MIE 1603 / 1653 - Integer Programming
Winter 2019
Assignment #5

Due Date: April 4th, Thursday, no later than 9:00AM.

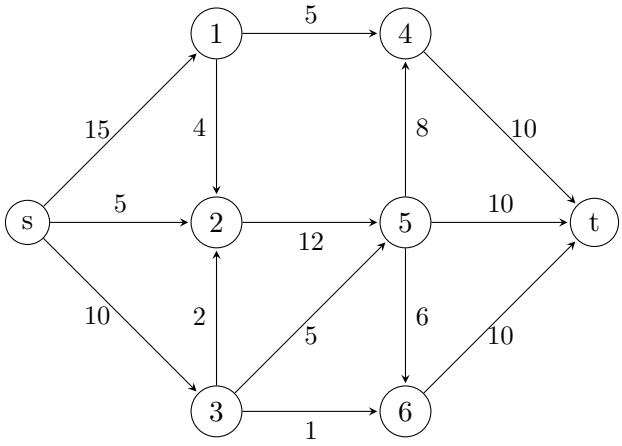
Solve the following problems. You may work in groups of *two* students (from the same course code) or individually. Turn in one solution set with both group members listed on it. Groups must work independently of each other. You *must* cite any references (texts, papers or websites) you have used to help you solve these problems.

You can submit your written solutions using Quercus (pdf file) or give a hard-copy to the professor or TA in their office hours (or lectures). Please submit your code (e.g., a python file) via Quercus (i.e., do not provide a print out of your code, nor include it in the pdf). If you are using Jupyter Notebook, please submit an exported .py file, rather than an .ipynb file.

1. [15pts] Consider the *production requirement* problem. Consider a set of factories $M = \{1, ...m\}$ and a set of costumers $N = \{1, ..., n\}$. Each factory $j \in M$ can produce at most p_j units of a product and each costumer $i \in N$ requires r_i units of such product (all factories/costumers produce/require the same product). Notice that a factory can sell products to multiple costumers and each costumer can receive products from one or more factories. Due to traveling limitations, each factory $j \in M$ can sell products to a subset of consumers $N_j \subseteq N$. The problem asks if it is possible to satisfy the requirements of all costumers. If the answer is affirmative, the problem asks the amount of products each factory sold to each costumer.

   Show how this problem can be formulated and solved as a *maximum flow* problem. Your solution should specify the data for the maximum flow problem you would need to solve (e.g., graph and edge capacities) and how the resulting solution would be mapped to a solution for the production requirement problem.

2. [15pts] Consider the following directed graph with a given maximum capacity for each arc.



   (a) Calculate the maximum flow from node s to t.

(b) Use strong duality to show that the flow in part (a) is optimal.

3. [15pts] Consider the following 0-1 knapsack problem.

$$\begin{aligned} \max\ & 3x_1 + 6x_2 + 4x_3 + x_4 + 7x_5 + 2x_6 + 5x_7 + x_8 \\ s.t.\ & 25x_1 + 60x_2 + 43x_3 + 15x_4 + 80x_5 + 30x_6 + 45x_7 + 22x_8 \leq 100 \\ & x_i \in \{0,1\} \qquad \forall i \in \{1,...,8\} \end{aligned}$$

Prove that the following constraints are valid for the problem:

$$x_1 + x_3 + x_4 + x_6 + x_8 \leq 4 \tag{1}$$
$$x_3 + x_7 + x_8 \leq 2 \tag{2}$$
$$x_2 + x_3 + x_5 \leq 1 \tag{3}$$

4. [55pts] Consider the following formulation of the multi-period production planning problem with I products and T time periods.

$$\min \sum_{i=1}^{I}\sum_{t=1}^{T} h_i s_{i,t} + \sum_{i=1}^{I}\sum_{t=1}^{T} f_i y_{i,t}$$
$$s.t.\ s_{i,t-1} + x_{i,t} = D_{i,t} + s_{i,t} \qquad \forall i \in \{1,\dots,I\}, t \in \{1,\dots,T\} \tag{4}$$
$$s_{i,0} = SI_i \qquad \forall i \in \{1,\dots,I\} \tag{5}$$
$$s_{i,T} = SF_i \qquad \forall i \in \{1,\dots,I\} \tag{6}$$
$$x_{i,t} \leq M_{i,t} \cdot y_{i,t} \qquad \forall i \in \{1,\dots,I\}, t \in \{1,\dots,T\} \tag{7}$$
$$\sum_{i=1}^{I} \alpha_{k,i} \cdot x_{i,t} \leq L_K \qquad \forall t \in \{1,\dots,T\}, k \in \{1,2\} \tag{8}$$
$$x_{i,t}, s_{i,t} \geq 0,\ y_{i,t} \in \{0,1\} \qquad \forall i \in \{1,\dots,I\}, t \in \{1,\dots,T\}$$

For each product $i \in \{1,\dots,I\}$ and time period $t \in \{1,\dots,T\}$, variable $x_{i,t}$ represents the amount of product i produced in period t, $y_{i,t}$ indicates if any amount of product i was produced in period t, and $s_{i,t}$ represents the inventory level of product i at time t.

The data for this problem are the inventory cost h_i, the setup cost f_i, the initial inventory SI_i and the ending inventory SF_i for each product $i \in \{1,\dots,I\}$. The problem also considers a demand $D_{i,t}$ for each product i and time period t, the capacity L_k for each machine $k = 1, 2$ and the required usage $\alpha_{i,k}$ for each unit of product i on machine k.

Constraint (4) specifies the inventory balance conditions. Constraints (5) and (6) specify the initial and final inventory levels, respectively. Constraint (7) corresponds to the production setup, where we use $M_{i,t} = \sum_{k=t}^{T} D_{i,k} + SF_i$ as our big-M value. Lastly, constraint (8) correspond to the capacity constraint of each machine.

Notice that for a single product, this problem is equivalent to the lot-sizing problem introduced in Chapter 7. In this question you will implement the (ℓ, S) inequalities seen in lecture (Chapter 7) in both a cut-and-branch and branch-and-cut algorithm, and compare your results

to original MIP (i.e., without the (ℓ, S) inequalities). In order to better illustrate the effect of these cuts you need to turn off the Gurobi automatic cuts, `m.setParam('Cuts', 0)` in all three algorithms.

- The cut-and-branch can be implemented by initially not declaring the binary decision variables to be binary, and then using a loop that alternates between solving the LP and adding additional violated cuts. Once no more violated cuts are found you declare the binary variables to be binary and solve the problem one more time.
- The branch-and-cut will need to be implemented within a callback routine. Since we are only adding valid inequalities, we need to use User Cuts (i.e, no need to use Lazy Cuts).

A skeleton code, `HW5_skeleton.py` and a single data instance `HW5_data.tx` is provided on the course website. The skeleton includes the MIP formulation and only requires you to implement the callback function for the branch-and-cut, the loop for the cut-and-branch and the (ℓ, S) cut generation function. If you choose to use the skeleton code, feel free to make any changes.

Test the three options (i.e., the original MIP formulation, cut-and-branch, and branch-and-cut) on the same data instance `HW5_data.tx`. You should report the total solution time, the root relaxation objective value, and the total number of branch-and-bound nodes with and without the (ℓ, S) inequalities. Use a time limit of 5 minutes. If it does not solve to optimality in the time limit, also report the best lower and upper bounds together with the optimality gap at the time limit.

You should turn in a **summary of the results** of the experiments above. **Significant credit will be taken off for submissions that only include printouts of the code running**. Submit your code **and** the outputs generated during the solution processes to Quercus. If you are using Jupyter Notebook, submit an exported .py file, rather than an .ipynb file.