QBUS2310 Management Science
Assignment 2

Semester 1, 2022

Out: 2nd May 2022

Due: 26th May 2022 at 11:59pm

Instructions

- This assignment consists of five problems, some involving multiple parts. Some parts require a written response and others involve coding. The parts that require a written response are described in this document, while the coding questions are described in the associated Jupyter notebook (`.ipynb` file).
- When a problem asks you to **formulate** a model, you need to provide your mathematical formulation with clear justification of variables, constraints and objective. If you decide to label any of the data with algebraic symbols, you must clearly define these (e.g., let a_{ij} be the amount of material i required by product j).
- The written parts have to be typed up. This means no handwriting and no screen shots. If you are using MS Word, use the equation editor to make your mathematics look pretty. We recommend using LaTeX or a similar system for typesetting your answer.
- You should submit a PDF to GradeScope for the written parts and **match the page number with the questions that you answered**. You can find the detailed instructions on how to scan and submit your assignments through GradeScope on Canvas. If you fail to match the page to the corresponding question, the marker will not be able to view your response and thus you will be awarded 0 marks for the question.
- You should answer the coding questions by modifying the Jupyter notebook appropriately, and submit it through Canvas.
- All the problems can be done using only the material from this class, and we will deduct points from solutions that refer to outside material.

Question:	1	2	3	4	5	Total
Points:	15	0	25	20	25	85

1. *Tournament elimination via integer programming.* Suppose we have n teams competing in a tournament. The teams will play every other team some number of times, and each outcome will be win-lose (no draws). The winner is the team that has the highest number of overall wins; for simplicity we will allow for more than one winner in the event of ties, and each game must end in win or loss (no draws). At certain points in the tournament, we wish to determine which teams are eliminated, meaning they have no chance of being the winner. The data available at a given point is the following:

   - w_i, the number of wins so far for each team i.
   - r_{ij}, the number of games remaining to be played between teams i and j. Note that these are fixed numbers, not decision variables.

   We say that team k is eliminated if we can guarantee that there is at least one other team that will finish with a better record than them, given that they win all of their remaining games. In other words, even if team k finished with the best possible $w_k + \sum_j r_{kj}$ wins given the remaining schedule, there will always be at least one other team that has strictly more wins than k, regardless of how other results play out.

   An easy condition to check for elimination is that there exists $w_i > w_k + \sum_j r_{kj}$. However, this is not the only condition. For instance, if k has 22 wins and no games left to play, but there are two teams i and j with 21 wins who must play each other three times, then at least one of i and j will finish with 23 or more wins (regardless of the results, since one team must win at least 2 games), eliminating k.

   The subtlety is that eliminating k depends on more than just the head-to-head record between k and other teams. It also depends on how other teams perform against each other. There are numerous combinations for how other results will play out, and checking each one manually is a daunting task. However, by organising this in an integer programming model, and then a network flow model, we can solve this using existing machinery.

   In this question, we will formulate an *integer programming model* for determining whether a given team k is eliminated or not.

   Let z_i be the number of wins for team i from remaining games, assuming that team k has won all of the remaining games. Team k is eliminated if, for all possible outcomes, $\max_{i \neq k}\{z_i + w_i\} > w_k + \sum_j r_{kj}$. We wish to describe all feasible outcomes z_i for $i \neq k$ using linear constraints. If we can do this, then we can check whether k is eliminated simply by minimizing the maximum over all feasible outcomes.

   We will define $x_{ij,i}$ to be the number of times that i wins over j in their remaining games. We will use the notational convention that $x_{ij,i} = x_{ji,i}$ and $x_{ij,j} = x_{ji,j}$.

   (a) (3 points) Note that $x_{ij,i}$ should be an integer variable. What are bounds on $x_{ij,i}$?

   (b) (3 points) Relate $x_{ij,i}$ and $x_{ij,j}$ with a linear constraint (involving r_{ij}).

   (c) (3 points) Relate z_i and $x_{ij,i}$ with a linear constraint.

   (d) (6 points) Formulate an optimization model that can be used to determine whether k is eliminated or not, and explain how we would use it.

2. *Tournament elimination via network flows.* Consider the same tournament elimination problem as question 1, where we built a linear program to check whether a given team k is eliminated or not. Solving an integer program for this is doable, but is unnecessary for this problem. We will use the important property of network flow models $\min_x \left\{c^\top x : Ax = b,\ 0 \le x \le u\right\}$ that when using integer data b, u, the optimal flow solutions will be integer as well, where A is the node-arc incidence matrix of some network.

   (a) (2 points (bonus)) Consider a bipartite network where nodes on the left side represent pairs of teams with games remaining, and the right side represents teams. Explain how variables $x_{ij,i}$ can represent arcs in this network.

   (b) (2 points (bonus)) How can variables z_i be incorporated into the network?

   (c) (2 points (bonus)) We need to represent the constraint relating $x_{ij,i}$ and $x_{ij,j}$ in our network. How can we do this? (Think about exploiting flow balance constraints.)

   (d) (2 points (bonus)) Similarly, explain how to represent the constraint relating z_i and $x_{ij,i}$.

   (e) (3 points (bonus)) Modelling $\max_{i \neq k}\{z_i + w_i\}$ is not possible with a network model. But to check that k is not eliminated, we don't need to model the maximum fully, we just need to check that it's less than $w_k + \sum_j r_{kj}$, which is the same as $z_i + w_i \le w_k + \sum_j r_{kj}$ for each $i \neq k$. Explain how this can be incorporated into the network model.

   (f) (4 points (bonus)) Define missing components of the network model (if any) and explain how solving it determines whether k is eliminated or not.

3. (25 points) **Please see the Jupyter notebook for further details.** Implement the tournament elimination model in Gurobi by completing the given functions in the Jupyter notebook.

4. *Team formation.* In this problem, we will consider a *team formation problem* where we need to assign agents to different teams in order to maximize utility.

   Suppose that you are a unit coordinator running a unit with a group work assessment. In this class, you have n students and need to form T different teams of size t and for simplicity, assume that $n = Tt$. We can express the friendship status between the students through an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} = [n]$ and $\{(i,j),(j,i)\} \in \mathcal{E}$ if student i is friends with student j. Additionally, define U to be a symmetric $n \times n$ matrix of utilities, where U_{ij} denotes the utility when student i is in a group with student j (which may be negative as there may be conflict between students) and assume that U has 0s on the diagonal for simplicity.

   Note. In the rest of the problem, please ensure that your final answer can be added to an IP to solve the problem, i.e., all functions are linearly representable.

   (a) (4 points) Define x_{ik} to be whether student i is in group k. What constraints do we have on this variable?

   (b) (4 points) Let $y_{ijk} = x_{ik}x_{jk}$ be a binary variable which is on if both students i and j are in team k. Clearly, we also have $y_{ijk} = y_{jik}$. What linear constraints can we add to model this?

   To promote diversity, you plan to form teams such that each team has at most m different friendship pairs. How can you model this as a IP?

   (c) (4 points) Now, to further promote diversity, you plan to form teams such that each student knows at most p students in their group beforehand. Using y_{ijk}, what linear constraint allows us to model this?

   (d) (4 points) The utility for each group is defined to be the sum of all the pairwise utilities between all the students in each team, which has an upper bound of $U_{\max}$ for each team. (You can think of this being the maximum mark attainable.)

   Suppose you want to maximize the lowest utility over all teams. How can you model this as a IP?

   (e) (4 points) Now suppose that each team is required to perform a different task. For example, a production where one team is responsible for lighting, one for sound, etc. Now you have T different utility matrices, denoted $U^{(k)}$, $k \in [T]$, where $U_{ij}^{(k)}$ is the pairwise utility of having students i and j perform task k. Assume that the total utility of a group is calculated in the same way as described in part (d) where the total utility for each task is capped at $U_{\max}$.

   Suppose you now want to maximize the total utility across all tasks. How would you model this?

5. *Chance constraints and the big-M technique.* We consider a stochastic variant of the production planning problem.

   Recall that we have n products indexed by $j \in [n]$ and m materials indexed by $i \in [m]$. We decide amounts x_j for each product j to produce. Each unit of product j requires a_{ij} units of material i. We define $a_i = (a_{i1}, \ldots, a_{in}) \in \mathbb{R}^n$ for each $i \in [m]$, and $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$. We obviously require $x \geq 0$. We will consider the simple case without holding costs, and assume that any amount x_j of product j that we produce can be sold for $c_j x_j$ profit. Thus letting $c = (c_1, \ldots, c_n)$, the total profit of our plan is $c^\top x$. Ordinarily, we know the amount b_i of material i we have available, and choose x so that $a_i^\top x \leq b_i$ for each $i \in [m]$. Thus we would ordinarily solve

$$\begin{aligned} \max_x \quad & c^\top x \\ \text{s.t.} \quad & x \geq 0 \\ & a_i^\top x \leq b_i, \quad i \in [m]. \end{aligned}$$

   In the stochastic variant, however, the vector $b = (b_1, \ldots, b_m) \in \mathbb{R}^m$ is *not* known exactly. Instead, we are given N possible scenarios for the b vector, which we label $b^1, \ldots, b^N$. Note that b_i^k is the amount of material $i \in [m]$ available in scenario $k \in [N]$. We assume that each scenario has equal probability $1/N$ of occurring.

   In *chance-constrained programming*, we use this data in the following way. Let $\tilde{b} \in \mathbb{R}^m$ be the random vector that equals b^k with probability $1/N$, for each $k \in [N]$. Fix some $p \in [N]$. We will require that our production plan x satisfy the *chance constraint*

$$\mathbb{P}\left[a_i^\top x \leq \tilde{b}_i, \ \forall i \in [m]\right] \geq \frac{p}{N},$$

i.e., we wish to choose the production plan x so that the production constraints $Ax \leq b^k$ are satisfied for at least p out of N scenarios b^k. Thus we solve the following model:

$$\begin{aligned} \max_{x} \quad & c^\top x \\ \text{s.t.} \quad & x \geq 0 \\ & \mathbb{P}\left[a_i^\top x \leq \tilde{b}_i,\ \forall i \in [m]\right] \geq \frac{p}{N}. \end{aligned}$$

Note: this model does not specify *which* of the scenarios should be satisfied; instead, we let the optimization model make this choice for us, in a way that maximizes the profit $c^\top x$.

In parts (a) and (b), we will derive a linear formulation for the chance constraint $\mathbb{P}\left[a_i^\top x \leq \tilde{b}_i,\ \forall i \in [m]\right] \geq p/N$. Then in part (c) and (d), we will implement it in Gurobi.

(a) (4 points) For each scenario k, introduce a binary variable $z_k \in \{0,1\}$. We can use the big-M technique to write a set of constraints that are equivalent to the following implication:

$$z_k = 0 \implies Ax \leq b^k.$$

In other words, you want to ensure that when z_k is set to 0, we will enforce each of the constraints $a_i^\top x \leq b_i^k$ for each $i \in [m]$. This can be done by adding the linear constraints

$$a_i^\top x \leq b_i^k + M_i^k z_k \quad \forall i \in [m],$$

for some large enough constants M_i^k.
Assuming that $a_{ij} > 0$ and $b_i^k \geq 0$ for all $i \in [m],\ j \in [n]$, explain why

$$M_i^k = \max_{k' \in [N]} b_i^{k'} - b_i^k.$$

are valid big-M values.

(b) (2 points) Together with the constraints from part (a), give a constraint on the z vector that is required to ensure the chance constraint $\mathbb{P}\left[a_i^\top x \leq \tilde{b}_i,\ \forall i \in [m]\right] \geq p/N$ holds.

(c) (6 points) **Please see the Jupyter notebook for further details.** Implement the chance constrained production planning model in Gurobi by completing the given functions in the Jupyter notebook.

(d) (2 points) On the given data in the Jupyter notebook, run the model you implemented in part (c) and report the runtime. You might want to run the code a couple of times and average the times. You do not need to submit the code for this, just report the runtime.

If we can find smaller but still valid big-M constants, then the runtime of the model will improve. Parts (e) to (g) will investigate a method to do this, and compare the runtime with part (d).
To this end, for each $i \in [m]$, define the set

$$Q_i = \left\{ (x,z) \in \mathbb{R}^n \times \mathbb{R}^N : \begin{array}{c} z_k \in \{0,1\},\ k \in [N] \\ a_i^\top x \leq b_i^k + M_i^k z_k,\ k \in [N] \\ \sum_{k \in [N]} z_k \leq N - p \end{array} \right\}.$$

In other words, $(x,z) \in Q$ whenever each entry of z is binary, the sum of the entries of z is at most $N-p$, and whenever $z_k = 0$, x satisfies $a_i^\top x \leq b_i^k$.
The Q_i are related to our model in the following way: x satisfies $\mathbb{P}[Ax \leq \tilde{b}] \geq p$ if and only if

$$\text{there exists a binary vector } z \in \{0,1\}^N \text{ such that } (x,z) \in \bigcap_{i \in [m]} Q_i.$$

You should convince yourself that this is true (but you don't need to submit anything for it).

(e) (4 points) Fix some $i \in [m]$. Define $\bar{b}_i^{(p)}$ to be the pth largest value amongst $b_i^1, \ldots, b_i^N$. Show that if $(x,z) \in Q_i$, then it satisfies the constraint

$$a_i^\top x \leq b_i^k + \left(\bar{b}_i^{(p)} - b_i^k\right) z_k, \quad \forall k \in [N].$$

Hint: to do this, fix a point $(x, z) \in Q_i$ and $k \in [K]$. Show that when $z_k = 0$, then $a_i^\top x \leq b_i^k$ is satisfied. Then show that when $z_k = 1$, $a_i^\top x \leq \bar{b}_i^{(p)}$ is satisfied.
This shows that we can replace the old big-M values M_i^k from part (a) with new big-M values $\bar{M}_i^k = \bar{b}_i^{(p)} - b_i^k$.

(f) (3 points) **Please see the Jupyter notebook for further details.** Implement the chance constrained production planning model in Gurobi by completing the functions in the Jupyter notebook. You should implement your model with the new big-M values $\bar{M}_i^k = \bar{b}_i^{(p)} - b_i^k$.

(g) (4 points) On the given data in the Jupyter notebook, run the model you implemented in part (f) and report the runtime. You might want to run the code a couple of times and average the times. You do not need to submit the code for this, just report the runtime. Comment on the differences between the runtimes compared to the differences between the big-M values. What is the “extra bit of information” that we used to derive the new values $\bar{M}_i^k$ in part (e) compared to the original values M_i^k in part (a)?