Reagan Bennos, Dalton Chenoweth, Stephen Desilets, Cesar Martinez

8/5/2023

**Methodology for Brewery Distribution Problem**

**1. Problem Formulation**

The brewery distribution problem is a transportation optimization problem aimed at minimizing transportation costs while satisfying customer demands for different commodities at various demand points. The objective is to determine the optimal shipping quantities between four distinct breweries and three distinct packaging facilities and between packaging facilities and fifteen different demand points.

**2. Decision Variables**

We have defined the decision variables as follows:

1. `inputs\_1[i][j]`: The quantity of commodity manufactured and shipped from brewery `i` to packaging facility `j`.

2. `inputs\_2[j][k]`: The quantity of commodity shipped from packaging facility `j` to demand point `k`.

**3. Objective Function**

The objective function is to minimize the total transportation costs, which includes the costs from breweries to packaging facilities and from packaging facilities to demand points.

**Objective Function:**

Minimize: ∑[inputs\_1[i][j] \* brew\_to\_pack\_shipping\_costs[i][j] for i in brewing, j in packaging] +

∑[inputs\_2[j][k] \* pack\_to\_demand\_shipping\_costs[container][k][j] for j in packaging, k in demanding, container in pack\_to\_demand\_shipping\_costs]

**4. Constraints**

1. Demand Constraints: The quantities shipped from packaging facilities to demand points must meet or exceed the demand for each commodity at each demand point.

∀ c in commodities, ∀ k in base\_demand.keys():

∑[inputs\_2[j][k] \* commodities[c]["Container"] for j in packaging] >= demand[c][k] if k in demand[c] else 0

1. Brewery Capacity Constraints: The quantities shipped from breweries to packaging facilities must not exceed the maximum manufacturing capacity and must meet the minimum capacity requirements for each brewery.

∀ i in brewing:

∑[inputs\_1[i][j] for j in packaging] <= brewing\_maximum[i]

∑[inputs\_1[i][j] for j in packaging] >= brewing\_minimum[i]

1. Packaging Facility Capacity Constraints: The quantities shipped from packaging facilities to demand points must not exceed the maximum packaging capacity and must meet the minimum packaging capacity requirements for each packaging facility.

∀ j in packaging:

∑[inputs\_2[j][k] for k in demanding] <= packaging\_maximum[j]

∑[inputs\_2[j][k] for k in demanding] >= packaging\_minimum[j]

1. 4. Packaging Output Constraint: The total output from each packaging facility must be equal to the total input from the corresponding brewery.

∀ j in packaging:

∑[inputs\_2[j][k] for k in demanding] == ∑[inputs\_1[i][j] for i in brewing]

1. Brewery Liquid Commodity and Container Constraints: The quantities of specific commodities and containers used by each brewery must not exceed the specified maximum liquid quantity for that commodity.

∀ i in brewing, ∀ c in commodities:

required\_liquid = commodities[c]["Liquid"]

max\_qty = liquid\_constraints[required\_liquid]["Max Qty"]

if breweries\_liquids\_config[i][required\_liquid] == 1:

∑[inputs\_1[i][j] \* commodities[c]["Container"] for j in packaging] <= max\_qty

else:

0 <= 0 (No effect, since the brewery doesn't use the liquid)

**5. Output**

The model code outputs the optimal solution for the brewery distribution problem, including:

1. Status: The status of the optimization problem (e.g., Optimal, Infeasible, Unbounded, etc.).

2. Shipping quantities from each brewery to each packaging facility.

3. Shipping quantities from each packaging facility to each demand point.

4. Aggregated brewing and packaging outputs.

5. Total demand and total shipping costs.

**Appendix:**

The summary below provides an explanation of the variables and data used in the Python script. Here's a brief description of the key variables and data used:

**Variables:**

brewing: A list containing the names of the breweries.

brewing\_maximum: A dictionary representing the maximum beer that can be manufactured by each brewery.

brewing\_minimum: A dictionary representing the minimum units of beer that must be manufactured at each brewery.

packaging: A list containing the names of the packaging facilities.

packaging\_maximum: A dictionary representing the maximum capacity of each packaging facility.

base\_demand: A dictionary representing the aggregate demand for each demand point (DP).

packaging\_minimum: A dictionary representing the minimum capacity required at each packaging facility.

total\_demand: The sum of all demands across all demand points.

brew\_to\_pack\_shipping\_costs: A 2D list representing the transportation costs from each brewery to each packaging facility.

pack\_to\_demand\_shipping\_costs: A nested dictionary representing the transportation costs from each packaging facility to each demand point for different commodities.

**Data:**

commodities: A dictionary containing the information about the commodities, including the liquid and container they represent.

demand: A dictionary representing the demand for each commodity at each demand point.

liquid\_constraints: A dictionary specifying the maximum liquid quantities for different types of liquids.

breweries\_liquids\_config: A dictionary representing which liquids are used by each brewery.

routes\_1 and routes\_2: Lists containing tuples representing the possible combinations of shipping routes from breweries to packaging facilities and from packaging facilities to demand points, respectively.