Operations Research, Spring 2025 (113-2)
Midterm Project

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

1 Submission rules

- This midterm project is due at ***23:59, May 10***. Submissions late by no more than twelve hours get 10 points off; those late by no more than 24 hours get 20 points off; those late by more than 24 hours get no point.
- For this midterm project, students should work in teams. Unless a student gets special approval from the instructor, her/his team members should be the same as those for the final project.
- 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***, some ***computer programs*** (in C++, Python, or any common modern high-level languages), and a ***CSV or XLSX file*** to NTU COOL. Make sure that the submitted PDF file contains the student IDs and names. Those who fail to do these will get 10 points off.
  - The computer programs should contain everything you implement for this project, including but are not limited to a program that invokes a solver for Problem 1, a program that invokes a solver for Problem 2, a program that implements a heuristic for Problem 3, several programs that generate instances, implement the two benchmarks, and run the experiment for Problem 4, and several programs that complete the experiment in Problem 5.
  - These computer programs should be grouped into five folders, one for each problem, and named reasonably. It should be easy for the TAs to check these computer programs to verify whether the solutions and experiment results in the report can be reasonably generated by the computer programs. In most cases, the TAs will not try to execute your computer programs; however, if they find obvious inconsistency between the computer programs and the outcomes, they will try to check whether the report is written without using

the submitted computer programs in a right way. If so, the team will lose (many) points accordingly.

  - The CSV or XLSX file should contain detailed experiment results in the following format. It should contain 210 rows, one for each instance. It should contain the following columns: (1) scenario ID, (2) instance ID, (3)–(5) objective values reported by the LP relaxation, proposed algorithm, and very naive heuristics, (6)–(8) computation times required to run the three solution approaches, and (9)–(10) two optimality gaps in percentages. In most cases, the TAs will not try to check the correctness of all these numbers; however, if they find obvious inconsistency among the result file, the computer programs, and the numbers in the report, they will try to check whether the report is written without conducting the experiment and summarizing the raw results in a right way. If so, the team will lose (many) points accordingly.

- 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 points off.

- The maximum length of the report is ***ten pages***, including everything. Everything starting from page 11 will not be graded.

2 The problem and an example instance

2.1 The case

IEDO is a company which imports products from an overseas manufacturer to sell to domestic consumers. As a reseller, the company's business relies heavily on purchasing and the control of inventory. On one hand, if a consumer cannot get a product when she/he wants, lost sales occurs and hurts the company's profit. On the other hand, if there are too many unsold products, holding costs emerge.

You are newly hired as an operations researcher by the company. Right after you started to work, you realized why the company hired you. In the past, they made all the purchasing decisions, including purchasing quantities, shipping methods, and the timing of ordering, by hands, pencil, paper, spreadsheet software, and experience. That was not bad, but indeed there is room for improvement.

Today, you sat down at your desk and started to organize all the information you collected in the past two weeks. The ordering cycle time is a month. More precisely, at the beginning of each month, the company needs to determine the order quantities and shipping methods of all products and place an order to the manufacturer. While the ordering decision must be made for every month, you have decided to start by considering the next ordering timing, March 1, as an example.

2.1.1 Demand and inventory holding costs

The company orders ten products from the manufacturer. Monthly forecast demands as well as the current on-hand inventory levels of all products in the next six months are given in Table 1. For example, for product 2 it is estimated that the monthly demand for April will be 101 units. Note that for some products the initial inventory levels are not enough to cover future demands. Purchasing should thus be considered.

Product	Initial inventory	Monthly forecast demands					
		March	April	May	June	July	August
1	800	138	55	172	194	94	185
2	600	190	101	68	185	13	136
3	425	79	179	21	49	199	200
4	350	142	103	78	131	146	155
5	400	35	62	83	90	197	49
6	524	91	95	107	127	116	183
7	453	105	164	19	116	119	175
8	218	37	155	10	77	168	32
9	673	108	185	188	176	81	172
10	200	46	178	162	200	154	199

Table 1: Monthly forecast demands and initial inventory levels

Inventory should be prepared to face future demands and avoid costs related to inventory holding. Holding one unit of a product by one month incurs a unit holding cost for that product (due to warehousing and interest loss, among others). The company take 2% of the unit purchasing cost as the per unit per month holding cost of a product. The purchasing costs per unit and holding costs per unit per month of all products are listed in Table 2. For example, it costs the company \$2,000 to purchase one unit of product 2, and that means holding one unit of product 2 for one month costs the company \$2,000 $\times$ 0.02 = \$40.

Product	Price ($)	Costs ($)	
		Purchasing	Holding
1	10700	5000	100
2	5000	2000	40
3	26900	9000	180
4	19600	9000	180
5	6900	2000	40
6	31500	9000	180
7	14100	7000	140
8	16400	5000	100
9	33000	9000	180
10	18600	7000	140

Table 2: Sales prices, purchasing costs, and holding costs

As an example, suppose that your initial inventory for product 1 is 800 units, and you order 38 units of product 1 that will be available for sales in May. It then follows that the ending inventory for product 1 will be 662, 607, 473, 279, 185, and 0 for each of the following six months. The total inventory holding cost is thus

$$\$100 \times (662 + 607 + 473 + 279 + 185 + 0) = \$220{,}600.$$

2.1.2 Shipping methods

There are three shipping methods to choose from: express delivery, air freight, and ocean freight. The three methods have different lead times and costs. For each method, the lead time include the processing time needed by the manufacturer and shipper, custom clearance, and the processing time needed ourselves, etc. In any case, it is the time it needs between the order is placed and the products are ready to be sold to end consumers. To keep things simple, the company says you may assume that an order is always placed on the first day of a month, and products are always ready to be sold on the last day of a month. The lead times of express delivery, air freight, and ocean freight are one, two, and three months, respectively. For example, if you place an express delivery order on March 1, the ordered products will get ready on March 31.

There are three types of shipping costs. A variable shipping cost is charged per unit of product. A fixed shipping cost is charged per order. These two costs exist for express delivery and air freight. For ocean freight, there is no variable cost but a container cost, which will be detailed below. The fixed cost of initiating an express delivery order, an air freight, or an ocean freight is \$100, \$80, or \$50, respectively. Table 3 contains variable costs for all products. A container cost is charged per container regarding ocean freight.[1] A container, whose usable capacity is considered as 30 cubic meter (CBM), costs \$2,750.[2] Other costs remain the same. Table 3 contains volume information for all products.

Product	Variable cost (\$)		Volume (CBM)
	Express delivery	Air freight	
1	44	18	0.073
2	89	45	0.005
3	86	38	0.043
4	91	46	0.063
5	50	21	0.045
6	51	25	0.086
7	83	46	0.079
8	96	49	0.082
9	80	35	0.068
10	49	20	0.098

Table 3: Shipping costs and product volumes

As an example, suppose that for each product you order 200 units, where 50 units are shipped by air and 150 units are by sea. The total cost for air freight is

$$\$80 + (\$18 + \$45 + \cdots + \$20) \times 50 = \$17{,}230,$$

which is the sum of the fixed cost per order and variable costs. The total cost for ocean freight is

$$\$50 + \$2,750 \left\lceil \frac{(0.073 + 0.005 + \cdots + 0.098) \times 150}{30} \right\rceil = \$50 + \$2,750 \times \lceil 3.21 \rceil = \$11{,}050.$$

Note that those the fourth container is not fully utilized, the company still needs to pay its full cost. Finally, as nothing is ordered through express delivery, no fixed or variable cost should be paid for this method. The total shipping cost is thus \$17,230 + \$11,050 = \$28,280.

[1] In practice, a container may be of one of two standard containers: 20 feet or 40 feet. To make the problem easier, in this case assignment we assume all containers are of 20 feet.

[2] The true usable capacity of a 20-foot container is 32.6 CBM. To simplify calculation, your company assumes that all ordered products may be put inside a container as long as their total volume does not exceed 30 CBM.

2.1.3 In-transit inventories

The in-transit inventories of all products in the coming months are given in Table 4. For example, for product 6, it is estimated that 18 units will be delivered at the end of March and be available for sales in April, and 23 units will be delivered at the end of April and be available for sales in May. In general, all in-transit inventories are assumed to get ready at the end of a month and should be included in the ending inventory of that month (and thus should be included in calculating the inventory holding cost). Note that because the maximum lead time is three months (if using the ocean freight), we only consider the inventory in-transit for the following two months.

Product	In-transit inventory	
	End of March	End of April
1	0	0
2	48	0
3	0	20
4	153	0
5	0	0
6	18	23
7	28	45
8	0	0
9	109	34
10	0	0

Table 4: In-transit inventories that will be ready at the end of each month

2.1.4 Demand fulfilment with minimum cost

The company's mission is to find the minimum-cost plan that can fulfil all demands on time, where the costs include holding costs and shipping costs.

2.2 An integer program that describes the problem

The IEDO company's ordering problem can be precisely described by an integer program. In this section, we present this formulation. Note that a formulation not only gives us a way to solve a problem but also precisely defines a problem.

First of all, let's use i, j, and t for the indices of products, shipping methods, and periods, respectively. We then define S^I, S^J, and S^T as the sets of products, sets of shipping methods, and sets of periods. In sets S^J, we use 1, 2, etc. to denote shipping methods; similarly, in sets S^T, we use 1, 2, etc. to denote periods. For the instance given in the previous section, we have $S^I = \{1, 2, ..., 10\}$, $S^J = \{1, 2, 3\}$, where 1, 2, and 3 stands for express delivery, air freight, and ocean freight, respectively, and $S^T = \{1, 2, ..., 6\}$, where 1, 2, ..., and 6 stands for March, April, ..., and August, respectively. As the shipping methods that are fast enough for us to receive products shipped through that method at the end of each month is different, we define J_t as the set of methods that should be considered for month $t \in S^T$. In the given instance, we have $J_1 = \{1\}$, $J_2 = \{1, 2\}$, $J_3 = \{1, 2, 3\}$, and $J_t = S^J$ for $t \in \{4, 5, 6\}$.

With the indices and sets defined above, the parameters and variables may then be defined. All of them are listed in Table 5.[3]

Notation	Meaning
D_{it}	Demand of product $i \in S^I$ in month $t \in S^T$
C_i^H	Holding cost per unit per period of product $i \in S^I$
C_i^P	Purchasing cost per unit of product $i \in S^I$
C_{ij}^V	Variable shipping cost per unit of product $i \in S^I$ using method $j \in S^J$
C_j^F	Fixed shipping cost of using method $j \in S^J$
T_j	Shipping lead time of method $j \in S^J$
I_{it}	Number of in-transit product $i \in S^I$ received at the end of month $t \in S^T$
$I_{i,0}$	Initial inventory level of product $i \in S^I$ (ready to be sold in month 1)
V_i	Volume of a unit of product $i \in S^I$
C^C	Cost of a container
V^C	Volume of a container
M	A large number (which can be set to $\sum_{i \in S^I} \sum_{t \in S^T} D_{it}$)
x_{ijt}	Order quantity of product $i \in S^I$ at the beginning of month $t \in S^T$ with method $j \in S^J$
v_{it}	Ending inventory level of product $i \in S^I$ in month $t \in S^T \cup \{0\}$
y_{jt}	1 if shipping method $j \in S^J$ is used at the beginning of month $t \in S^T$ or 0 otherwise
z_t	Number of containers used at the beginning of month $t \in S^T$

Table 5: List of notations used in the integer program

[3]Note that we following the naming rule that (1) all the parameters are in upper case, and all the variables are in lower case, and (2) all the indices are in lower case in subscripts, and superscripts are used to denote variable/parameter names in upper case.

While we are given the example instance, we know we want to define a mathematical program for the *problem*, not for the specific *instance*. This is reflected by the way we define variables and parameters. In the example instance, we have the shipping lead times be $T_1 = 1$, $T_2 = 2$, and $T_3 = 3$; however, they can be other values in other instances. Therefore, to make our formulation general, we still define the parameter T_j. Similarly, though in the example instance only periods 1 and 2 have in-transit inventories, we define it for all periods for generality. In the example instance, all we need to do is to assign 0 to I_{it} for all product $i \in S^I$ and all periods $t \in \{3, 4, 5, 6\}$.

The ordering problem may now be formulated as an integer program as follows. First, the objective function is

$$\min \quad \sum_{i \in S^I} C_i^H \sum_{t \in S^T} v_{it} + \sum_{t \in S^T} \sum_{j \in S^J} \left(\sum_{i \in S^I} (C_{ij}^V + C_i^P) x_{ijt} + C_j^F y_{jt} \right) + \sum_{t \in S^T} C^C z_t, \qquad (1)$$

where the three terms are the holding costs, purchasing cost plus variable shipping cost plus fixed shipping cost, and container costs. Second, there are several functional constraints. For inventory balancing, we have

$$v_{i,t} = v_{i,t-1} - D_{it} + I_{it} + \sum_{j \in J_t} x_{ij,t-T_j+1} \qquad \forall i \in S^I, t \in S^T \qquad (2)$$

$$v_{i,0} = I_{i,0} \qquad \forall i \in S^I, \qquad (3)$$

Note that we use $x_{ij,t-T_j+1}$ rather than x_{ijt} due to the existence of shipping lead time. In the example instance, at the end of period t, we get what we order through express delivery (method 1) at the beginning of period t, what we order through air freight (method 2) at the beginning of period $t-1$, and what we order through ocean freight (method 3) at the beginning of period $t-2$. The sum of the three quantities are therefore $x_{i,1,t} + x_{i,2,t-1} + x_{i,3,t-2}$, which may be aggregated to $\sum_j x_{ij,t-T_j+1}$. Note that for some early periods some shipping methods are impossible to provide replenishment. This is why the summation is only for shipping methods in the set J_t.

In this problem, in each period we need the initial inventory to be enough to cover the demand in that period (because in-transit products and ordered products will be received and ready to be sold at the end of periods). Therefore, we need

$$v_{i,t-1} \geq D_{it} \quad \forall i \in S^I, t \in S^T. \qquad (4)$$

We also need to relate order quantities and whether a shipping method is used by imposing

$$\sum_{i \in S^I} x_{ijt} \leq M y_{jt} \quad \forall j \in S^J, t \in S^T \qquad (5)$$

where M is a very large number that can be set it to be $\sum_{i \in S^I} \sum_{t \in S^T} D_{it}$.[4] We also need a constraint to count the number of containers used for ocean freight, which is

$$z_t \geq \frac{\sum_{i \in S^I} V_i x_{i,3,t}}{V^C} \quad \forall t \in S^T \tag{6}$$

Finally, the nonnegative or binary constraints are

$$\begin{aligned}
& x_{ijt} \geq 0 \quad \forall i \in S^I, j \in S^J, t \in S^T \\
& v_{it} \geq 0 \quad \forall i \in S^I, t \in S^T \\
& y_{jt} \in \{0, 1\} \quad \forall j \in S^J, t \in S^T \\
& z_t \in \mathbb{N} \cup \{0\} \quad \forall t \in S^T.
\end{aligned} \tag{7}$$

An example Python program that invokes Gurobi Optimizer to solves the above integer program is provided as OR113-2_midtermProject_exampleCode.py.

2.3 Solving the example instance

We may utilize the integer program to solve the example instance by invoking a solver like Gurobi Optimizer. In an optimal ordering plan, we do not order anything from June to August, and the ordering plan for March, April, and May are summarized in Tables 6, 7, and 8, respectively. The associated total cost is $16,890,367.02 (accurate to the second digit after the decimal point). Note that maybe in practice the number of products ordered should be integers, but the order quantity x_{ijt} is set to be fractional in the formulation for solvability. Nevertheless, even if integer values are required in practice, our fractional solution may still provide a decision maker a good reference.

3 Your tasks

Below are the tasks for you to complete in this project. In the first two problems, we ask you to do some mathematical programming practice. In the last three problems, we ask you to design your own heuristic algorithm to solve the original problem described in the previous section. Moreover, we will guide you to run numerical experiments to examine the effectiveness and efficiency of your algorithm.

[4]This value of M may not be good. When you implement a program to invoke a solver to solve the example instance (the exact problem or its relaxation), you will be allowed to use other values for M.

Product	Shipping method		
	Express	Air	Ocean
8	0	0	61
10	24	162	200

Table 6: Ordering plan in March

Product	Shipping method		
	Express	Air	Ocean
3	0	0	82
4	0	0	97
5	0	0	67
8	0	0	168
10	0	0	36.449

Table 7: Ordering plan in April

Product	Shipping method		
	Express	Air	Ocean
1	0	0	38
2	0	0	45
3	0	0	200
4	0	0	155
5	0	0	49
6	0	0	154
7	0	0	172
8	0	0	32
9	0	0	94
10	0	117.551	199

Table 8: Ordering plan in May

Problem 1 (20 points)

While in the original problem described in the previous section we need to fulfil all demands on time, in this problem we allow shortage to happen. Unit shortage cost incurs in a similar way as the unit inventory cost. Interestingly, two things may happen when a consumer cannot get the product she/he wants: She/he may make a request and wait until the product is available or leave empty-handed. We say there is a backorder in the former case or a lost sales in the latter case. For this problem, we assume that all shortage become lost sales.

When there is a lost sales, the shortage cost is the difference between the sales price and purchasing cost of the product (for example, the lost sales cost per unit per month of product 1 is \$10,700 − \$5,000 = \$5,700). We now use the following example to show the calculation process of inventory costs (including holding and lost sales costs). From the previous section, we know that the initial inventory for product 3 is 425 units. Suppose that we do not order product 3 in the six months. Table 9 records the detailed steps of calculating all inventory-related costs. In the first month, we have 425 − 79 = 346 units left, which is your stocking level at the end of March. For April, May, and June, the stocking levels are all positive. For July, 82 consumers cannot get product 3, and all 82 consumers leave. For August, 200 more consumers cannot get product 3 and thus will leave. The total holding cost is \$180 × (346 + 167 + 166 + 117) = \$143,280, and total lost sales cost is \$17,900 $\times$ (82 + 200) = \$5,047,800. The total inventory-related costs is \$5,191,080.

	March	April	May	June	July	August
Demand	79	179	21	49	199	200
Inventory in-transit	0	0	20	0	0	0
Ending inventory	346	167	166	117	−82	−200
Stocking level	346	167	166	117	0	0
Lost sales	0	0	0	0	82	200

Table 9: Lost sales example

Please formulate a mixed integer linear program that finds an optimal ordering plan. As shortage is allowed now, your goal is to minimize total cost by making trade-off between all the related costs. In your report, write down your mathematical models using rigorous mathematical notations and expressions. If you use the notations defined in this document, you do not need to redefine it again in your report; however, you need to define your own variables or parameters before using them. Similarly, if you believe any constraint may be completely reused in your model, you may simply refer the equation number rather than rewrite it again.

Note that in the formulation provided in the previous section, the value of M may not be good. When you implement your program to invoke a solver to solve the example instance, you are allowed to use other values for M. You are even allowed to make it M_{jt} and assign different values for different methods and periods. If you do so, you need to clearly write down the way you assign values to M (or M_{jt}).

With your formulation, solve the instance given in this document using a solver to come up with an optimal ordering plan for the next coming six months. Do not forget to present your ordering plan in a business language.

Problem 2 (20 points)

Continue from the previous problem, we now consider the possibility of backorder. In other words, when a consumer cannot buy a product, a consumer may make a request and wait until the product is available (which is the case of backorder) or leave empty-handed (which is the case of lost sales).

When there is a backorder, a customer gets 5% of the sales price as a discount for each month she/he waits. The shortage cost of having one backorder is thus 5% of the sales price. According to historical records and experience, the backorder percentage of each product is estimated. The backorder costs per unit per month and backorder percentages of all products are listed in Table 10. For example, as the company may sell product 2 at the price of \$5,000, giving out 5% of discount due to let one customer wait for one month results in the shortage cost \$5,000 $\times$ 0.05 = \$250 per unit per month. Having one unit of lost sales, however, is a profit loss of \$5,000 − \$2,000 = \$3,000 as the lost sales cost. Luckily, 70% of all consumers will choose to wait when product 2 is out of stock.

Product	Costs of backorder (\$)	Backorder percentage
1	535	0
2	250	0.7
3	1345	0.1
4	980	1
5	345	1
6	1575	0.3
7	705	0.6
8	820	0.2
9	1650	0.1
10	930	0.5

Table 10: Backorder informtaion

As an example, suppose that we own 400 units of product 5 at the beginning of March, and we order nothing for product 5 throughout the six months. It then follows that the ending inventory for product 5 will be 365, 303, 220, 130, −67, and −116 for each of the following six months, where a "negative inventory level" means shortage. Note that for product 5 all consumers will wait upon shortage. The total inventory-related cost is thus

$$\$40 \times (365 + 303 + 220 + 130) + \$345 \times (67 + 116) = \$40{,}720 + \$63{,}135 = \$103{,}855.$$

As another example, consider product 3, whose initial inventory level is 425 units. Suppose that we still do not order product 3. Table 11 records the detailed steps of calculating all inventory-related costs. In the first month, you have 425 − 79 = 346 units left, which is your stocking level at the end of March. For April, May, and June, the stocking levels are all positive. For July, 82 consumers cannot get product 3, and 10% of them (8.2) prefer to wait while 90% of them (73.8) prefer to leave. For August, as 200 more consumers cannot get product 3, in total there are 8.2 + 200 = 208.2 consumers who wish to (but cannot) get product 3 in August. Note that this number should be 8.2 + 200, not 82 + 200, because 73.8 consumers leave at the end of July. Now, at the end of August, $208.2 \times 0.1 = 20.82$ consumers are still waiting, and $208.2 \times 0.9 = 187.38$ consumers leave. The total holding cost is \$180 $\times$ (346 + 167 + 116 + 117) = \$134,280, total backorder cost is \$1,345 $\times$ (8.2 + 20.82) = \$39,031.9, and total lost sales cost is \$17,900 $\times$ (73.8 + 187.38) = \$4,675,122. The total inventory-related costs is \$4,848,433.9.[5]

	March	April	May	June	July	August
Demand	79	179	21	49	199	200
Inventory in-transit	0	0	20	0	0	0
Ending inventory	346	167	166	117	−82	−208.2
Stocking level	346	167	166	117	0	0
Backorder	0	0	0	0	8.2	20.82
Lost sales	0	0	0	0	73.8	187.38

Table 11: Backorder example

Please formulate a mixed integer linear program that finds an optimal ordering plan that considers both backorder and lost sales and minimizes the total cost. In your report, write down your mathematical models using rigorous mathematical notations and expressions. Similar to Problem 1, you may reuse the notations and constraints defined in this document, and you may define your own way to assign values to M or M_{jt}.

With your formulation, solve the instance given in this document using a solver to come up with an optimal ordering plan for the next coming six months. Do not forget to present your ordering plan in a business language.

Problem 3 (20 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 many freight carriers that offer you more than ten shipping methods with different speeds and costs. It is thus possible that an MILP solver like Gurobi Optimizer cannot give you an optimal solution in a reasonable amount of time. You then need a heuristic algorithm.

For the original ordering problem described in Section 2 (not the extended problems described in Problems 1 and 2), please design and implement a heuristic

[5] You may wonder how a "number of consumers" may be fractional. This is of course to simply the problem and calculation. After all, even if one insists to round these fractional values, the resulting costs will not be too far from those calculated with no rounding.

algorithm to obtain a near-optimal solution by using a short amount of time. You will demonstrate the effectiveness and efficiency of your algorithm in the next two problems. In this problem, please describe the design of your algorithm, which ideally should make sense, i.e., doing something that intuitively should be useful. You may draw a flowchart, write text description, write pseudocode, provide numeric examples, or using any way you like to introduce your algorithm. There should also be a basic big-O analysis regarding the time complexity of your algorithm. The goal is to make people understand your algorithm and be convinced that it will find near-optimal solutions in a reasonable amount of time.

After describing your algorithm and analyzing the time complexity, use your algorithm to solve the example instance. Present the ordering plan reported by your heuristic algorithm in a business language and the associated objective value. Compare that with the optimal solution provided in Section 2 to claim that your algorithm finds a good solution at least for the example instance.

As we ask you to submit your computer programs, there is no need to copy and paste the program you implement into your report. Nevertheless, in most cases the grading for this problem will be solely based on your report. In particular, we focus on how clear your presentation is and how convincing and intuitive your solution approach is. Please try your best to convince the instructing team that your algorithm is intuitively good.

Problem 4 (20 points)

Recall that we designed and implemented a heuristic algorithm in the previous problem. Even if we tried to solve the example instance to test the performance of our heuristic algorithm, how may we be confident in the performance for solving remaining instances? Only one instance is not enough!

To make ourselves more confident, we may want to conduct some numerical experiments before we use our heuristic algorithm in everyday operations (to the instructing team or to an organization that will use our heuristic algorithm in practice). To do so, we write another program to generate random instances, let our algorithm solves these random instances, and compare the objective values generated by our algorithm with some benchmark. We will separate this task into two parts, one in this problem and one in the next problem. In this problem we prepare the experiment architecture, including random instances and benchmarks. In the next problem we examine the performance of our heuristic algorithms.

Constructing an experiment. A good experiment consists of *factors*, *levels*, *scenarios*, and *instances*. A factor is typically a parameter that we will manipulate its value in the experiment. Sometimes a factor has multiple parameters involved. For each factor, we design a few levels, where each level corresponds to a parameter value (if there is only one parameter in that factor). Each scenario is then a combination of factor levels. Lastly, in each scenario we randomly generate multiple instances to be input into our heuristic algorithms and benchmarks.

As an example, for the original ordering problem, we may consider the following three factors, each with three levels:

1. The scale of an instance, which includes the number of products and number of periods. The three levels are "small" with ten products and six periods, "medium" with one hundred products and twenty periods, and "large" with five hundred products and fifty periods.
2. The container cost. The three levels are "low" with \$1,375 per container, "medium" with \$2,750 per container, and "high" with \$5,500 per container.
3. The inventory holding cost. The three levels are "low" with 1% of purchasing cost, "medium" with 2% of purchasing cost, and "high" with 4% of purchasing cost.

With these three factors and their levels, there are two typical ways to create scenarios. In the first way, people create $3 \times 3 \times 3 = 27$ scenarios to include all combinations of factor levels. In this way, scenario 1 stands for small instances, low container costs, and low holding costs, scenario 2 stands for small instances, low container costs, and medium holding costs, ..., and scenario 27 stands for large instances, high container costs, and high holding costs. In the second way, we first combine all the middle level of all factors to form a basic scenario, and then in each extended scenario we adjust the level of one and exactly one factor. In total we will have $1 + 2 \times 3 = 7$ scenarios as shown in Table 12, where scenario 1 is the basic scenario and scenarios 2 to 7 are the extended scenarios. For example, in scenario 5, we will have one hundred products and twenty periods, \$5,500 per container, and 2% of purchasing cost as the holding cost.

In each scenario, we generate several, say, ten or thirty, instances. To do so, we need to assign values to those parameters that are not included in the factors. For some of these parameters, we fix them values throughout all instances in the whole experiment; for some others, we give each of them a probability distribution so that in each instance the parameter value is drawn from the distribution. Take our ordering problem as an example, Table 13 summarizes a set of ways to assign values to the all parameters:

- For the number of shipping methods, container volume, fixed shipping costs, and

Scenario ID	Levels		
	Scale	Container cost	Holding cost
1	medium	medium	medium
2	small	medium	medium
3	large	medium	medium
4	medium	low	medium
5	medium	high	medium
6	medium	medium	low
7	medium	medium	high

Table 12: Seven scenarios in the experiment

shipping lead times, we fix their values to the values specified in the case across all scenarios.

- For demands, purchasing costs, variable shipping costs, initial inventory levels, in-transit inventor levels, and product volumes, we give each of them a probability distribution. Note that not all parameters are drawn from a simple distribution. In particular, to ensure that the variable cost of shipping a product through air freight is lower than that through express delivery, for each product we first draw a value to be its express variable shipping cost and then draw a proportion to multiply to the express variable shipping cost to obtain the air variable shipping cost. The relationship between the first-period demand and initial inventory is similar. For in-transit inventory levels, to reflect the fact that many product do not have in-transit quantities (because they were not ordered in the previous months), with only 50% of probability we draw a value from the uniform distribution between 0 and 50. In other words, there is a 50% probability the in-transit quantity will be zero.

- For the number of products, number of periods, holding costs, container costs, the way to set their values differ from scenarios to scenarios.

All the above settings allow you to generate many random instances to cover different scenarios. For example, if we use the seven scenarios defined above and generate ten instances per scenario, we will generate seventy instances. The experiment then is much better than one with only five instances. It is also better than an experiment with 1000 instances that all belong to the same scenarios.

Parameter	Way to assign values
$\lvert S^J \rvert$	3
V^C	The value specified in the case
C_j^F	The values specified in the case
T_j	The values specified in the case
D_{it}	Uniform between 0 and 200
C_i^P	Uniform between \$1,000 and \$10,000
$C_{i,1}^V$	Uniform between \$40 and \$100
$C_{i,2}^V$	$\alpha_i C_{i,1}^V$, where α_i is uniform between 0.4 and 0.6
$I_{i,2}$	50%: 0; 50%: uniform between 0 and 50
$I_{i,1}$	50%: 0; 50%: uniform between 0 and 200
$I_{i,0}$	Uniform between $D_{1,t}$ and 400
V_i	Uniform between 0 and 1
$\lvert S^I \rvert$	According to the chosen level in that scenario
$\lvert S^T \rvert$	According to the chosen level in that scenario
C_i^H	According to the chosen level in that scenario
C^C	According to the chosen level in that scenario

Table 13: Ways to assign values to parameters

Benchmarks. Regarding benchmarks, one typical candidate is a mathematical program. As long as your instances are not too big, you may use a solver to generate an optimal solution and compare your heuristic solution with the optimal one. When your instances are big, another typical candidate is the relaxation of your mathematical program. You may take away the integer constraint or some other constraints to find an upper bound or lower bound of the objective value of an optimal solution.

It is also common for one to compare the proposed algorithm with some simple (or even stupid) ones. After all, if the naive algorithm can also perform well, the original problem is not difficult (and thus the proposed algorithm is not valuable). For example, a simple heuristic is the following. For each given instance, we consider only express delivery. Moreover, we do not keep any inventory; we always order exactly in a month what we need at the beginning in each month. In other words, after we use all the initial inventory and in-transit inventory to fulfil demands, afterwards we will not have any inventory at the end of each period.

Each instance will be solved by a mathematical program solver (and maybe it is the relaxation that is solved), the proposed algorithm, and the very naive heuristic. Ideally, from the perspective of objective value, the proposed algorithm should be close to the mathematical program solver and far from the very naive heuristic; from the perspective of computation time, the proposed algorithm should be close to the very naive heuristic and far from the mathematical program solver. Numerical experiments will tell us whether this is true (and whether we should try to improve our proposed algorithm).

A complete example. To get a more concrete idea about how to conduct experiments to analyze the performance of your heuristic algorithm, you are suggested to take a look at the paper "fairAllocation_230822.pdf" provided to you with this document. You may go through the first paragraph of Section 3 very quickly to understand the problem, and then jump to Sections 5.1, 5.2, and 5.4 to see an example about designing factors and scenarios, generating instances, designing benchmarks, running experiments, and describing all of these. There are something that you do not need to do:

- While in Section 5.2 of that paper you see a paragraph describing the genetic algorithm, you are not required to do this. When benchmarking your heuristic algorithm, a very simple algorithm is enough.
- While in Section 5.3 of that paper you see another version of the problem, you do not need to have multiple versions. Then of course you do not need to worry about the performance of your heuristic algorithm in such an another version, even though this is done in Section 5.4 of that paper.
- While in Section 5.5 of that paper you see some analysis regarding computation time, you are not required to do this.

What you need to do in this problem. In this problem, please follow the above instructions to conduct your own experiments. While we will leave the examination of your proposed algorithm (the one introduced in Problem 3) in the next problem, in this problem we ask you to generate scenarios, generate instances, implement a mathematical solver (actually you should have done so pretty much in Problems 1 and 2), implement the very naive heuristic, and run the experiment to find the optimality gap for the very naive heuristic. You must follow the above instructions to (1) set up factors, levels, and the seven scenarios; (2) generate thirty instances for each scenario; (3) implement a program that invokes a solver to solve the linear relaxation of the integer program; and (4) implement the very naive heuristic. You may then input the 210 instances to the program that invokes a solver and the very naive heuristics and then obtain 210 optimality gaps (in percentages); you may then present the average and standard deviation of the thirty optimality gaps for each scenario; you may then summarize and interpret your findings.

Note that in the formulation provided in the previous section, the value of M may not be good. When you implement your program to invoke a solver to solve the relaxation of instances, you are allowed to define your own way to assign values to M (or M_{jt}) as long as you clearly write it down. Please be careful in doing this, as different values of M (or M_{jt}) may result in different outcomes when solving the relaxed programs.

The result should somewhat make sense. For example, you know the very naive heuristic does not consider ocean freight and inventory. Therefore, its optimality gap should be relatively higher when ocean freight and inventory is cheap. If you do not see that at the end, something should be wrong. In other words, experiments can also help you check whether you are doing everything correctly.

To make grading easier, do not try to modify the experiment setting. You will design and conduct your own experiment in the final project.

Problem 5 (20 points)

The last step is to add your proposed algorithm (designed in Problem 3) into the experiment (constructed in Problem 4). Then report the average and standard deviation of the thirty optimality gaps for each scenario; you may then summarize and interpret your findings. Beside objective values, please also report the computation time of solving the relaxation, running your proposed algorithm, and running the very naive heuristic. You may want to convince people that your algorithm is efficient.

Please keep this in mind: Designing an algorithm is not hard; making it convincing is. Please do everything you may do to make your proposed algorithm convincing.

Note. Though mentioned above, here we remind you again: Problems 3 to 5 should be done with respect to the basic ordering problem described in Section 2, not the extended problems described in Problems 1 and 2.