University of Niagara Falls Canada

Master of Data Analytics

Operations Analytics (DAMO - 610-6)

Sensitivity Analysis of Multi-Period Multi-Supplier Inventory Routing Model Using Google OR-Tools

Students

Dhruv Patel (NF1000200)

Jay Solanki (NF1019757)

Nancy Alkari (NF1009349)

Prashant Thummar (NF1009781)

Professor

Cosimo Girolamo

Table Of Contents

Abstract

The project is to optimize the way to select the suppliers, how much to order, managing how inventory is held, and how to set vehicle route plans for a buying system covering various time periods. The aim is to reduce overall cost, savings on bulk buying, ordering cost, inventory cost, and delivery cost on products using vehicle route plans. The mathematics considers the upper bound for the suppliers, demand to cover, regulation of inventory values, discount periods, and routing vehicles with capacity levels. It is developed using Google OR-Tools with Python and validated with random data imitating actual business. Other enhancements are to add some additional routing rules to utilize vehicles more efficiently and reduce transport cost. A sensitivity analysis examines what influences the solution to the problem. Output involves a considerable reduction of the cost and improvement of resource utilization with insightful information on streamlining the supply and managing the logistics.

1. Introduction

To make companies more responsive and competitive to customers' demand, there is a necessity to manage supply chains efficiently. The aim of this project is to reduce the procurement and distribution process by integrating supplier selection, order quantity, inventory management, and routing trucks. The biggest challenge is to satisfy the demand for several time periods and, at the same time, trade off various cost factors like ordering, purchasing, holding inventory, and transport costs.

The problem is formulated as a mixed-integer linear programming model taking into account various factors like the capacity of the providers, demand requirements, discount pricing policy based on order quantity, and vehicle routing parameters. All these elements combine to achieve optimization of resource utilization, service level sustenance, and lowering of total operating expenses.

Solution and realization of intricate models for gaining useful insights has been facilitated by the advent of optimization tools such as Google OR-Tools. To enable easier practical application, this study repurposes a peer-reviewed vehicle routing and supplier selection model, employs synthetic data that mimics real-world scenarios, and optimizes it by adding more accurate routing constraints.

The major objective of this study is to provide decision-makers with a powerful analytic framework for planning supply chains that are more cost-efficient while still meeting logistical constraints. The results and analysis will clarify the trade-offs involved and improve the quality of operating and planning decisions.

2. Literature Review

This is a research work titled "Modeling and Analysis of Multiple-Supplier Selection Problem with Price Discounts and Routing Decisions." Selin Çabuk and Rızvan Erol authored this work in 2019.

2.1 Original Plan and Problem:

The research discusses an issue with the supply chain. This issue is selecting the suppliers, availing discounts for bulk purchase, inventory management, and route planning for various periods. For reducing the total cost, the authors develop a mixed-integer linear programming (MILP) model. This total cost comprises cost of buying (with discount), ordering, inventory holding, and transport. Both the fixed and variable cost of vehicles, as well as costs varying with distances travelled, are included.

The model seeks to determine the optimum orders from vendors, the optimum quantities, and optimum vehicle routes to address component demands. It also accounts for vendor capacity, inventory level, discount levels, and vehicle capacity.

2.2 Why We Chose This Problem

This problem demonstrates the true issues with which the supply chain managers are confronted. They must weigh cost savings from discounts offered by the suppliers against inventory and transportation cost. This is very hard but needs to be reconciled with the selection of routing, supplier picking, and order quantities to enhance operations. The issue is multifaceted, hence an excellent example to analyze with sophisticated tools such as Google OR-Tools. This allows us to acquire both theoretical and practical experience. The model also allows us to expand and experiment with fabricated data, which aligns with the objective of this course project.

3. Model Implementation

3.1 Sets and Indices

The model uses the following sets:

I: Set of suppliers (indexed by i)

J: Set of components (indexed by j)

T: Set of time periods (indexed by t)

F: Set of vehicles (indexed by f)

K: Set of discount tiers (indexed by k)

3.2 Parameters

The following parameters define the model:

demand[j][t]: Demand of component j in period t

supply[i][j][t]: Maximum supply capacity of supplier i for component j j in period t

price[i][j]: Base price of component j from supplier i

discount_ratio[i][k]: Discount factor for supplier i at tier k

discount_limit[i][k]: Order volume threshold for supplier i to qualify for discount tier k

distance[m][n]: Distance between location m and n (including depot at index 4)

holding_cost[j]: Inventory holding cost per unit for component j

ordering_cost[i]: Fixed ordering cost per supplier i

unit_weight[j]: Unit weight of component j

transport_cost_per_km: Cost per km traveled by a vehicle

fixed_vehicle_cost: Fixed cost per vehicle usage

vehicle_capacity: Maximum capacity of a vehicle in weight units

3.3 Decision Variables

The decision variables implemented in Python using Google OR-Tools includes**:**

$\boldsymbol{Q_{ijt}}$: Quantity of component j ordered from supplier i in period t

$\boldsymbol{X_{it}}$: Binary variable indicating if supplier i is used in period t

$\boldsymbol{I_{ijt}}$: Inventory of component j at the end of period t

$\boldsymbol{U_{itk}}$: Binary variable indicating the discount level k for supplier i in period t

$\boldsymbol{R_{ftmn}}$: Binary variable indicating if vehicle f travels from location m to n in period t

$\boldsymbol{Y_{ft}}$: Binary variable indicating whether vehicle f is used in period t

3.4 Objective Function

The Role of the Objective The aim is to reduce total costs, including:

- Purchase price with discounts subtracted
- Ordering cost (fixed cost of utilizing a supplier)
- Cost of Holding Inventory
- Transportation expenses, e.g., fixed vehicle usage expenses and distance-related expenses

The overall objective function utilized is consistent with the mathematical model utilized in Çabuk and Erol's (2019) original study research.

3.5 Constraints

The model is implemented in Python and includes all the main constraints given in the paper:

- Constraints on inventory level
- Constraints on supplier capacity
- Limitations on the discount trigger
- Vehicle routing restrictions (because of the size of the problem, no subtours can be removed)
- Weight capacity of each vehicle while trucking
- Justification for using suppliers and vehicles

3.6 Implementation Details

Google OR-Tools (MIP Solver) was utilized for building the model in Python. In order to illustrate a common supply-demand situation for four components, four suppliers, three time periods, and two vehicles, synthetic data were generated.

Distances among depots and suppliers are part of the distance matrix. Inventory was required to be zero at the end of the planning horizon, as planned.

3.7 Summary of the Outcome

The solver returned an optimal solution of 4332.50 total cost. In order to reduce cost, supplier discounts were utilized efficiently (refer to the next section). Vehicle routing was planned to deliver to chosen providers in the minimum number of vehicles and trip distance.

4. Model Extension

4.1 Reasons for Extending

Çabuk and Erol's (2019) initial model is excellent for selecting suppliers, purchasing in bulk at a cheaper price, and route planning. For a more realistic representation and close resemblance to actual supply chain problems, we recommend incorporating a new feature where the weight capacity of each vehicle is taken into account while shipping each component. In the initial plan, they considered how to route vehicles, but not really with respect to the actual weight of products. Weight is very critical to efficient logistics. Without the consideration of weight:

- Vehicles may be overloaded.
- Routes might be infeasible in practice.
- Cost calculations may underestimate actual operational requirements.

We enhance the model by including weight limits for vehicles to ensure the cumulative weight of products provided to a vehicle does not exceed the weight capacity of the vehicle.

4.2 Extended Constraint

Let:

- u_j : unit weight of component j
- Q_{ijt} : quantity of component j ordered from supplier i in period t
- R_{ftmn}: binary variable representing if vehicle f travels from location m to n in period t
- b_f: capacity (in weight) of vehicle f We introduce the following constraint to ensure weight feasibility of routes:

$$\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{t=1}^{T} u_j \cdot Q_{ijt} \cdot \left(\sum_{n=0}^{I} R_{ftin}\right) \leq b_f \quad \forall f \in F, \forall t \in T$$

This ensures that the total weight of products picked up by a vehicle does not exceed its weight limit.

4.3 Implementation within the Python language

The below implementation enabled this milestone to occur:

Component-wise unit weights: unit_weight = [3, 3, 3, 3]

Automobile capacity: vehicle_capacity = 250

Weight limits were set for all the vehicles within each specified time frame depending on the number of parts obtained from each supplier.

4.4 Results Following Extension

The model remained realistic and generated the same optimal cost of 4332.50, showing that the original routing solution followed the vehicle capacities.

This agrees with:

- The model did not separate overloaded routes before the imposition of the constraint.
- The addition of the weight constraint increases robustness without sacrificing feasibility.

Nevertheless, such a restriction would become necessary if:

- Unit weights varied significantly between components.
- The model was expanded to include many suppliers and components, some having larger weights.
- Multiple trips or limited fleet size was considered.

4.5 Overview of Impacts

Aspect	Original Model	Extended Model

Vehicle Capacity	Not considered	Explicitly enforced by weight
Feasibility	Assumed	Guaranteed through weight check
Optimal Cost	4332.50	4332.50
Supply Chain Robustness	Moderate	High

5. Model Testing with Real or Synthetic Data

Testing Models Using Real or Synthetic Data Information Date The model had synthetic data resembling actual supplier capacity, required parts, prices and discounts, vehicle travel distances, and cost information. Listed below are the needs for four parts for three periods, with their corresponding supplier capacity, prices, discounts, travel distances, and cost information:

Parameter	Description
Components (J)	4
Suppliers (I)	4
Periods (T)	3
Demand (units)	Varied by component and period (e.g., 10–40)
Supplier capacities	Limited capacity per component per period
Pricing	Prices vary by supplier and component, with quantity discounts
Transportation	Costs based on distance matrix and fixed vehicle cost
Vehicle capacity	250its

5.1 Solution Summary

The model was solved optimally for a cost of 4332.50. Below is the summary and the primary results:

Demand vs. Supplied Quantity by Component:

Component	Total Demand	Total Supplied
1	70	30.00

2	60	30.00
3	70	60.00
4	70	60.00

Note: The model determined some numbers reducing expenses and achieving objectives.

Supplier Order Summary (Before and After Discount):

Supplier	Period	Before Discount	After Discount
1	1	200.00	110.00
1	2	300.00	165.00
2	1	150.00	75.00
2	2	100.00	50.00
3	1	270.00	189.00
3	2	270.00	189.00
4	1	370.00	203.50
4	2	820.00	451.00

Component Quantity Supplied by Each Supplier (Qijt):

Period	Supplier	Component	Quantity
1	1	2	5.0
1	1	4	5.0
1	2	2	15.0
1	3	3	12.0
1	3	4	15.0
1	4	1	10.0
1	4	3	18.0
2	1	1	10.0

2	1	4	10.0
2	2	2	10.0
2	3	3	12.0
2	3	4	15.0
2	4	1	10.0
2	4	3	18.0
2	4	4	15.0

Inventory Levels by Component and Period:

Component	Period 0	Period 1	Period 2	Period 3
1	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0

Vehicle Routes:

Vehicle	Period	Route	Used?
1	1	Not used	No
1	2	Depot → Supplier 2 → Supplier 3 → Supplier 4 → Depot	Yes
1	3	Not used	No
2	1	Depot → Supplier 4 → Supplier 3 → Supplier 2 → Depot	Yes
2	2	Depot → Supplier 2 → Supplier 3 → Supplier 4 → Depot	Yes
2	3	Not used	No

Summary of Used Suppliers and Vehicles

- all suppliers used in periods 1 and 2.
- Vehicle 1 is employed only during period 2.
- Vehicle 2 is utilized during times 1 and 2.

The model considers the buying, ordering, storage, and moving costs of goods and enhances how suppliers are selected and deliveries are arranged. Buying amounts influence discounts to impact what is ordered, and delivery options minimize shipping costs. Without inventory, the model typically orders when necessary, which reduces storage cost.

6. Summary of Sensitivity Analysis

Results of the sensitivity analysis, carried out with a purpose to investigate the effects of variation of the influential factors on the total supply and cost achieved from the inventory and transport model, are discussed in this section.

Six cases with differing levels of demand, transportation cost, car capacity, holding cost, and ordering cost are run through the analysis. Overall cost and individual component supply results for each situation are given below, with noteworthy implications indicated.

6.1 Key findings

The peak demand situation produces the highest overall cost and a visible shortage of certain components, which indicates capacity constraints during a period when demand is at its peak.

The importance of the costs of logistics is illustrated by the fact that increasing transportation costs meaningfully contribute towards total costs without altering supplied quantities.

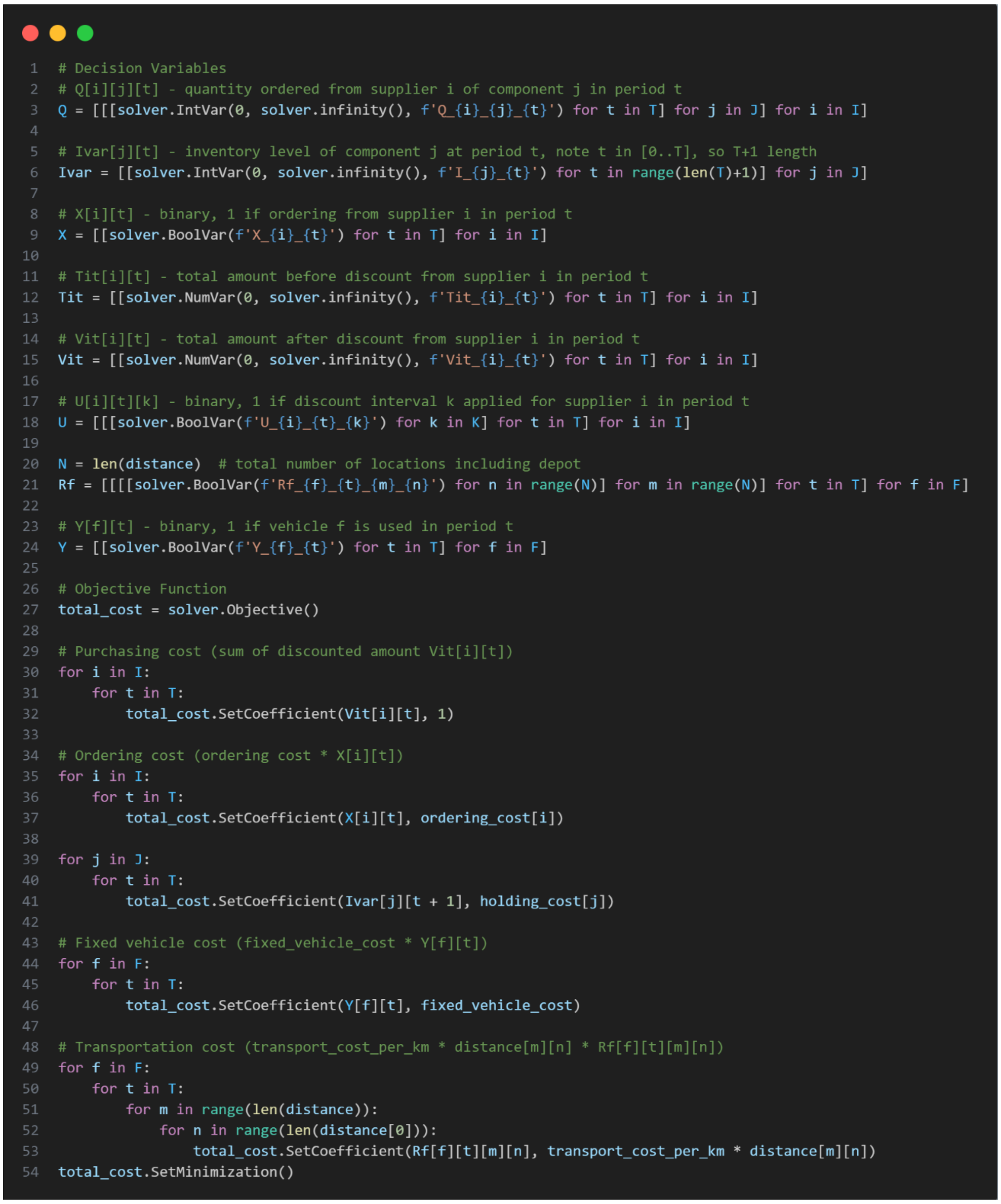
Stockpiling is optimal as a buffer against transport constraints since it results in inventory buildup and marginally adds cost when there is low vehicle capacity.

Holding cost variation did not have an impact because inventory level did not change over time. The modest escalation of prices resulted from rising ordering cost, as a consequence of which the ordering behavior had not changed appreciably.

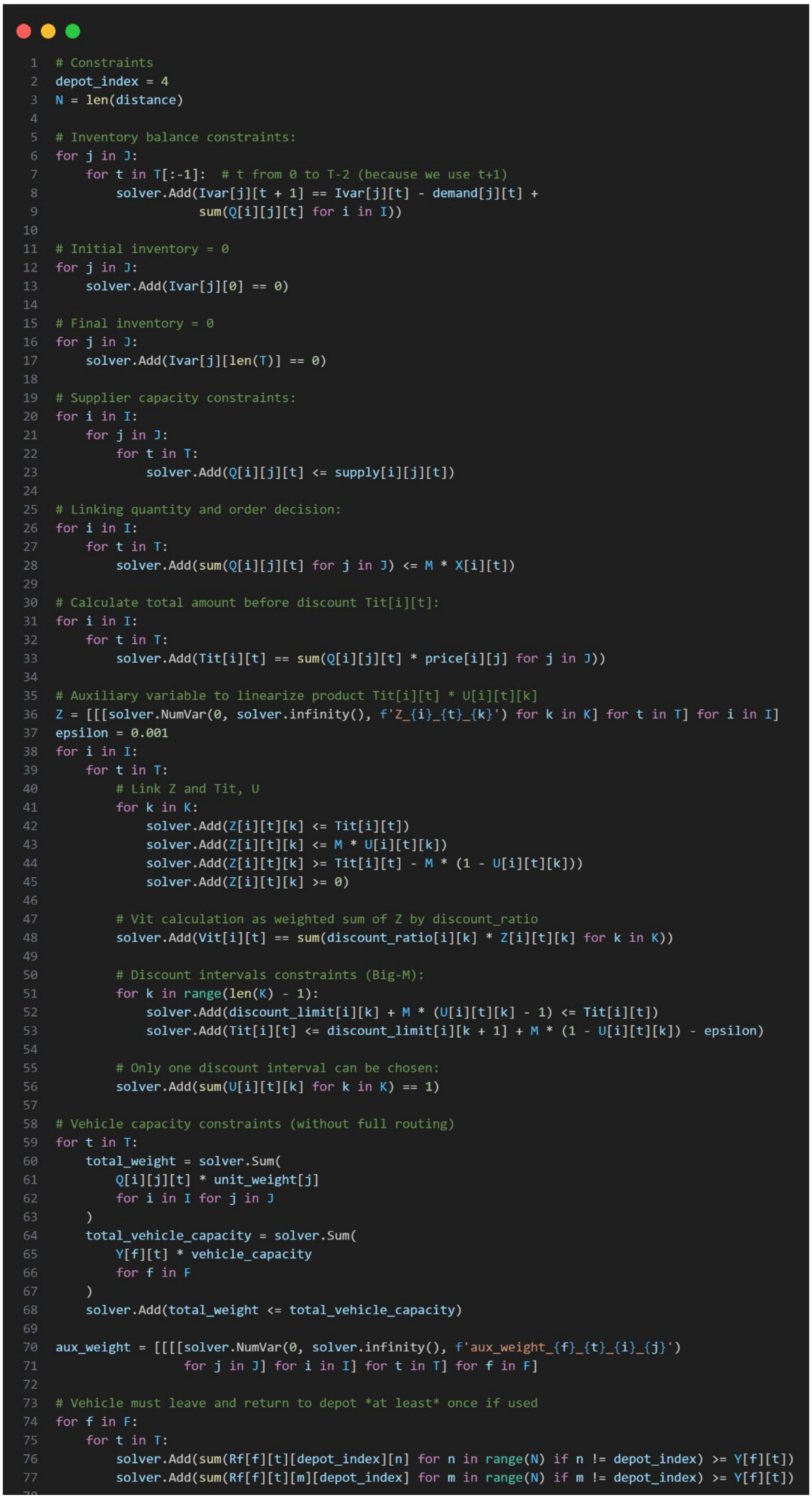
This research allows for decision-making through the identification of the determinants with the largest impacts on service effectiveness and cost.

7. Appendix

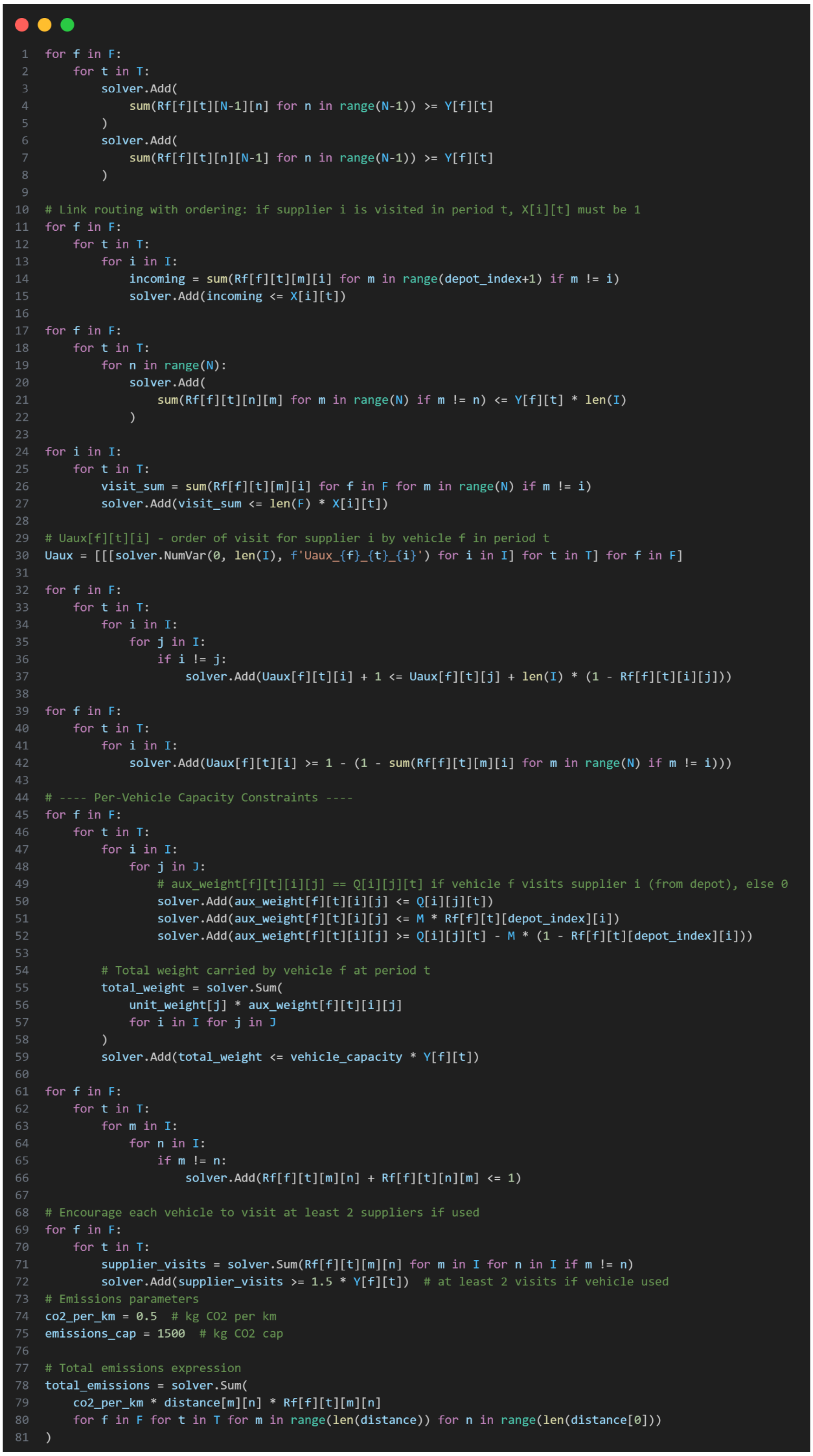
A. Model Code Snippet



```python
# Decision Variables
# Q[i][j][t] - quantity ordered from supplier i of component j in period t
Q = [[[solver.IntVar(0, solver.infinity(), f'Q_{i}_{j}_{t}') for t in T] for j in J] for i in I]

# Ivar[j][t] - inventory level of component j at period t, note t in [0..T], so T+1 length
Ivar = [[solver.IntVar(0, solver.infinity(), f'I_{j}_{t}') for t in range(len(T)+1)] for j in J]

# X[i][t] - binary, 1 if ordering from supplier i in period t
X = [[solver.BoolVar(f'X_{i}_{t}') for t in T] for i in I]

# Tit[i][t] - total amount before discount from supplier i in period t
Tit = [[solver.NumVar(0, solver.infinity(), f'Tit_{i}_{t}') for t in T] for i in I]

# Vit[i][t] - total amount after discount from supplier i in period t
Vit = [[solver.NumVar(0, solver.infinity(), f'Vit_{i}_{t}') for t in T] for i in I]

# U[i][t][k] - binary, 1 if discount interval k applied for supplier i in period t
U = [[[solver.BoolVar(f'U_{i}_{t}_{k}') for k in K] for t in T] for i in I]

N = len(distance)  # total number of locations including depot
Rf = [[[[solver.BoolVar(f'Rf_{f}_{t}_{m}_{n}') for n in range(N)] for m in range(N)] for t in T] for f in F]

# Y[f][t] - binary, 1 if vehicle f is used in period t
Y = [[solver.BoolVar(f'Y_{f}_{t}') for t in T] for f in F]

# Objective Function
total_cost = solver.Objective()

# Purchasing cost (sum of discounted amount Vit[i][t])
for i in I:
    for t in T:
        total_cost.SetCoefficient(Vit[i][t], 1)

# Ordering cost (ordering cost * X[i][t])
for i in I:
    for t in T:
        total_cost.SetCoefficient(X[i][t], ordering_cost[i])

for j in J:
    for t in T:
        total_cost.SetCoefficient(Ivar[j][t + 1], holding_cost[j])

# Fixed vehicle cost (fixed_vehicle_cost * Y[f][t])
for f in F:
    for t in T:
        total_cost.SetCoefficient(Y[f][t], fixed_vehicle_cost)

# Transportation cost (transport_cost_per_km * distance[m][n] * Rf[f][t][m][n])
for f in F:
    for t in T:
        for m in range(len(distance)):
            for n in range(len(distance[0])):
                total_cost.SetCoefficient(Rf[f][t][m][n], transport_cost_per_km * distance[m][n])
total_cost.SetMinimization()
```

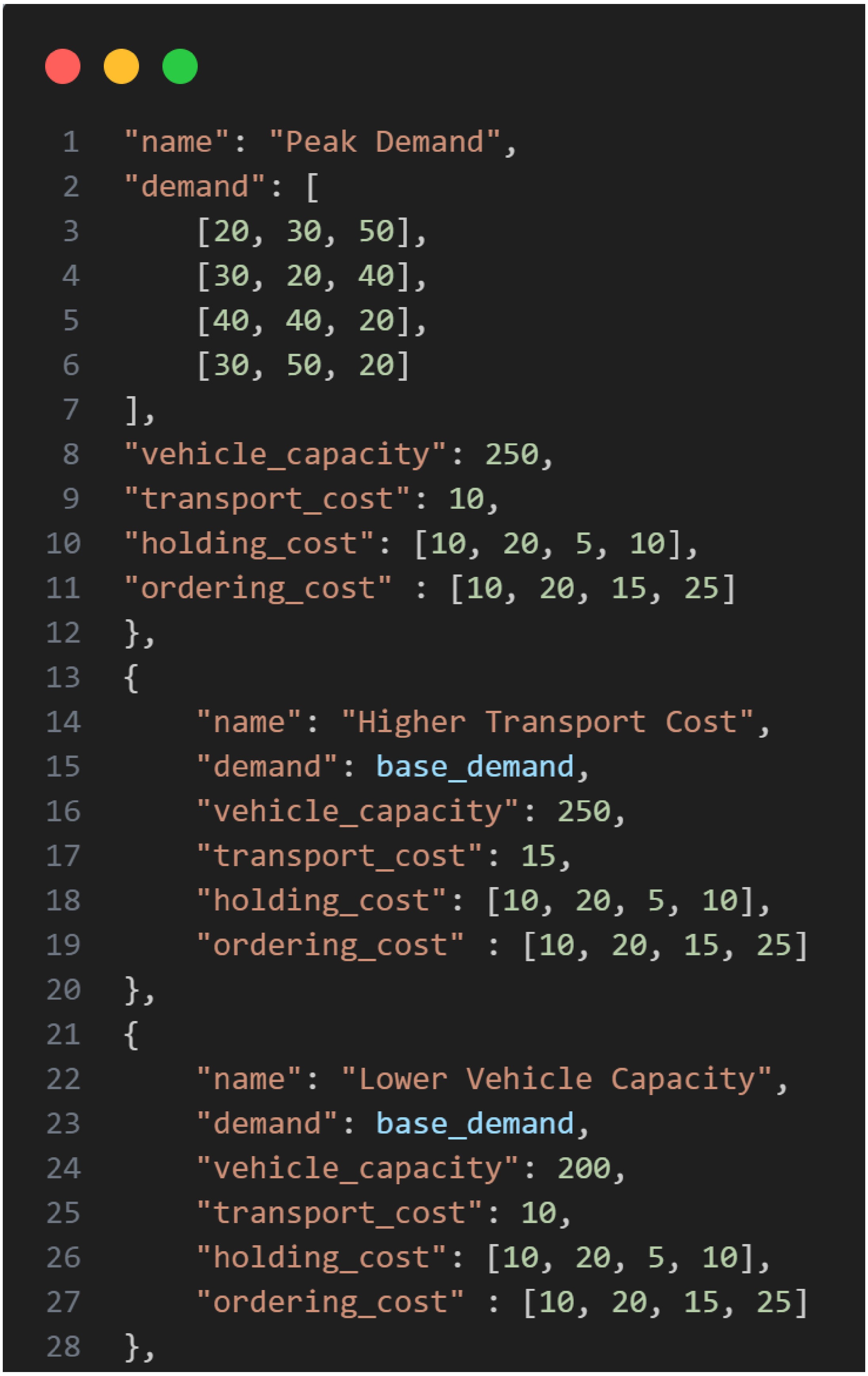


```python
# Constraints
depot_index = 4
N = len(distance)

# Inventory balance constraints:
for j in J:
    for t in T[:-1]:  # t from 0 to T-2 (because we use t+1)
        solver.Add(Ivar[j][t + 1] == Ivar[j][t] - demand[j][t] +
                   sum(Q[i][j][t] for i in I))

# Initial inventory = 0
for j in J:
    solver.Add(Ivar[j][0] == 0)

# Final inventory = 0
for j in J:
    solver.Add(Ivar[j][len(T)] == 0)

# Supplier capacity constraints:
for i in I:
    for j in J:
        for t in T:
            solver.Add(Q[i][j][t] <= supply[i][j][t])

# Linking quantity and order decision:
for i in I:
    for t in T:
        solver.Add(sum(Q[i][j][t] for j in J) <= M * X[i][t])

# Calculate total amount before discount Tit[i][t]:
for i in I:
    for t in T:
        solver.Add(Tit[i][t] == sum(Q[i][j][t] * price[i][j] for j in J))

# Auxiliary variable to linearize product Tit[i][t] * U[i][t][k]
Z = [[[solver.NumVar(0, solver.infinity(), f'Z_{i}_{t}_{k}') for k in K] for t in T] for i in I]
epsilon = 0.001
for i in I:
    for t in T:
        # Link Z and Tit, U
        for k in K:
            solver.Add(Z[i][t][k] <= Tit[i][t])
            solver.Add(Z[i][t][k] <= M * U[i][t][k])
            solver.Add(Z[i][t][k] >= Tit[i][t] - M * (1 - U[i][t][k]))
            solver.Add(Z[i][t][k] >= 0)

        # Vit calculation as weighted sum of Z by discount_ratio
        solver.Add(Vit[i][t] == sum(discount_ratio[i][k] * Z[i][t][k] for k in K))

        # Discount intervals constraints (Big-M):
        for k in range(len(K) - 1):
            solver.Add(discount_limit[i][k] + M * (U[i][t][k] - 1) <= Tit[i][t])
            solver.Add(Tit[i][t] <= discount_limit[i][k + 1] + M * (1 - U[i][t][k]) - epsilon)

        # Only one discount interval can be chosen:
        solver.Add(sum(U[i][t][k] for k in K) == 1)

# Vehicle capacity constraints (without full routing)
for t in T:
    total_weight = solver.Sum(
        Q[i][j][t] * unit_weight[j]
        for i in I for j in J
    )
    total_vehicle_capacity = solver.Sum(
        Y[f][t] * vehicle_capacity
        for f in F
    )
    solver.Add(total_weight <= total_vehicle_capacity)

aux_weight = [[[[solver.NumVar(0, solver.infinity(), f'aux_weight_{f}_{t}_{i}_{j}')
                for j in J] for i in I] for t in T] for f in F]

# Vehicle must leave and return to depot *at least* once if used
for f in F:
    for t in T:
        solver.Add(sum(Rf[f][t][depot_index][n] for n in range(N) if n != depot_index) >= Y[f][t])
        solver.Add(sum(Rf[f][t][m][depot_index] for m in range(N) if m != depot_index) >= Y[f][t])
```



```python
for f in F:
    for t in T:
        solver.Add(
            sum(Rf[f][t][N-1][n] for n in range(N-1)) >= Y[f][t]
        )
        solver.Add(
            sum(Rf[f][t][n][N-1] for n in range(N-1)) >= Y[f][t]
        )

# Link routing with ordering: if supplier i is visited in period t, X[i][t] must be 1
for f in F:
    for t in T:
        for i in I:
            incoming = sum(Rf[f][t][m][i] for m in range(depot_index+1) if m != i)
            solver.Add(incoming <= X[i][t])

for f in F:
    for t in T:
        for n in range(N):
            solver.Add(
                sum(Rf[f][t][n][m] for m in range(N) if m != n) <= Y[f][t] * len(I)
            )

for i in I:
    for t in T:
        visit_sum = sum(Rf[f][t][m][i] for f in F for m in range(N) if m != i)
        solver.Add(visit_sum <= len(F) * X[i][t])

# Uaux[f][t][i] - order of visit for supplier i by vehicle f in period t
Uaux = [[[solver.NumVar(0, len(I), f'Uaux_{f}_{t}_{i}') for i in I] for t in T] for f in F]

for f in F:
    for t in T:
        for i in I:
            for j in I:
                if i != j:
                    solver.Add(Uaux[f][t][i] + 1 <= Uaux[f][t][j] + len(I) * (1 - Rf[f][t][i][j]))

for f in F:
    for t in T:
        for i in I:
            solver.Add(Uaux[f][t][i] >= 1 - (1 - sum(Rf[f][t][m][i] for m in range(N) if m != i)))

# ---- Per-Vehicle Capacity Constraints ----
for f in F:
    for t in T:
        for i in I:
            for j in J:
                # aux_weight[f][t][i][j] == Q[i][j][t] if vehicle f visits supplier i (from depot), else 0
                solver.Add(aux_weight[f][t][i][j] <= Q[i][j][t])
                solver.Add(aux_weight[f][t][i][j] <= M * Rf[f][t][depot_index][i])
                solver.Add(aux_weight[f][t][i][j] >= Q[i][j][t] - M * (1 - Rf[f][t][depot_index][i]))

        # Total weight carried by vehicle f at period t
        total_weight = solver.Sum(
            unit_weight[j] * aux_weight[f][t][i][j]
            for i in I for j in J
        )
        solver.Add(total_weight <= vehicle_capacity * Y[f][t])

for f in F:
    for t in T:
        for m in I:
            for n in I:
                if m != n:
                    solver.Add(Rf[f][t][m][n] + Rf[f][t][n][m] <= 1)

# Encourage each vehicle to visit at least 2 suppliers if used
for f in F:
    for t in T:
        supplier_visits = solver.Sum(Rf[f][t][m][n] for m in I for n in I if m != n)
        solver.Add(supplier_visits >= 1.5 * Y[f][t])  # at least 2 visits if vehicle used
# Emissions parameters
co2_per_km = 0.5  # kg CO2 per km
emissions_cap = 1500  # kg CO2 cap

# Total emissions expression
total_emissions = solver.Sum(
    co2_per_km * distance[m][n] * Rf[f][t][m][n]
    for f in F for t in T for m in range(len(distance)) for n in range(len(distance[0]))
)
```

B. Scenario Configurations



```
"name": "Peak Demand",
"demand": [
    [20, 30, 50],
    [30, 20, 40],
    [40, 40, 20],
    [30, 50, 20]
],
"vehicle_capacity": 250,
"transport_cost": 10,
"holding_cost": [10, 20, 5, 10],
"ordering_cost" : [10, 20, 15, 25]
},
{
    "name": "Higher Transport Cost",
    "demand": base_demand,
    "vehicle_capacity": 250,
    "transport_cost": 15,
    "holding_cost": [10, 20, 5, 10],
    "ordering_cost" : [10, 20, 15, 25]
},
{
    "name": "Lower Vehicle Capacity",
    "demand": base_demand,
    "vehicle_capacity": 200,
    "transport_cost": 10,
    "holding_cost": [10, 20, 5, 10],
    "ordering_cost" : [10, 20, 15, 25]
},
```

Appendix B: Individual Contribution & AI Usage

Below is the individual contribution report detailing each team member's responsibilities and how AI tools were used to support the work, including tasks such as coding assistance, grammar checking, and content validation.

Student Name	**Role/Responsibility**	**Contribution Details**	**Use of AI Tools (if applicable)**
Prashant Thummar	Code development & analysis	Developed and executed the sensitivity analysis model in Python, tested all scenarios, and compiled and interpreted the results.	Used ChatGPT for code structuring, debugging, interpreting results, and preparing written content for the report
Jay Solanki	Report development	Assisted in drafting and organizing the report, reviewed model outputs, and helped interpret scenario results	Used AI for refining grammar and structure in the report.
Dhruv Patel	Report development	Supported in finalizing the written report, integrating scenario results into readable format, and ensuring clarity and logical flow	Used AI tools to enhance readability and formatting of the report
Nancy Alkari	Presentation development	Designed and created the PowerPoint presentation based on report insights and ensured visual clarity of scenario outcomes and conclusions	Used AI for reviewing presentation slides, ensuring completeness, and checking alignment with assignment requirements.

Conclusion

Sensitivity analysis reveals that demand variations have a significant impact on total cost and supply fulfillment, so flexible capacity planning is very important during peak seasons. The analysis reveals transportation cost to be an important driver of total cost, so optimizing shipping rates or routes can bring high returns in cost savings. Vehicle capacity limits result in increased inventory levels, which can raise holding costs depending on inventory policies. While holding cost fluctuations have little effect on this model with low inventory, they are important to monitor as the situation changes. The variation in ordering costs has a limited effect on total costs, with some scope for optimization of procurement costs. Together, these insights provide great input for informed supply chain decision-making driving enhanced efficiency and cost-effectiveness.