Homework Assignment 1 Fall 2022

IE 532 Analysis of Network Data Due on Friday 09/23 (at 11:59 pm)

There are 4 questions in this assignment.

Please answer every question to the best of your knowledge and make sure to show your work for partial credit.

Make sure to cite your sources, if you decide to use more material to help you solve the exercises.

Remember that collaboration between students is allowed and encouraged, but please give your collaborators proper reference by letting me know who you worked with and what they contributed.

Only electronic submissions are accepted for this assignment.

You may either type your answers using LaTeX or other word processing software, or scan your handwritten answers and submit them. If the latter, please make sure the scanned copy is legible.

If submitting multiple files, create a folder and compress it (in either .zip, .rar, or .7z format, among others) before submitting.

Question 1: Eulerian paths (revisited)

Recall the DNA sequencing problem we discussed during class and the Eulerian path problem that was associated with it. A bioinformatics group, motivated by the same lecture material, decided to put this to the test. Alas, the graph they ended up with is **not** Eulerian. They believe the issue to be with their DNA reads, which ended up being noisy and hence they lost exactly one of the subsequences.

(a) First of all, had their graph been Eulerian, how could they solve the Eulerian path problem and obtain a valid DNA sequence? Using networkx and Python, find the Eulerian path using the following set of subsequences of $k = 3$:

$$s = \{CAT, ATT, ATG, TAT, TTG, TGC, TGT, GTA\}.$$

Construct the resultant graph and then identify an Eulerian path on it.

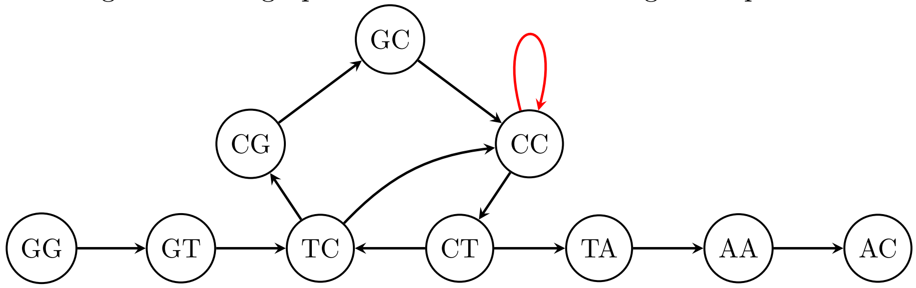
As a hint: use the **nx.eulerian_circuit(graph, source)** function that networkx provides us with to find an Eulerian cycle. Of course that assumes that you edit your graph so that an Eulerian cycle exists first!

(b) As mentioned originally, the graph the group obtained was not Eulerian. They attributed that to noisy data and they would like to somehow fix it. **Recommend a way for them to render the obtained graph Eulerian by adding the necessary nodes and edges.** Put your algorithm to use with the following set of subsequences of $k = 3$, s:

$$s = \{GGT, GTC, TCC, CCC, CCT, CTC, TCG, CGC, GCC, CTA, TAA, AAC\}.$$

The constructed network would be as in Figure 1:

Figure 1: The graph created with one missing subsequence.



Question 2: Matching problems

(a) Assume you are given an $N \times N$ chessboard with some missing places (call them "holes"). We want to place as many rooks as possible on the chessboard: rooks can be located at any of the $N \times N$ locations that are not "holes". However, we need to make sure that no two rooks can attack each other. Assume that two rooks attack each other if they are placed on the same row or column, regardless of whether there are "holes" between them or not. Figure 2 shows an example of a valid placement of rooks.

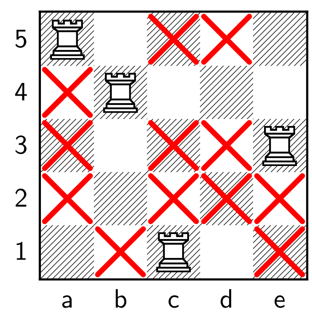


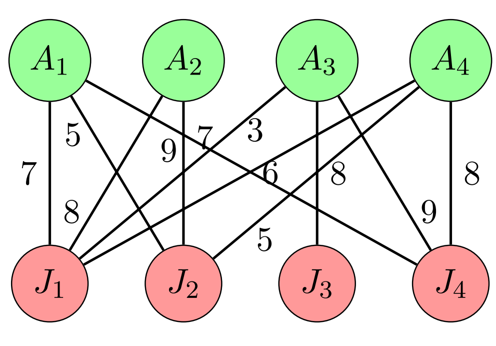
Figure 2: A valid assignment of four rooks in a 5×5 chessboard with 12 "holes" (marked with a red "X". Could we do better?

Set this problem up as a maximum matching problem. Provide a solution approach for solving it (based on the algorithm shown in class for solving maximum matching problems).

(b) Consider an assignment problem of n positions to n applicants. Each assignment comes with a cost $c_{ij} > 0$. We want to create a *stable and minimum cost* matching of positions to applicants with the extra restriction that no assignments is costlier than $\tilde{C}$. Provide a solution approach (could be a heuristic approach) for solving this problem, adapting the algorithm shown in class for solving maximum matching problems as well as Gale-Shapley's algorithm with preferences and the Hungarian algorithm for assignment problems.

To showcase how your algorithm behaves, you may use the following small instance. Consider a 4×4 assignment with costs equal to the values shown below and assume we pick $\tilde{C} = 9$.

	J_1	J_2	J_3	J_4
A_1	7	5	15	9
A_2	8	7	10	12
A_3	3	10	8	9
A_4	6	5	12	8



Question 3: Fixed cost facility location

In class, we discussed about the facility location problem and provided one of its formulations. However, in reality, opening a facility is associated with a *fixed cost* that needs to be paid before the facility can be used. On top of that, different facilities have different *capacities*; that is, they can only provide so many items to the customers.

Assume that a company is deciding where to open up their new warehouses from a set of 6 potential sites. Their goal is to satisfy all 10 of their customers in this location. The cost of shipping one item from a warehouse to a customer is shown in Table 1. Each facility in order to be "opened" comes with a fixed cost (the cost to open the facility) and a capacity (the amount of products that can be shipped from that facility): these are shown in Table 2. Last, the demands of the customers are known to be 25, 15, 20, 10, 15, 15, 25, 20, 30, and 20, respectively.

	Warehouses					
Customer	1	2	3	4	5	6
1	3	5	8	12	10	11
2	6	7	10	12	8	10
3	5	3	8	10	12	10
4	7	6	7	10	9	5
5	10	8	5	8	6	5
6	12	6	4	7	5	4
7	7	10	6	6	8	3
8	5	6	9	4	4	7
9	8	10	7	5	4	6
10	10	12	8	3	4	6

Table 1: The shipping costs from each facility (warehouses) to each customer.

Warehouse	Fixed Cost	Capacity
1	1750000	75
2	2000000	50
3	2500000	120
4	2250000	100
5	1500000	60
6	1000000	50

Table 2: The fixed cost and capacity of each potential site.

(a) Formulate the problem mathematically. Be careful to define your data, variables, constraints, and objective function, as shown in class. Then, use Gurobi to solve the problem with the data provided.

(b) What if... customers do not want to be satisfied by more than one warehouse? This behavior is called *single-sourcing* and is common with some products that cannot be combined when originating in different sources. What change would you make to your problem? When using Gurobi to solve the problem, do you obtain the same answer?

Question 4: Sudoku!

The game of Sudoku is a very popular, combinatorial, number-placing game. The puzzle maker provides a partially completed 9×9 grid. The goal is then to fill every square in the grid with a number between 1 and 9. The catch?

- Every row needs to have every number between 1 and 9 exactly once.
- Every column needs to have every number between 1 and 9 exactly once.
- The are 9 sub-squares (rows 1-3 and columns 1-3, rows 1-3 and columns 4-6, rows 1-3 and columns 7-9, rows 4-6 and columns 1-3, and so on). Every sub-square also needs to have every number between 1 and 9 exactly once.

Formulate a suitable integer program to solve the sudoku problem. With your formulation, use Gurobi to solve the Sudoku of Figure 3 (left) [1].

Figure 3: A sudoku puzzle (left) and its solution (right).

		6		8		5		
9			3				2	
	4				1			7
					5	6		
	2			4			1	
3								9
		1	9				3	
				2		4		
	5				7			

1	3	6	7	8	2	5	9	4
9	8	7	3	5	4	1	2	6
5	4	2	6	9	1	3	8	7
7	9	8	1	3	5	6	4	2
6	2	5	8	4	9	7	1	3
3	1	4	2	7	6	8	5	9
4	7	1	9	6	8	2	3	5
8	6	9	5	2	3	4	7	1
2	5	3	4	1	7	9	6	8

[1] An indicative solution is shown in the right side of Figure 3. That said, there could be alternative solutions – so please verify that your obtained solution forms a valid Sudoku!