Data Scientist Logistics Case

The assignment

When a customer orders products in our web shop, these products need to be delivered to the customer from one or more of our warehouses.

At bol.com we try to process and ship these orders in the most cost-effective way.
To do so, we need to make some important decisions. In this case we focus on two of them:

1. **Stock allocation:** The decision which products from our assortment do we keep on stock at which warehouse(s).
2. **Order sourcing:** How do we break up a shop order (an order from a customer on the web shop) into one or more warehouse orders. A warehouse order consists of one or more products that are processed on a pack line of that warehouse. Each warehouse order results in a shipment.

Setting

Consider a situation where we would have an assortment of 1.000 different products on our web shop. We have two warehouses (*warehouse A* and *warehouse B*) available from where we can deliver our products and each warehouse has the capacity to hold stock for only a selection of 600 products. In this assignment we assume that if a specific product is on stock in a certain warehouse, the stock quantity of that product is unlimited.

Costs

The operating costs in each warehouse consist of two components: the operating costs of processing products in the warehouse, and the shipping costs of shipping a package.

Pack lines

Each warehouse has two pack lines that can process products:

- **A MONO pack line:** from this line we can only process and ship packages containing a single product. This means that we cannot combine multiple products from the same order into one shipment.
- **A MULTI pack line:** from this line we can process and ship packages containing multiple products.

Each pack line in each warehouse comes with its own operating costs per product and shipping costs per shipment. Besides that, each pack line has a capacity indicating the number of products that it can process per day.

Order sourcing

When a shop order comes in, it needs to be transformed into one or more warehouse orders. The logic of how a shop order is transformed into warehouse orders is described under the header ‘Order sourcing logic’ and is already implemented by us.

A warehouse order can only be placed at a warehouse if:

1. All products are in stock.
2. The chosen pack line did not exceed its daily capacity yet.

If products from shop orders cannot be sourced from any of the warehouses, we consider them as lost sales. For each lost sale we calculate a fixed lost sale cost.

Stock allocation

It is your task to decide which product from our assortment we allocate to which warehouse. In the end, all we ask from you is to construct a table stating for each product id whether it is in stock in *warehouse A* and whether it is in stock in *warehouse B, or both*. The table will should look like the table below. Bear in mind that each warehouse can have at most 600 different products on stock.

productId	**warehouseA**	**warehouseB**
00 001	True	False
00 002	False	True
00 003	True	True
…	…	…
01 000	False	True

Material

We supplied you with the following material.

- Data: in the data folder you find all the data that can be used to solve this case, namely:
  - Assortment data (assortment.csv)
  - Historical order data (shop_orders.csv)
- Jupyter Notebook: a Jupyter Notebook containing a naïve example solution of the problem.
- Order sourcing module: A module containing the order sourcing simulation, which is imported into the Jupyter Notebook.

Order sourcing logic

A naïve stock allocation solution can be found in the notebook described above. In this notebook, a function called "*simulate"* is imported and available to run an order sourcing simulation. How orders are sourced is shown in diagram 1. This function requires a dataframe of shop orders and a dataframe of stock allocations. It returns a "*SimulationResult"* object containing the properties "*warehouse_order"* and "*lost_sales"* which are two Dataframes ready to use for analysis. In addition, a method called "*show_aggregated_results"* is available to report the number of warehouse orders sourced, the total costs, and the number of products that couldn't be sourced.

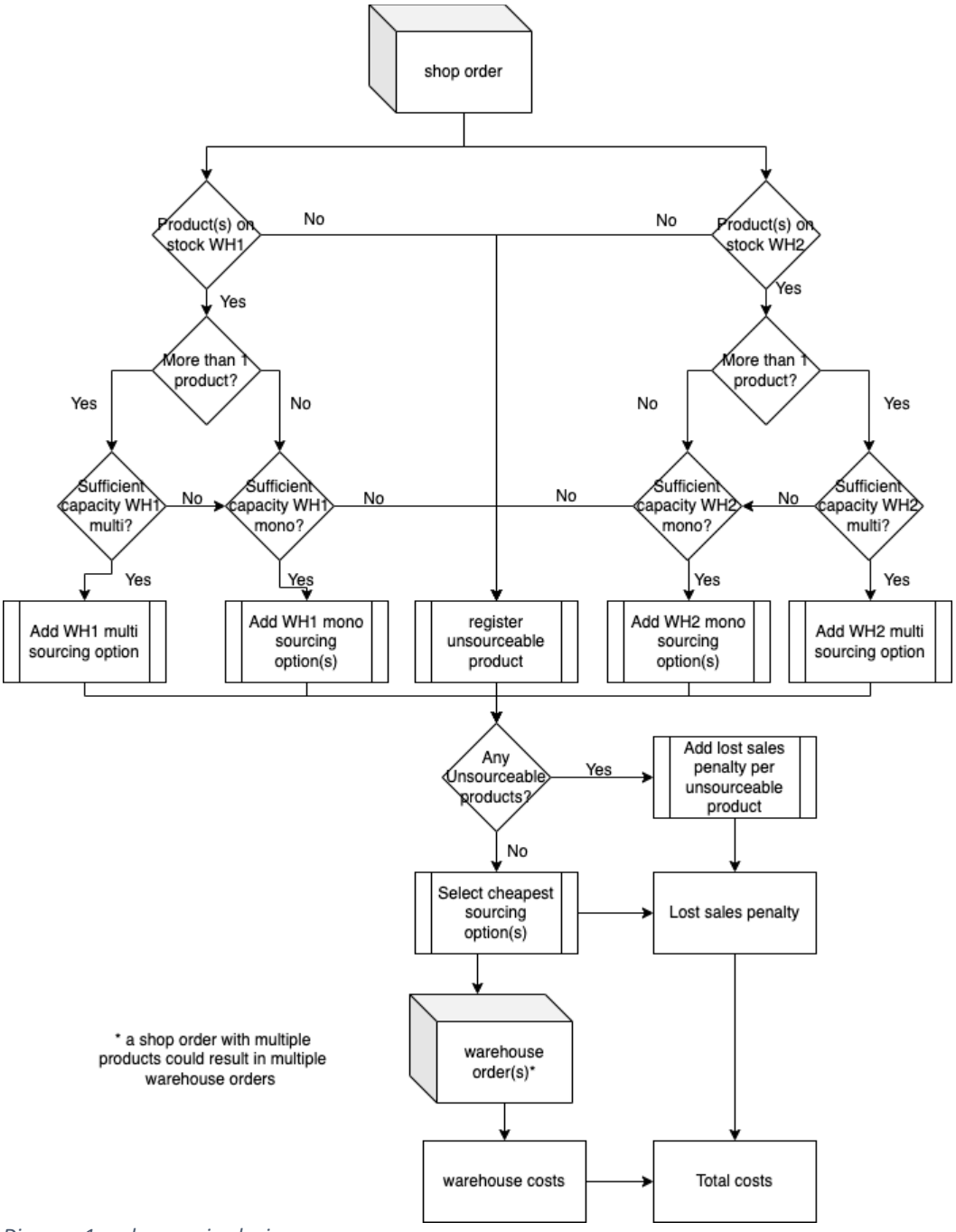

Diagram 1: order sourcing logic

Constraints and Costs

As described above, there are two warehouses with each two pack lines. As can be seen in the specifications below, each pack line has a daily processing capacity of products and its associated costs. Also, the assumed cost per lost sale can be found in the specifications below. Note that the *cost_per_shipment* is per shipment, which can be multiple products, and the *LOST_SALES_PENALTY* is per product.

```
warehouses = [
{"name": "warehouseA",
 "pack_lines": [
     {"name": "monoManual",
      "type": "mono",
      "capacity": 100,
      "cost_per_product": 0.7},
     {"name": "multiManual",
      "type": "multi",
      "capacity": 200,
      "cost_per_product": 0.9}
 ],
 "cost_per_shipment": 5,
 "stock_capacity": 600
 },
{"name": "warehouseB",
 "pack_lines": [
     {"name": "monoManual",
      "type": "mono",
      "capacity": 300,
      "cost_per_product": 0.5},
     {"name": "multiManual",
      "type": "multi",
      "capacity": 200,
      "cost_per_product": 0.7}
 ],
 "cost_per_shipment": 6,
 "stock_capacity": 600
 }
]

LOST_SALES_PENALTY = 10.0
```

Assignment Goal

Your goal is to minimize the total costs, which include the lost sales penalty, by improving stock allocation for the year 2021. We encourage you to implement a heuristic solution that demonstrates your understanding of the assignment, leveraging insights into the order sourcing logic and the provided data. We are primarily interested in how you approach and break down the problem. Use the Jupyter notebook and ensure that we can follow your line of thought, for example, by using visualizations, comments, and so on. During the interview, you will guide us through your notebook and code.

Be careful to spend no more than 4 hours on the assignment. Please keep your solution relatively simple and easy to implement. Feel free to add further explanations (in the notebook or in separate slides/text) about how you would have approached the problem if you had more time.

Bonus question (no implementation needed)

Instead of changing the stock allocation, one could also modify the order sourcing logic to reduce costs. What would you change in the order sourcing logic, such that costs can be further reduced? Please share your thoughts as comments in the notebook or in separate slides/text.