Operations Research, Spring 2021 (109-2)
Case Assignment 2

Instructor: Ling-Chieh Kung
Department of Information Management
National Taiwan University

1 Submission rules

- This case assignment is due at ***23:59, May 9***. Those who submit their works late but are late by less than one hour gets 10 points off. Works that are late more than one hour get no point.
- For this case assignment, students should work in teams. Each team should contain ***either four or five*** students. Teams that do not follow this rule gets no point.
- For each team, **one and exactly one** student should submit their work on behalf of all team members. Please submit a ***ZIP file*** containing a ***PDF file*** and a ***Python program*** (i.e., a .py file) through NTU COOL. Make sure that the submitted PDF file contains the student IDs and names. The Python program is only for Problem 3. Those who fail to do these will get 10 points off.
- You are required to ***type*** your work with LaTeX (strongly suggested) or a text processor with a formula editor. Hand-written works are not accepted. You are responsible to make your work professional in mathematical writing by following at least those specified in homework assignments. Those who fail to follow these rules may get at most 10 points off.
- The maximum length of the report is ***six pages***, including everything. Everything starting from page 7 will not be graded.

2 The tasks

Recall that there is a company called IEDO which imports products to sell to domestic consumers. In Case Assignment 1, we went through four problems, where the last problem best describe the real situation faced by the IEDO company. In that problem,

the company must determine when, how many, and through which method to order each product. Factors that must be taken into considerations include future demands, inventory costs, shortage costs (with lost sales and backorders), fixed costs for shipping, container costs for ocean freight, etc. Its objective is to minimize total cost.

In practice, there are more issues to consider. In this case study we introduce three of them. First, a retailer like the IEDO company typically purchases products from multiple vendors (brands). In each period, if IEDO purchase any unit of any product through any shipping method from a vendor, a fixed cost is incurred to process the order. In the MS Excel files that will be provided to you, the vendor-product mapping and the fixed ordering costs are provided in the sheets "Vendor-Product" and "Vendor cost", respectively. Second, there may be a lower bound for the order quantity of a product in a period, which requires the IEDO company to order at least that amount (sum over all shipping methods) in a period *if and only if* it orders that product in that period. These bounds are in the sheet "Bounds". Finally, there may be some pairs of two products that cannot be ordered in the same period (regardless of the shipping method). This information is provided in the sheet "Conflict", where each row contains the product IDs of the two products that cannot be ordered in the same period.

Below are the tasks you need to do. Throughout this case assignment, please set your order quantity for any product through any shipping method in any period as an *integer*.

1. (15 points) Add constraints and variables to and modify the objective function in your formulation for Problem 4 in Case Assignment 1 to model the three new issues listed above. Just write down your mathematical model is enough. Do not provide your source codes in Python, C++, or any programming language.

2. (15 points) You are given five MS Excel files, one for an instance. Write a computer program invoking Gurobi Optimizer to solve the five instances by finding optimal solutions. Write down the objective values of the five optimal solutions. Do not provide your source codes. Do not write down your ordering plan. In other words, you only need to write down five numbers and indicate their corresponding file names.

3. (30 points) In practice, instance size may be quite large if you are doing good business. You may order from more than ten vendors for a thousands of products; your time unit may be weeks rather than months; there may be hundreds of pairs of products that cannot be ordered together. It is thus possible that an MIP solver

like Gurobi Optimizer cannot give you an optimal solution in a reasonable amount of time. You then need a heuristic algorithm.

Please design and implement a heuristic algorithm to obtain a near-optimal solution by using a short amount of time. You will submit your implementation for the instructing team to input another five instances. These five instances, whose sizes are so large that relying on Gurobi Optimizer is not a practical idea, will be provided to you only after this assignment is due. Each instance counts for $b = \frac{30}{5} = 6$ points.

The instructing team will provide you a program in which file loading, data processing, time counting, feasibility check, and objective value calculation are all implemented. You need to follow the specification of the file to implement a function that perform your heuristic algorithm. For each instance, all submissions are graded together. Those submissions whose implementation results in syntax errors, run-time errors, or infeasible solutions will get no points. Those that give feasible solutions will be graded in the following way:

(a) For submission i, $i = 1, ..., n$, where n is the number of submissions, the instructing team will calculate its objective value z_i and computation time t_i. Figure 1 provides an example of six submissions. While submission 6 is quite time efficient, its objective value is high. On the contrary, submission 1 performs well in objective values yet time-consuming.

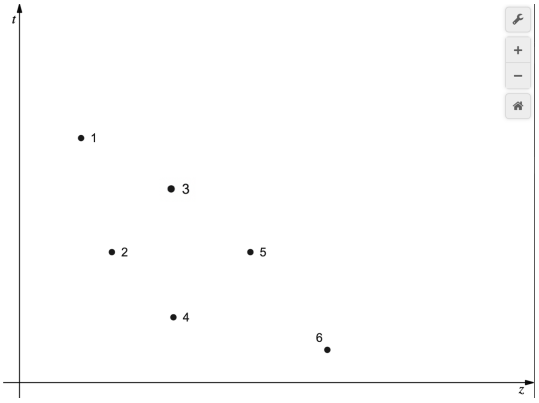

Figure 1: Example of submissions

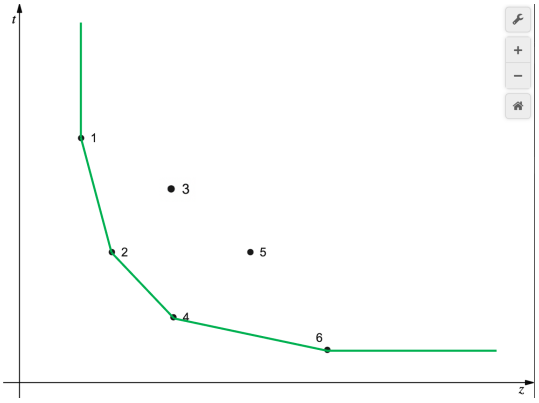

Figure 2: Example of a Pareto frontier

(b) All the submissions that lie on the *Pareto frontier* among the n submissions, i.e., not dominated by another submission, will get b points. Submission i is dominated by submission j if $z_i > z_j$ and $t_i > t_j$. As Figure 2 shows, in our example the Pareto frontier is formed by connecting submissions 1, 2, 4, and 6 (and extends to infinity from submissions 1 and 6). Submission 3 is dominated by submission 2. Submission 5 is dominated by both submissions 2 and 4.

(c) Those submissions that do not lie on the Pareto frontier will be graded according to their relative distance to the Pareto frontier in the dimension of objective value. More precisely, if submission i is not on the Pareto frontier, and the point (z', t_i) is on the Pareto frontier, submission i gets $(\frac{z'}{z_i})b$ points. For example, from Figure 3 we may see that submissions 3 and 5 will get roughly $\frac{1}{2}b$ and $\frac{2}{5}b$ points, respectively.

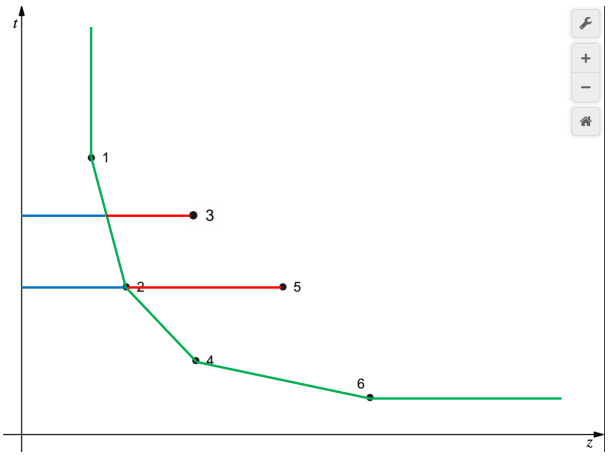

Figure 3: Example of submissions

The testing data in the five .xlsx files are in the "data" folder. The following are some informations hold for each instance and not included in the .xlsx files: We consider the next 26 periods (weeks). There are three methods for delivery: express delivery, air freight, and ocean freight. Each of them takes one, two, and three periods shipping times and \$100, \$80, and \$50 as the fixed costs per order, respectively. If you want to use ocean freight, the products should be delivered by containers. One container has size 0.5 CBM and costs you \$1500. Please note that the container size is no longer 30 CBM as in Case Assignment 1.

You are provided two .py files: algorithm_module.py and CS2_grading_program.py. You should implement your algorithm in algorithm_module.py. DO NOT change the function name and file name and make sure that this function returns the ordering plan in list. The instructing team will use CS2_grading_program.py to call the heuristic_algorithm funcion in the file algorithm_module.py you submit. Therefore, please ensure that your function may be called correctly. You may check out detailed instructions in these two .py files.

Important note. There is no restriction on the design of your heuristic algorithm. However, you must implement your heuristic algorithm in Python. We understand that it is better if there is no restriction on the programming language. However, as

we need to specify the input/output to give you a precise development instruction, facilitate grading, and use the same basis to time all submissions, we are sorry that we must make such a restriction.

4. (40 points) Use your own words to introduce your heuristic algorithm to the instructing team. You may write paragraphs of words, draw flaw charts, generate pseudocodes, and/or provide examples. Points will be given to you according to the logic of your algorithm and clearness of your description.