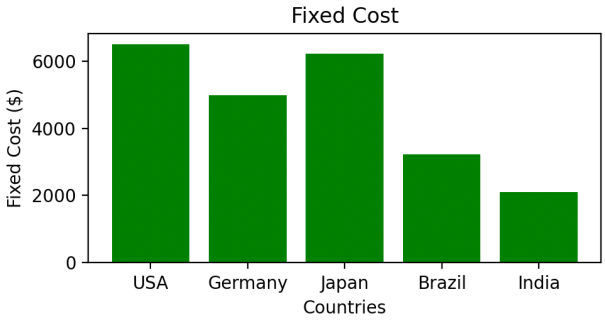
Freight Costs ($/Container)	USA	Germany	Japan	Brazil	India
USA	0	12250	1100	16100	8778
Germany	13335	0	8617	20244	10073
Japan	15400	22750	0	43610	14350
Brazil	16450	22050	28000	0	29750
India	13650	15400	24500	29400	0

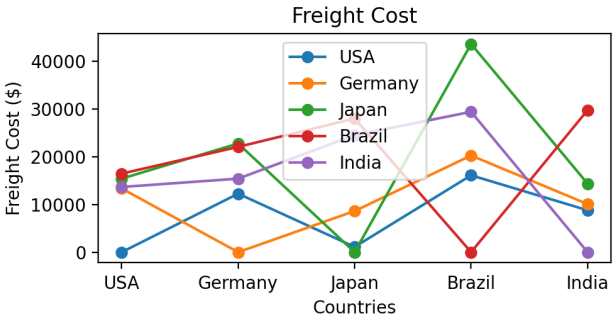
Variable Costs ($/Unit)	USA	Germany	Japan	Brazil	India
USA	12	12	12	12	12
Germany	13	13	13	13	13
Japan	10	10	10	10	10
Brazil	8	8	8	8	8
India	5	5	5	5	5

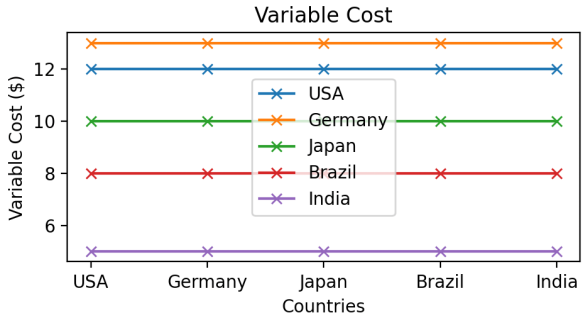
Plotting Input Data:











Results:

Objective value: 94431000.00000000
Enumerated nodes: 0
Total iterations: 10
Time (CPU seconds): 0.08
Time (Wallclock seconds): 0.08

Option for printingOptions changed from normal to all
Total time (CPU seconds): 0.09
(Wallclock seconds): 0.09

Total Costs = 94,431,000 ($/Month)

Status: Optimal
('Brazil','High') = 0.0
('Brazil','Low') = 1.0
('Germany','High') = 0.0
('Germany','Low') = 0.0
('India','High') = 1.0
('India','Low') = 0.0
('Japan','High') = 1.0
('Japan','Low') = 0.0
('USA','High') = 1.0
('USA','Low') = 0.0

('Brazil','Brazil') = 145000.0
('Brazil','Germany') = 0.0
('Brazil','India') = 0.0
('Brazil','Japan') = 0.0
('Brazil','USA') = 250000.0
('Germany','Brazil') = 0.0
('Germany','Germany') = 0.0
('Germany','India') = 0.0
('Germany','Japan') = 0.0
('Germany','USA') = 0.0
('India','Brazil') = 0.0
('India','Germany') = 90000.0
('India','India') = 160000.0
('India','Japan') = 0.0
('India','USA') = 1250000.0

('Japan','Brazil') = 0.0
('Japan','Germany') = 0.0
('Japan','India') = 0.0
('Japan','Japan') = 1500000.0
('Japan','USA') = 0.0
('USA','Brazil') = 0.0
('USA','Germany') = 0.0
('USA','India') = 0.0
('USA','Japan') = 200000.0
('USA','USA') = 1300000.0

Alternative Modelling Approach

As an alternative approach, we can model the above problem using Markov Decision Process (MDP) and the formulation has following constructs:

States: In our problem, the states represent the different configurations of production factories and goods in transit. Each state can be defined by the capacity of each production factory (high-capacity or low-capacity) in each country and the quantities of goods in transit between countries.

Actions: The actions represent the production decisions and transportation decisions. The production decisions involve choosing whether to produce goods in each country using high-capacity or low-capacity production factories. The transportation decisions involve choosing how many goods to transport between each pair of countries.

Transition Probabilities: The transition probabilities define the likelihood of transitioning from one state to another based on the chosen actions. These probabilities can be determined by historical data, logistic constraints, or estimated based on assumptions.

Rewards: The rewards represent the immediate benefits or costs associated with taking certain actions in each state. In this case, the rewards would be the production costs and freight costs incurred by producing goods and transporting them between countries.

Optimization Objective: The objective of the optimization is to find the policy (i.e., the sequence of actions) that minimizes the total cost consisting of production costs and freight costs over time, considering the stochastic nature of the system and the constraints on production capacities and transportation capacities.

To implement MDP-based optimization, we can apply reinforcement learning (Q-learning or policy iterations).

Conclusions

- The supply chain optimization model aimed to minimize total costs by efficiently allocating production and transportation resources across five countries.
- It successfully balanced demand fulfillment, capacity constraints, and cost minimization objectives.
- The optimal solution showed activation of production plants based on binary decisions and optimal transportation routes determined using continuous variables.
- Prioritizing local market demand before considering shipments to other countries ensured efficient resource utilization.
- Overall, the model enabled the company to operate cost-effectively and competitively in the global market.
- Markov Decision Process (MDP) implementation provides an alternative modeling approach. But it can be used for more complex optimization problem.

References

1. Optimization of E-Commerce Distribution Center Location, Authors: Kostikov E, Jilkova P, Stranska P K, Marketing and Management of Innovation, DOI: https://doi.org/10.21272/mmi.2021.2-14 (2021)
2. https://www.mathworks.com/help/optim/ug/mixed-integer-linear-programming-algorithms.html
3. https://medium.com/analytics-vidhya/optimization-modelling-in-python-scipy-pulp-and-pyomo-d392376109f4
4. https://people.maths.bris.ac.uk/~maejc/reports/umdp.pdf

Appendix

Program Listing – Input Data Plot

/**********Start**********/

```
import numpy as np
import matplotlib.pyplot as plt

# Data
countries = ['USA', 'Germany', 'Japan', 'Brazil', 'India']
capacity_low = [500, 500, 500, 500, 500]
capacity_high = [1500, 1500, 1500, 1500, 1500]
demand = [2800000, 90000, 1700000, 145000, 160000]
fixed_cost = [6500, 4980, 6230, 3230, 2110]
freight_cost = np.array([
    [0, 12250, 1100, 16100, 8778],
    [13335, 0, 8617, 20244, 10073],
    [15400, 22750, 0, 43610, 14350],
    [16450, 22050, 28000, 0, 29750],
    [13650, 15400, 24500, 29400, 0]
])
variable_cost = np.array([
    [12, 12, 12, 12, 12],
    [13, 13, 13, 13, 13],
    [10, 10, 10, 10, 10],
    [8, 8, 8, 8, 8],
    [5, 5, 5, 5, 5]
])

# Create separate plots for each category
plt.figure(figsize=(10, 8))

# Capacity plot
plt.subplot(3, 2, 1)
plt.bar(countries, capacity_low, label='Capacity Low')
plt.bar(countries, capacity_high, label='Capacity High', alpha=0.5)
plt.title('Capacity')
plt.xlabel('Countries')
plt.ylabel('Capacity (Kunits/month)')
plt.legend()

# Demand plot
plt.subplot(3, 2, 2)
plt.bar(countries, demand, color='orange')
plt.title('Demand')
plt.xlabel('Countries')
plt.ylabel('Units/month')
```

```
# Fixed Cost plot
plt.subplot(3, 2, 3)
plt.bar(countries, fixed_cost, color='green')
plt.title('Fixed Cost')
plt.xlabel('Countries')
plt.ylabel('Fixed Cost ($)')

# Freight Cost plot
plt.subplot(3, 2, 4)
for i, country in enumerate(countries):
    plt.plot(countries, freight_cost[i], marker='o', label=country)
plt.title('Freight Cost')
plt.xlabel('Countries')
plt.ylabel('Freight Cost ($)')
plt.legend()

# Variable Cost plot
plt.subplot(3, 2, 5)
for i, country in enumerate(countries):
    plt.plot(countries, variable_cost[i], marker='x', label=country)
plt.title('Variable Cost')
plt.xlabel('Countries')
plt.ylabel('Variable Cost ($)')
plt.legend()

plt.tight_layout()
plt.show()
```

/**********End**********/

Program Listing - Optimization

/**********Start**********/

```
import pandas as pd
from pulp import *

# Import Manufacturing Costs
manvar_costs = pd.read_excel('variable_costs.xlsx', index_col = 0)
# Import Freight Costs
freight_costs = pd.read_excel('freight_costs.xlsx', index_col = 0)
# Variable Cost
var_cost = freight_costs/1000 + manvar_costs
# Import Plant Fixed Costs
fixed_costs = pd.read_excel('fixed_cost.xlsx', index_col = 0)
# Import Low Capacity and High Capacity Plant
cap = pd.read_excel('capacity.xlsx', index_col = 0)
# Import Demand
```

```
demand = pd.read_excel('demand.xlsx', index_col = 0)

# Define Decision Variables
loc = ['USA', 'Germany', 'Japan', 'Brazil', 'India']
size = ['Low', 'High']

# Initialize Class
model = LpProblem("Supply Chain Optimization", LpMinimize)

# Create Decision Variables
x = LpVariable.dicts("production_", [(i,j) for i in loc for j in loc],
                     lowBound=0, upBound=None, cat='continuous')
y = LpVariable.dicts("plant_",
                     [(i,s) for s in size for i in loc], cat='Binary')

import pandas as pd
from pulp import *

# Import Manufacturing Costs
manvar_costs = pd.read_excel('variable_costs.xlsx', index_col = 0)
# Import Freight Costs
freight_costs = pd.read_excel('freight_costs.xlsx', index_col = 0)
# Variable Cost
var_cost = freight_costs/1000 + manvar_costs
# Import Plant Fixed Costs
fixed_costs = pd.read_excel('fixed_cost.xlsx', index_col = 0)
# Import Low Capacity and High Capacity Plant
cap = pd.read_excel('capacity.xlsx', index_col = 0)
# Import Demand
demand = pd.read_excel('demand.xlsx', index_col = 0)

# Define Decision Variables
loc = ['USA', 'Germany', 'Japan', 'Brazil', 'India']
size = ['Low', 'High']

# Initialize Class
model = LpProblem("Supply Chain Optimization", LpMinimize)

# Create Decision Variables
x = LpVariable.dicts("production_", [(i,j) for i in loc for j in loc],
                     lowBound=0, upBound=None, cat='continuous')
y = LpVariable.dicts("plant_",
                     [(i,s) for s in size for i in loc], cat='Binary')

# Solve Model
model.solve()
print("Total Costs = {:,} ($/Month)".format(int(value(model.objective))))
print('\n' + "Status: {}".format(LpStatus[model.status]))
```

```
# Dictionnary
dict_plant = {}
dict_prod = {}
for v in model.variables():
    if 'plant' in v.name:
        name = v.name.replace('plant_', '').replace('', '')
        dict_plant[name] = int(v.varValue)
        p_name = name
    else:
        name = v.name.replace('production_', '').replace('', '')
        dict_prod[name] = v.varValue
    print(name, "=", v.varValue)
```

/**********End**********/