Homework 4 - Midterm 1. **DUE OCT 6th 11:59 PM**

Name:

Student ID:

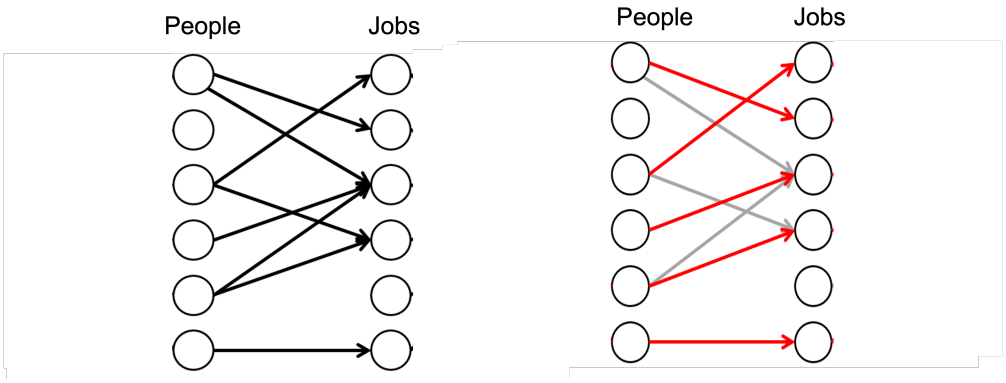
- Reasoning and work must be shown to gain partial/full credit
- Please include the cover-page on your homework PDF with your name and student ID. Failure of doing so is considered bad citizenship.

Maximum Matching

In order to solve this problem, we need to elaborate on how to use optimization of network flows, discussed in Lecture 4, to solve matching problems. The definition of *matching* is as follows:

Definition 1. *Matching A matching $\mathcal{M} \subset \mathcal{E}$ is a subset of edges in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ such that no two edges have vertexes in common. The* ***Maximum Matching*** *is the largest set $\mathcal{M}$ that is a matching set. This set is not unique in general.*

In bipartite graphs $\mathcal{G} = (\mathcal{A} \cup \mathcal{B}, \mathcal{E})$ the maximum matching set is useful to allocate, for instance, individual workers in $\mathcal{A}$ to individual tasks in $\mathcal{B}$, as shown in the figure:



There are two interesting problems.

1. One is finding such matching for the case where the workers are equally qualified to do the job; this amounts to finding the **maximum matching**. This problem can be solved through a *max-flow* optimization, using the Ford-Fulkerson algorithm (in Python `networkx.maximum_flow`).

2. The second one is finding an optimum match when the workers are *not equally qualified* to perform each job. In this case, not everyone can always pick the best for the task, so one way of addressing this problem is to maximize the average competence over all jobs. This problem can be mapped onto the so-called max-flow/min-cost problem, which uses the so-called *simplex method* (in Python `networkx.max_flow_min_cost`).

The numerical methods to solve them are explained, respectively, in the next two sections.

1. Solving the maximum matching with max-flow

A solution can be obtained by converting the graph to a flow network. To do this,

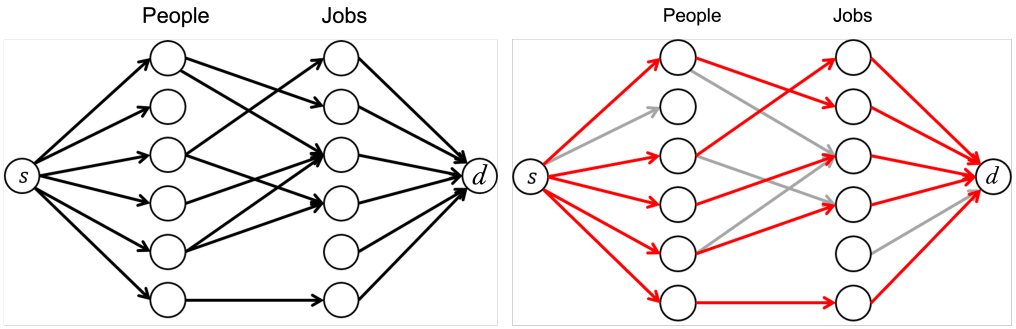
1. Convert the graph edges to a directed edges going from $\mathcal{A}$ to $\mathcal{B}$.

2. Add a source node s and a destination node t.

3. Draw edges from s to each node in $\mathcal{A}$ and from each node in $\mathcal{B}$ to t.

4. Set capacity $\overline{f}_{a,b} = 1$ for each edge in the new graph.

It can be shown that the max-flow solution of this flow network gives a maximum matching (might not be unique). The maximum flow equals the min-cut cardinality, which is going to be $\min(|\mathcal{A}|, |\mathcal{B}|)$.



2. Solving for the minimum cost or maximum utility matching

For the worker/job assignment problem, suppose the edge weight w_{ab} is the *competence* of worker $a \in \mathcal{A}$ to perform job $b \in \mathcal{B}$ or the *reward* for choosing the edge (a, b) in the matching, and the cost is equal to $-w_{ab}$ and capacities $\overline{\boldsymbol{f}} = \mathbf{1}$. The problem of finding a matching that minimizes the cost or maximizes the average competency is known *linear sum assignment problem*, and its integer formulation is as follows:

$$\begin{aligned} \text{minimize} \quad & -\boldsymbol{w}^\top \boldsymbol{f} && (1)\\ \text{subject to} \quad & \boldsymbol{f} \leq \mathbf{1}, && (2)\\ & \boldsymbol{B}^{\text{u}} \boldsymbol{f} \leq 1, && (3)\\ & \boldsymbol{f} \in \{0,1\}^{|\mathcal{E}|} && (4) \end{aligned}$$

where $\boldsymbol{f} \in \{0,1\}^{|\mathcal{E}|}$ is the integrality constraint, $\boldsymbol{B}^{\text{u}}$ is the unoriented incidence matrix of the bipartite graph[1] and the constraint $\boldsymbol{B}^{\text{u}}\boldsymbol{f} \leq 1$ ensures that at most one edge is picked for each node.

Because the integer problem is hard to solve, we can build a flow network problem that is not an integer problem but gives the same solution. As before, in addition to creating a flow network with capacities equal to one (i.e. $\overline{\boldsymbol{f}} = \mathbf{1}$), we use the edge cost equal to $-w_{ab}$, where we can assign zero costs $-w_{sa} = 0$ and $-w_{bt} = 0$ for the artificial edges we added from source s to the nodes in $\mathcal{A}$ and from the nodes in $\mathcal{B}$ to the target t. For this network flow problem, we already know that the maximum flow $y_s = \min(|\mathcal{A}|, |\mathcal{B}|)$ since this is the maximum flow over the minimum cut. The following problem formulation solves the minimum cost max-flow problem, with $\boldsymbol{B}$ the oriented incidence matrix:

$$\begin{aligned} \text{minimize} \quad & -\boldsymbol{w}^\top \boldsymbol{f} && (5)\\ \text{subject to} \quad & \boldsymbol{f} \leq \mathbf{1}, && (6)\\ & [\boldsymbol{B}\boldsymbol{f}]_s = -[\boldsymbol{B}\boldsymbol{f}]_t = \min(|\mathcal{A}|, |\mathcal{B}|), && (7)\\ & [\boldsymbol{B}\boldsymbol{f}]_u = 0, \quad u \neq \{s, t\} && (8) \end{aligned}$$

The python function to solve this problem is `networkx.max_flow_min_cost`.

With this in mind, you can now solve the following question. You need to simply understand that the formulation of the question is a special case of a matching problem and use the appropriate Python functions to solve them.

[1] `nx.incidence_matrix(G,oriented=False)`

1. (1–4 points) **Max-Flow for bipartite graphs**:*Python/NetworkX question*
The Davis Southern women social network is a dataset collected in the 1930s. It contains the observed attendance at 14 social events by 18 Southern women. It can be accessed on networkX using the `networkx.davis_southern_women_graph` generator.
   (a) Plot this graph in a bipartite layout (e.g., the left figure on Page 35 in Lecture 3).
   (b) Plot the graph that links the women that have been in the same social event.
   Hint: If you look at Lecture 3 slide 42, you can understand that the off-diagonal non-zero entries of $\boldsymbol{A}^2$ *are what you are looking for...*
   (c) Plot the graph of the social events where one or more guests were the same.
   Hint: same suggestion as the previous one.
   (d) Suppose that instead of social events, in the "Davis Southern Women Graph" the events are individual parking spaces available for EV charging, and that each woman can reach the spaces at about the same time. Let the weight, w_{ab}, on edge (a, b) indicate the distance to destination of woman a from parking space b.
   By setting $w_{ab} = 1$ for all $(a, b) \in \mathcal{E}$, answer the following questions:
      (i) Indicate the number of women that will not be able to park and charge their car.
      (ii) Convert the graph into a flow network using the steps highlighted in the introduction. Set the capacity of all the new edges to 1 and their weights to 0. Plot the flow network.
      (iii) Find the maximum bipartite matching for this graph using the flow network from the previous step.
   (e) Generate random weights for each link between the women and the parking spaces. These represent the distances. You can use a Rayleigh distribution for each (woman, spot) pair in the edge set using the function `numpy.random.rayleigh`.

```
import numpy
scale = 1
size = size = G.number_of_edges()
distances = numpy.random.rayleigh(scale=scale, size = size)
```

      (i) Convert the graph into a flow network using the steps highlighted in the introduction. Set the capacity of all the new edges to 1 and their weights to 0.
      (ii) Find the maximum bipartite matching that minimizes the average of distances of the edges in it.
      Useful methods: `networkx.max_flow_min_cost`

2. (1–2 points) **Spectral Clustering**:*Python/NetworkX question*
In this problem, we want to apply spectral clustering methods to the Davis Southern women social network. We want to find the groups of people that attend similar events (a group of friends). For this problem, use the bipartite graph **only** (**DO NOT** use the women's graph or the events' graph)
   (a) Calculate the degree of the women, and the degree of the events. What women have the most socially active life? What are the most popular events?
   (b) Compute the eigenvector centrality and the betweeness centrality of the graph. What do you observe?
   (c) Find the eigenvectors and eigenvalues of the Laplacian of the graph. Plot the eigenvalues in ascending order.

(d) Use spectral clustering to cluster the network. Use three clusters and plot the results. (*Hint: You may want to use k-means to do the clustering. Also, please use a different color for each cluster. Below, you will find a solution to the clustering problem*). What do you observe? Comment your results