Algorithm Documentation: Optimal Product Allocation from Multiple Warehouses to Multiple Orders

---

Overview:

The algorithm tackles the problem of optimally allocating a specific type of product from various warehouses to multiple orders. The objective is to minimize the maximum transportation time using Integer Linear Programming (ILP).

Mathematical Formulation:

Variables:

- (X): Transport time matrix of size (n \times m), where (X_{i,j}) is the time taken to transport the product from warehouse (j) to order (i).
- (Y): Demand vector of size (n), where (Y_i) is the demand for order (i).
- (Z): Inventory vector of size (m), where (Z_j) denotes the available stock in warehouse (j).
- (A): Allocation matrix of size (n \times m), where (A_{i,j}) is the volume of product allocated from warehouse (j) to order (i). This is our primary decision variable.
- (M): A scalar variable representing the maximum transportation time among all allocations.

Objective Function:

[\min M]

The primary objective is to minimize (M), which captures the maximum transportation time in the entire allocation matrix.

Constraints:

1. **Max Transport Time Constraint**:

[M \geq X_{i,j} \times A_{i,j} \quad \forall i \in {1,2,\ldots,n}, j \in {1,2,\ldots,m}]

For every combination of order (i) and warehouse (j), this constraint ensures that (M) is always greater than or equal to the product of the transport time and the allocated volume.

2. **Order Demand Constraint**:

[\sum_{j=1}^{m} A_{i,j} = Y_i \quad \forall i \in {1,2,\ldots,n}]

This ensures that the total allocation for each order matches its demand.

3. **Warehouse Inventory Constraint**:

$$\sum_{i=1}^{n} A_{i,j} \leq Z_j \quad \forall j \in \{1,2,\ldots,m\}$$

This guarantees that allocations from each warehouse do not exceed its inventory.

Algorithm Description:

Step 1: Input Validation

- Verify the inputs (Transport Time Matrix (X), Demand Array (Y), Stock Array (Z)) to ensure they are valid and consistent with each other in terms of dimensions and values.

Step 2: Initialize Allocation Matrix (A)

- Create an empty matrix (A) with the same dimensions as the Transport Time Matrix (X).

Step 3: Calculate Initial Allocation

- Distribute products according to an initial algorithm, perhaps starting with the shortest transport times first, while adhering to the constraints.

Step 4: Optimize for Minimum Maximum Transport Time

- Use an optimization algorithm (such as Linear Programming or other optimization techniques) to adjust the initial allocation in (A). The objective is to minimize the maximum transportation time, which is calculated by the Hadamard product of (A) and (X).

Step 5: Validate Final Allocation

- Ensure that the final allocation (A) satisfies all constraints:
    1. Sum of allocations for each order (i) should be equal to (Y_i).
    2. Sum of allocations for each warehouse (j) should not exceed (Z_j).
    3. All elements in (A) should be greater than or equal to 0.

Step 6: Calculate Minimized Maximum Transport Time

- Calculate the minimized maximum transport time as per the final (A) and (X).

Step 7: Output

- Return the Allocation Matrix (A) and the minimized maximum transport time as the outputs.

After laying out these steps, the next phases would involve implementing the code, followed by testing with sample data to validate that the algorithm is functioning as expected.

Conclusion:

TODO: need update, on how outer system call this API, specifically, it call this algo for diff product.

By using this algorithm, supply chain managers can make informed decisions about product allocation, optimizing transportation time, and ensuring timely deliveries. The mathematical formulation provides

clarity on the constraints and objectives, making the approach more transparent and understandable.