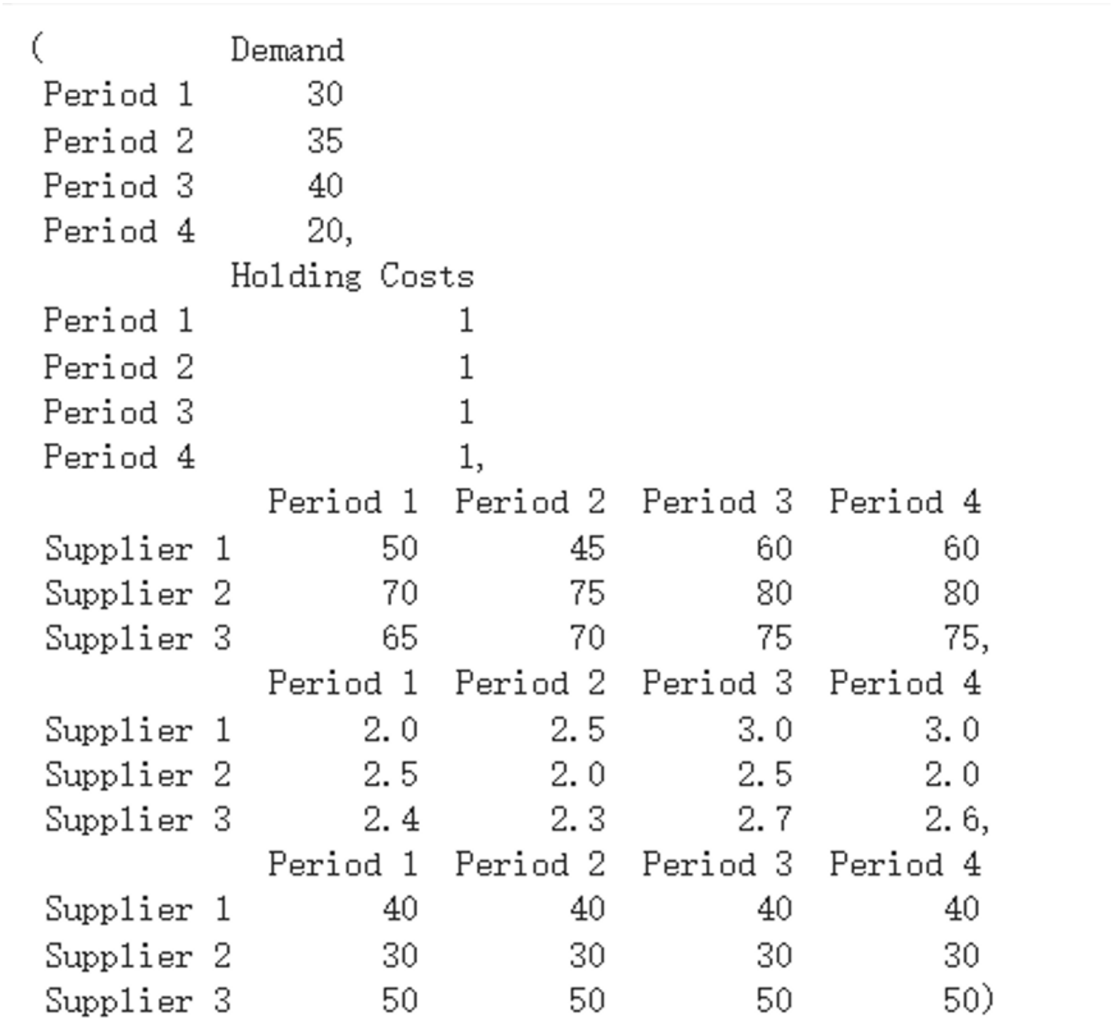
1. Introduction

According to Miller (2010), inventory management includes all activities to ensure that customers have the products or services they need.It includes the purchasing, manufacturing and distribution functions to meet the marketing needs and organizational requirements for delivering products to customers. In the highly competitive environment of electronic products, enterprises need to operate in an efficient manner in order to maintain strong competitiveness. Inventory management and purchasing management have become important areas of decision-making, as mismanagement of these two areas has a significant impact on the overall performance and operations of an organization. In order to accurately determine the size of the inventory, it is necessary to establish an appropriate inventory policy based on the specificity of the technical process and the turnover achieved（Yugang，2009;Ballou,2005）. In this study, with three suppliers and different pricing and ordering costs for each supplier, it is also necessary to consider the firm's inventory holding costs, and making the decision to minimize inventory costs is difficult. Therefore, firms need to use appropriate mathematical planning methods to make optimal decisions: the supplier to be selected for each period and the amount of products to be ordered in order to minimize the total inventory cost.

2. Problem Identification

The focus of this study is a retail and curtain manufacturing company called English Blinds. John Moss, CEO of Enterprise, mentioned how Enterprise used to use algorithm-based forecasting tools and manual oversight to gain the insights needed to manage inventory and ensure that the business was able to meet demand without holding a surplus or commodities that were stuck in seasonality or hard to move. Firms have been in the same business since the 1980s and therefore have reliable and consistent core data that can help them forecast seasonal demand and identify trends and forms of purchase. This study can take advantage of the ability of firms to predict demand more accurately and develop an optimization algorithm model that will help firms be able to minimize the total cost of inventory. Assuming that the company has three suppliers offering identical products. The three suppliers have provided pricing and ordering costs for each quarter of the following year, and the firm's demand in each period is fixed and can be known in advance. Also, inventory holding costs need to be considered in the problem study. The firm wants to minimize the annual inventory cost and compress the total cost expenditure of the firm by meeting the known demand for the product for four quarters, thus increasing the funds for the development of the next new product. Therefore, a decision needs to be made as to which supplier to buy from and in what quantities for each quarter of the following year. If the lot size for a given period exceeds the firm's demand for that period, the excess product will be retained in inventory.

As shown in Figure 1, the pricing, ordering costs, and warehouse holding costs of the company's three suppliers have been the same each quarter for the past two years and for the new year. However, the company did not consider using optimization models and algorithms to minimize inventory costs in the first two years, and the ordering strategy was different for each year, although they all met the demand for each quarter. However, as can be seen from Table 1, the total inventory cost varied greatly over the two years. Now that a new year has begun, in order to minimize the inventory cost, the company intends to use the optimization model and algorithm to achieve the goal.



```
(           Demand
 Period 1      30
 Period 2      35
 Period 3      40
 Period 4      20,
            Holding Costs
 Period 1              1
 Period 2              1
 Period 3              1
 Period 4              1,
              Period 1  Period 2  Period 3  Period 4
 Supplier 1        50        45        60        60
 Supplier 2        70        75        80        80
 Supplier 3        65        70        75        75,
              Period 1  Period 2  Period 3  Period 4
 Supplier 1       2.0       2.5       3.0       3.0
 Supplier 2       2.5       2.0       2.5       2.0
 Supplier 3       2.4       2.3       2.7       2.6,
              Period 1  Period 2  Period 3  Period 4
 Supplier 1        40        40        40        40
 Supplier 2        30        30        30        30
 Supplier 3        50        50        50        50)
```

Figure 1. Assumed initial data

	The First Year	The Second Year
Period 1	Supplier 2 provided 30.0 units.	Supplier 2 provided 30.0 units.
Period 2	Supplier 1 provided 35.0 units.	Supplier 3 provided 35.0 units.
Period 3	Supplier 1 provided 40.0 units.	Supplier 3 provided 40.0 units.
Period 4	Supplier 1 provided 20.0 units.	Supplier 3 provided 20.0 units.

Total inventory cost	577.5	605.5

Table 1. Procurement Strategy and Total Inventory Cost Trends Over the Past Two Years

On the contrary, if proper inventory planning is not done in advance, on the one hand, excessive inventory will increase the cost of inventory, and it will lead to the enterprise's cash being hoarded in unsold products, which could have been used for other investment or operating activities, or even cause cash flow constraints (Felea, 2008;Coyle, Bardi, and Langley, 2003). And unnecessary stockpiling of inventory affects a company's cost of goods sold; high inventory levels may require promotion through discounting, thus compressing profit margins. On the other hand, if inventory does not meet demand, it can affect customer satisfaction, which in turn affects customer brand loyalty and may require increased investment in marketing. Poor inventory management can affect other aspects such as production planning and logistics, e.g. lack of lack of inventory in a particular quarter requires temporary increase in capital investment and manpower for logistics, thus affecting the overall business operational efficiency.

3. Optimisation Model

This paper assumes:

- Products are transported directly from the supplier to the consumer without any intermediaries involved.
- The holding cost per quarter of product is constant, independent of storage time or quantity.
- The demand for the product in each time period is known and constant.
- Only one product is considered.
- Only one product is considered; Only one supplier can purchase the product in each time period.

- The number of products available from the three suppliers varies from quarter to quarter, and varies from quarter to quarter.
- The firm is able to predetermine the demand for its products.
- Order quantity discounts are not taken into account.
- Inventory holding cost is linearly related to the number of products stored in inventory; order delivery lead times are deterministic and identical, which we assume to be zero.
- This paper assumes zero initial inventory at the end of the first quarter and zero inventory at the end of the last quarter.
- The firm's needs for the product should be satisfied at each quarter in a year.
- Cross-period purchasing is allowed, e.g., purchases can be made one quarter in advance to satisfy demand in the next two quarters

Sets, parameters and decision variables are defined as follows:

U Numerical identifiers for suppliers, with u taking values 1 through 3 to distinguish each supplier.

T Numerical labels for the distinct time periods, with t ranging from 1 to 4.

d_t The required quantity of the product by the firm for each period t.

h_t Cost per unit to hold inventory for each period t.

s_{ut} The static cost charged by supplier uu for placing an order in period t.

p_{ut} Price per unit charged by supplier uu for the product in period t.

I_t The total inventory held at the end of each period t.

X_{ut} The volume of product ordered from supplier uu during period t.

Y_{ut} This is set to 1 if an order is placed with supplier uu during period tt, and 0 otherwise.

The mathematical model of the problem can be formulated as follows,

$$Minimize \sum_{t=1}^{T}\sum_{u=1}^{U} s_{ut}Y_{ut} + \sum_{t=1}^{T}\sum_{u=1}^{U} p_{ut}X_{ut} + \sum_{t=1}^{T} h_t I_t$$

$$I_t = I_{t-1} + \sum_{u=1}^{U} X_{ut} - d_t \quad \forall t$$

$$X_{ut} \leq Y_{ut} \sum_{1}^{t} d_k \qquad \forall u, t$$

$$Y_{ut} = 0\ or\ 1 \qquad \forall u, t$$

$$X_{ut}, I_t \geq 0 \qquad \forall u, t$$

The choice of the model was based on several considerations that highlight the relevance and potential benefits of the model. First, the model simplifies the complex supplier selection and ordering volume decisions faced by companies into manageable mathematical problems by defining well-defined objective functions, decision variables, and constraints that allow all aspects of the problem to be quantified and directly optimized. Second, the model takes into account fixed demand and supplier capacity constraints for each quarter. By optimizing the model, companies can ensure that market demand is accurately met without exceeding supplier capacity, avoiding inventory backlogs and potential out-of-stock issues. Finally, by adopting a data-based optimization model, companies can make their supply chain decision-making process more transparent and fact-based, helping to reduce the risk of decisions based on hunches or incomplete information.

4. Mathematical Algorithm

Optimization algorithms can be classified into two broad categories based on the method of operation, such as deterministic and probabilistic algorithms (Aguiar e Oliveira Junior et al., 2012). In this paper, deterministic dynamic programming is chosen to solve the problem under study.Dynamic Programming is an algorithmic paradigm that solves complex issues by partitioning them into a collection of simpler subproblems. techniques developed to solve very complex problems by breaking the problem into many sub-problems (de Cooman and Troffaes, 2005). Dynamic programming helps solve tough problems by breaking them down into easier steps. It's like solving a puzzle where figuring out one piece helps you see where the next piece goes(Kavitha and Ratchagar, 2013; de Cooman and Troffaes, 2005). Deterministic Dynamic Programming (DDP) is an algorithm for multi-stage decision problems for situations where the future state of the system and the outcomes associated with the decision are known and predictable. The core of the algorithm is to decompose a complex decision problem into a series of simple subproblems, and to find a globally optimal solution by solving these subproblems step by step.

The core principles of deterministic dynamic programming are optimal substructure and overlapping subproblems. Optimal substructure means that the optimal solution of a problem contains the optimal solutions of its subproblems. This means that the optimal solution of the whole problem can be constructed by combining the optimal solutions of the subproblems. Overlapping subproblems mean that multiple subproblems are repeated during the problem solving process.DDP improves the efficiency by solving each subproblem only once and storing its solution, thus avoiding the repetitive work of computing the same subproblems. The algorithmic mechanism is as follows: first, the entire decision problem is decomposed into a series of stages. Next, define one or more states for each stage to describe the state of the system. Construct a recurrence relation (also riding on the Bellman equation) for computing the optimal solution of the current stage based on the optimal solution of the previous stage. Define the initial or boundary conditions and start from the initial stage and gradually solve the optimal policy for each stage by recursive relations. In the final stage, the optimal path of the whole decision-making process is retraced based on the saved decisions and states.

In the problem studied in this paper, the need to select the optimal supplier and order quantity in each of the four quarters of a year in order to minimize the total inventory cost and meet the product demand in each quarter requires that the decisions for each quarter be optimized.Moreover, the product demand, supplier capacity constraints, and costs are known for each quarter.The problem of deciding which supplier to buy from in each quarter can be viewed as a subproblem that recurs in each quarter.Therefore, this problem is well suited for deterministic planning to solve.

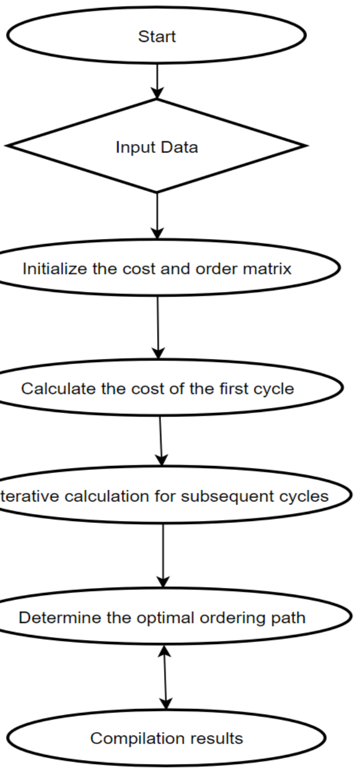

Start
Input Data
Initialize the cost and order matrix
Calculate the cost of the first cycle
Iterative calculation for subsequent cycles
Determine the optimal ordering path
Compilation results

Figure 2. Flowchart Deterministic Dynamic Programming

5. Outcome Interpretation

By applying a deterministic dynamic programming algorithm, we derive the firm's optimal ordering path for each quarter of the causal year, which covers which supplier to order from as well as the ordering quantity for each period.The results show that 30, 35, 40 and 20 pieces are ordered from supplier 1 in each quarter.With the above ordering plan, the firm is able to meet its requirements for the whole year with a minimum inventory cost of 542.5. This study uses a virtual dataset to demonstrate how to utilize the existing precise demand forecasting capabilities of enterprises to achieve the minimization of annual total inventory costs.

The results show the importance of using deterministic dynamic planning approach in supply chain management.It can minimize the total cost of inventory for the year by finely scheduling the supplier selection and order quantities for each quarter. The final result obtained not only matches the demand cycle of the firm, but also takes into account the capacity constraints of the suppliers, realizing a significant reduction in the total cost of inventory and increasing the financial benefits of the firm. In addition, for similar firms in this study, i.e., firms with better product demand forecasting capabilities and more reliable suppliers, such results can guarantee the firm's achievement of its planned annual goals for its sales strategy and help the firm to adopt a targeted marketing strategy that enhances its overall competitiveness in the industry.

6. Limitations

Firstly, this paper uses hypothetical datasets, which would be the first limitation.The use of hypothetical datasets may affect the accuracy and reliability of the results as they do not reflect the various complexities and variations of real business operations, and the conclusions and recommendations obtained at the end may not be able to provide sufficient support for real business decisions. Secondly, the model and deterministic dynamic programming algorithm selected in this paper are helpful in inventory cost optimization but have some inherent limitations which may affect their effectiveness in real-world applications. The algorithms assume that demand, cost, and supplier capacity are deterministic, which ignores uncertainties in real-world business environments such as demand fluctuations, supplier instability, and price volatility. Models typically do not take into account other attributes of suppliers, such as quality, reliability, or the strategic importance of collaboration. In reality, supplier selection is not only based on cost, but may also involve brand risk, supply chain resilience, and many other considerations. The algorithm assumes a linear relationship between holding costs and inventory levels, which may not be applicable to all types of products, especially those with expiration risks or shelf-life constraints. For more complex problems involving multiple products, multiple time period discounts, changes in supplier capacity, seasonal fluctuations in demand, etc., simple deterministic dynamic programming may not provide an effective solution. Models often do not include practical constraints such as inventory space limitations, supplier contract terms, or delivery time windows, which are important in real-world decision making.

The efficiency of the algorithms decreases as the problem size grows, especially when the decision variables and state space increase significantly, and more efficient optimization algorithms or heuristics may be required. Therefore, when applying such models to real-world situations, a clear understanding of these limitations is needed, and consideration should be given to incorporating other methods, such as Monte Carlo simulation, stochastic programming, or machine learning techniques, to compensate for the shortcomings of deterministic dynamic programming. In addition, models may need to be adapted and customized to a particular situation to better reflect the specific environment and strategic objectives of the enterprise.

7. Alternative Methodologies

In this study, we consider several approaches to solve the supplier selection and order quantity optimization problems, focusing on Deterministic Dynamic Programming (DDP), Mixed Integer Linear Programming (MILP) and Markov Decision Processes (MDP). Deterministic Dynamic Programming (DDP) is an effective method for solving multi-stage decision problems, especially for situations where the problem can be decomposed into a series of time-series decisions, each of which depends on the previous state. In the problem studied in this paper, the ordering decision in each cycle depends on the result of the previous cycle, and the supplier's capacity constraints are fixed in each cycle.The advantage of DDP is that it solves the problem incrementally, optimizes the decision stage by stage, and makes the interpretation and implementation of the solution intuitive and easy to manage.Mixed Integer Linear Programming (MILP) is an effective method for solving problems containing integers. MILP) is very powerful when dealing with optimization problems with integer variables and extensive constraints, especially when the problem has to consider multiple decision variables and complex constraints at the same time. Considering that the supplier selection in this research problem is a binary decision problem with integer limitations on ordering quantities, and that costs and constraints can be expressed linearly, MILP can provide a way to find a globally optimal solution by considering all the constraints and objective functions at once. In future scenarios where more complex scenarios with more complex parameters and constraints are studied, MILP can be considered as this approach is usually superior to DDP in terms of computational efficiency and solution accuracy.Markov Decision Processes (MDP) are suitable for situations where uncertainty exists in the decision environment, especially when the state transitions of the system are affected by stochasticity. If there are large uncertainties in demand, supply conditions, or costs in future studies and these uncertainties can be described by probabilistic models, MDP can optimize the expected total cost and quantify the risk under different decisions. This makes MDP ideal for dealing with uncertainty and calculating risk adjustment strategies.

References

Miller, R. (2010) *Inventory control: Theory and practice*. New Jersey: Prentice Hall.

Yugang, Y. & De Koster, M.B.M., 2009. On the Suboptimality of Full Turnover-Based Storage. ERIM Report Series Reference No. ERS-2009-051-LIS. [online] Available at: http://ssrn.com/abstract=1485947 [Accessed 23 April 2024].

Felea, M. (2008) "The role of inventory in the supply chain," *Amfiteatru Economic*, 24, pp. 113.

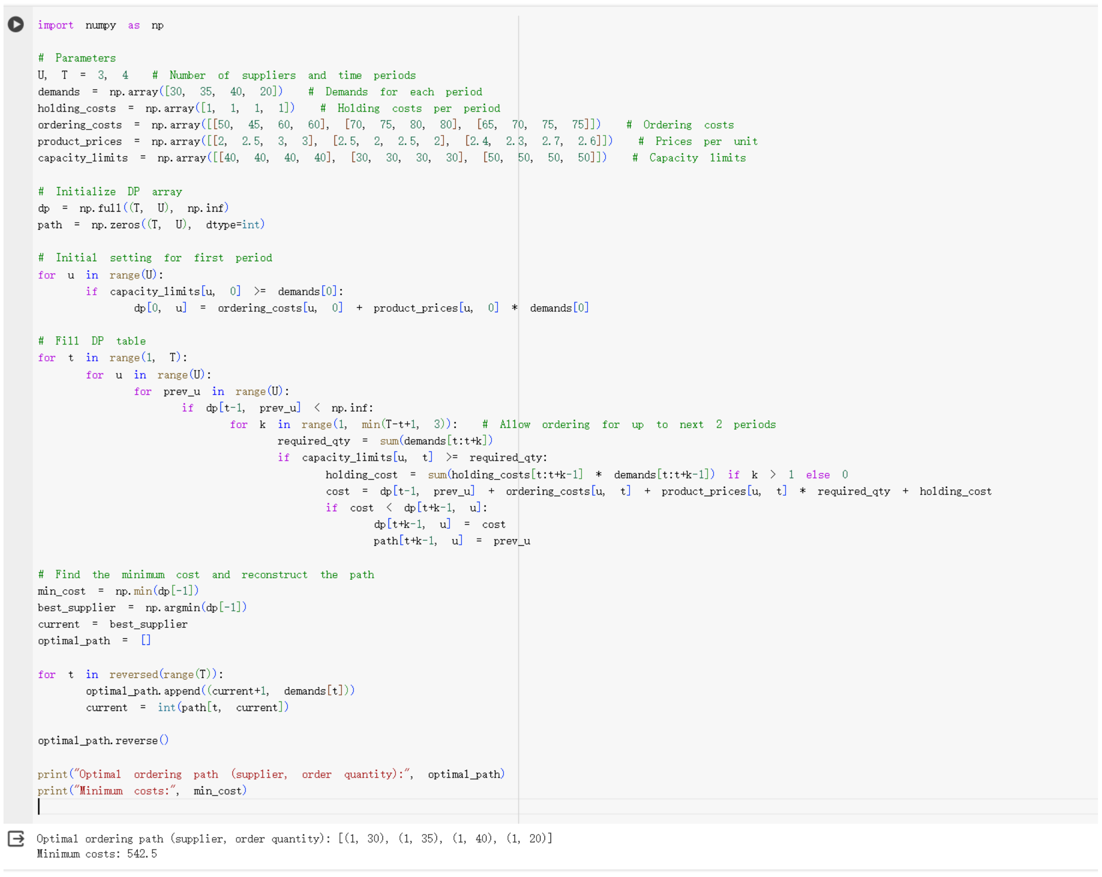
Coyle, J.J., Bardi, E.J. and Langley, C.J. Jr. (2003) *The Management of Business Logistics: A Supply Chain Perspective,* 7th ed. Mason: South-Western.

Aguiar e Oliveira Junior, H., Ingber, L., Petraglia, A., Rembold Petraglia, M. and Soares Machado, M.A., 2012. *Global Optimization and Its Applications*. Berlin, Heidelberg: Springer, pp.11-20.

S. Kavitha and N. P. Ratchagar, 2013. A Deterministic Dynamic Programming Approach for Optimization Problem with Quadratic Objective Function and Linear Constraints. *International Journal of Mathematical, Computational, Physical, Electrical and Computer Engineering,* 7(3), pp.565-569.

de Cooman, G. and Troffaes, M.C.M., 2005. Dynamic programming for deterministic discrete-time systems with uncertain gain. International Journal of Approximate Reasoning, 39(2-3), pp.257-278.

Appendices

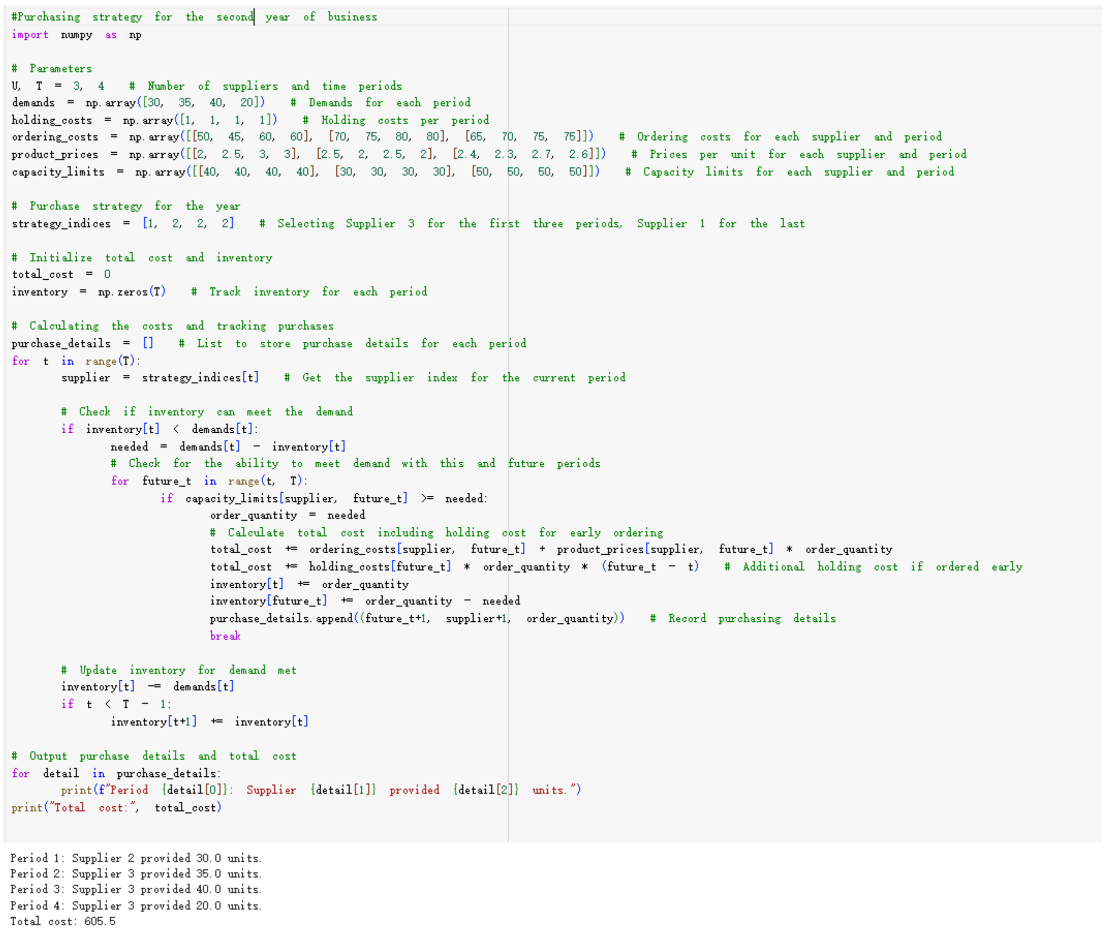


```python
import numpy as np

# Parameters
U, T = 3, 4   # Number of suppliers and time periods
demands = np.array([30, 35, 40, 20])   # Demands for each period
holding_costs = np.array([1, 1, 1, 1])   # Holding costs per period
ordering_costs = np.array([[50, 45, 60, 60], [70, 75, 80, 80], [65, 70, 75, 75]])   # Ordering costs
product_prices = np.array([[2, 2.5, 3, 3], [2.5, 2, 2.5, 2], [2.4, 2.3, 2.7, 2.6]])   # Prices per unit
capacity_limits = np.array([[40, 40, 40, 40], [30, 30, 30, 30], [50, 50, 50, 50]])   # Capacity limits

# Initialize DP array
dp = np.full((T, U), np.inf)
path = np.zeros((T, U), dtype=int)

# Initial setting for first period
for u in range(U):
    if capacity_limits[u, 0] >= demands[0]:
        dp[0, u] = ordering_costs[u, 0] + product_prices[u, 0] * demands[0]

# Fill DP table
for t in range(1, T):
    for u in range(U):
        for prev_u in range(U):
            if dp[t-1, prev_u] < np.inf:
                for k in range(1, min(T-t+1, 3)):   # Allow ordering for up to next 2 periods
                    required_qty = sum(demands[t:t+k])
                    if capacity_limits[u, t] >= required_qty:
                        holding_cost = sum(holding_costs[t:t+k-1] * demands[t:t+k-1]) if k > 1 else 0
                        cost = dp[t-1, prev_u] + ordering_costs[u, t] + product_prices[u, t] * required_qty + holding_cost
                        if cost < dp[t+k-1, u]:
                            dp[t+k-1, u] = cost
                            path[t+k-1, u] = prev_u

# Find the minimum cost and reconstruct the path
min_cost = np.min(dp[-1])
best_supplier = np.argmin(dp[-1])
current = best_supplier
optimal_path = []

for t in reversed(range(T)):
    optimal_path.append((current+1, demands[t]))
    current = int(path[t, current])

optimal_path.reverse()

print("Optimal ordering path (supplier, order quantity):", optimal_path)
print("Minimum costs:", min_cost)
```

```
Optimal ordering path (supplier, order quantity): [(1, 30), (1, 35), (1, 40), (1, 20)]
Minimum costs: 542.5
```

```
#Purchasing strategy for the first year of business
import numpy as np

# Parameters
U, T = 3, 4  # Number of suppliers and time periods
demands = np.array([30, 35, 40, 20])  # Demands for each period
holding_costs = np.array([1, 1, 1, 1])  # Holding costs per period
ordering_costs = np.array([[50, 45, 60, 60], [70, 75, 80, 80], [65, 70, 75, 75]])  # Ordering costs for each supplier and period
product_prices = np.array([[2, 2.5, 3, 3], [2.5, 2, 2.5, 2], [2.4, 2.3, 2.7, 2.6]])  # Prices per unit for each supplier and period
capacity_limits = np.array([[40, 40, 40, 40], [30, 30, 30, 30], [50, 50, 50, 50]])  # Capacity limits for each supplier and period

# Purchase strategy for the year
strategy_indices = [1, 0, 0, 0]  # Selecting Supplier 3 for the first three periods, Supplier 1 for the last

# Initialize total cost and inventory
total_cost = 0
inventory = np.zeros(T)  # Track inventory for each period

# Calculating the costs and tracking purchases
purchase_details = []  # List to store purchase details for each period
for t in range(T):
    supplier = strategy_indices[t]  # Get the supplier index for the current periodAA

    # Check if inventory can meet the demand
    if inventory[t] < demands[t]:
        needed = demands[t] - inventory[t]
        # Check for the ability to meet demand with this and future periods
        for future_t in range(t, T):
            if capacity_limits[supplier, future_t] >= needed:
                order_quantity = needed
                # Calculate total cost including holding cost for early ordering
                total_cost += ordering_costs[supplier, future_t] + product_prices[supplier, future_t] * order_quantity
                total_cost += holding_costs[future_t] * order_quantity * (future_t - t)  # Additional holding cost if ordered early
                inventory[t] += order_quantity
                inventory[future_t] += order_quantity - needed
                purchase_details.append((future_t+1, supplier+1, order_quantity))  # Record purchasing details
                break

    # Update inventory for demand met
    inventory[t] -= demands[t]
    if t < T - 1:
        inventory[t+1] += inventory[t]

# Output purchase details and total cost
for detail in purchase_details:
    print(f"Period {detail[0]}: Supplier {detail[1]} provided {detail[2]} units.")
print("Total cost:", total_cost)
```

```
Period 1: Supplier 2 provided 30.0 units.
Period 2: Supplier 1 provided 35.0 units.
Period 3: Supplier 1 provided 40.0 units.
Period 4: Supplier 1 provided 20.0 units.
Total cost: 577.5
```



```python
#Purchasing strategy for the second year of business
import numpy as np

# Parameters
U, T = 3, 4  # Number of suppliers and time periods
demands = np.array([30, 35, 40, 20])  # Demands for each period
holding_costs = np.array([1, 1, 1, 1])  # Holding costs per period
ordering_costs = np.array([[50, 45, 60, 60], [70, 75, 80, 80], [65, 70, 75, 75]])  # Ordering costs for each supplier and period
product_prices = np.array([[2, 2.5, 3, 3], [2.5, 2, 2.5, 2], [2.4, 2.3, 2.7, 2.6]])  # Prices per unit for each supplier and period
capacity_limits = np.array([[40, 40, 40, 40], [30, 30, 30, 30], [50, 50, 50, 50]])  # Capacity limits for each supplier and period

# Purchase strategy for the year
strategy_indices = [1, 2, 2, 2]  # Selecting Supplier 3 for the first three periods, Supplier 1 for the last

# Initialize total cost and inventory
total_cost = 0
inventory = np.zeros(T)  # Track inventory for each period

# Calculating the costs and tracking purchases
purchase_details = []  # List to store purchase details for each period
for t in range(T):
    supplier = strategy_indices[t]  # Get the supplier index for the current period

    # Check if inventory can meet the demand
    if inventory[t] < demands[t]:
        needed = demands[t] - inventory[t]
        # Check for the ability to meet demand with this and future periods
        for future_t in range(t, T):
            if capacity_limits[supplier, future_t] >= needed:
                order_quantity = needed
                # Calculate total cost including holding cost for early ordering
                total_cost += ordering_costs[supplier, future_t] + product_prices[supplier, future_t] * order_quantity
                total_cost += holding_costs[future_t] * order_quantity * (future_t - t)  # Additional holding cost if ordered early
                inventory[t] += order_quantity
                inventory[future_t] += order_quantity - needed
                purchase_details.append((future_t+1, supplier+1, order_quantity))  # Record purchasing details
                break

    # Update inventory for demand met
    inventory[t] -= demands[t]
    if t < T - 1:
        inventory[t+1] += inventory[t]

# Output purchase details and total cost
for detail in purchase_details:
    print(f"Period {detail[0]}: Supplier {detail[1]} provided {detail[2]} units.")
print("Total cost:", total_cost)
```

```
Period 1: Supplier 2 provided 30.0 units.
Period 2: Supplier 3 provided 35.0 units.
Period 3: Supplier 3 provided 40.0 units.
Period 4: Supplier 3 provided 20.0 units.
Total cost: 605.5
```