HW2 (Group)

Network Analytics, MSc BA

Due by **8 Feb 2021**

Programming: You are allowed to consult on the programming part with your colleagues and on the web but the final code has to be written and debugged entirely by your group.

In the following, you have to search for the appropriate functions in NetworkX/numpy/scipy yourselves.

1. (10 points) In a previous cohort I asked the students to write down the number of emails they exchanged with other members of the cohort. Data might be having some problems, but we proceed nevertheless. Use built-in functions in NetworkX on the data **HW2_ who_talks_to_whom.xlsx** to do an organizational network analysis report (1 page max)--- essentially calculating centrality measures (try at least one eigenvalue based one) and clustering coefficients and gaining some insight into the network. The objective for me (your client) is to identify who are the leaders and opinion-makers in the cohort, and any other qualitative insights you might obtain.

2. [15 points]

For this problem, you don't need to program from scratch but just modify the TSP example program on the gurobi website.

The data **HW2_qatar_tsp.txt** contains latitude and longitude data of 194 cities from Qatar. Calculate the distance matrix (use an approximation like here which is quite accurate for short distances; or use packages like haversine or geopy). The x and y-cordinates are the latitude and longitude in decimal form multiplied by 1000. EUC_2D means take these as Cartesian co-ordinates. Can also use haversine treating them as longitude and latitude. You are free to use any other functions you find.

a. (5 points) Plot the latitude and longitude as a scatter plot using a drawing package (some options are: matplotlib basemap toolkit (the most advanced, but also the most difficult to use), geopy, gmplot, plotly …). It should look roughly like this.

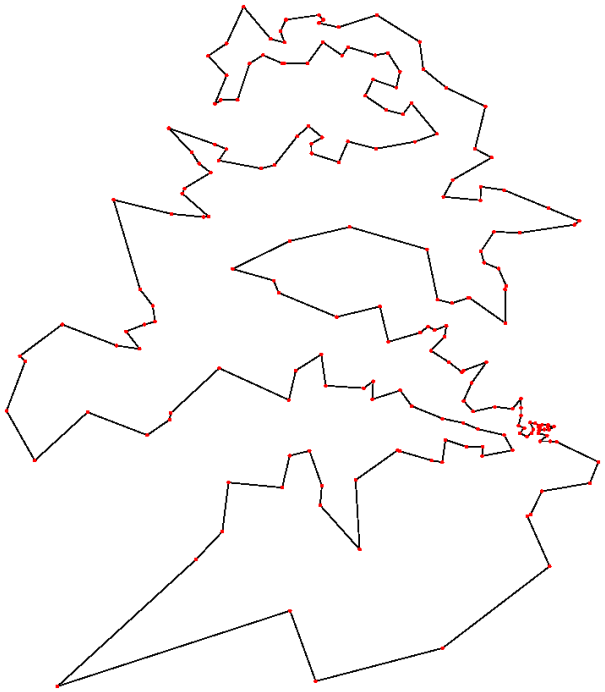
b. [5 points] The most powerful integer programming solver is Gurobi (along with CPLEX). They give a free one-year license if you download and install from a University IP address. Use the most powerful computer you have in your group (cores and memory).

Connect with the Python interface of Gurobi and find an optimal tour using the sub-tour elimination integer programming formulation we did in class. You cannot generate all the sub-tour elimination constraints at once (too many). Instead, (the example program on Gurobi) does the following standard algorithm technique to solve problems with an exponential number of constraints:

i. Start with a problem with only the degree =2 constraints.
ii. Check the resulting solution for sub-tours. If there is a sub-tour, add the sub-tour elimination constraints corresponding to a cut defined by the sub-tour (i.e. you are eliminating that sub-tour in the next round).
iii. Repeat till you find a tour (in which case, it is optimal). Make sure you do not discard the solution from the previous round, but start from what you had found.

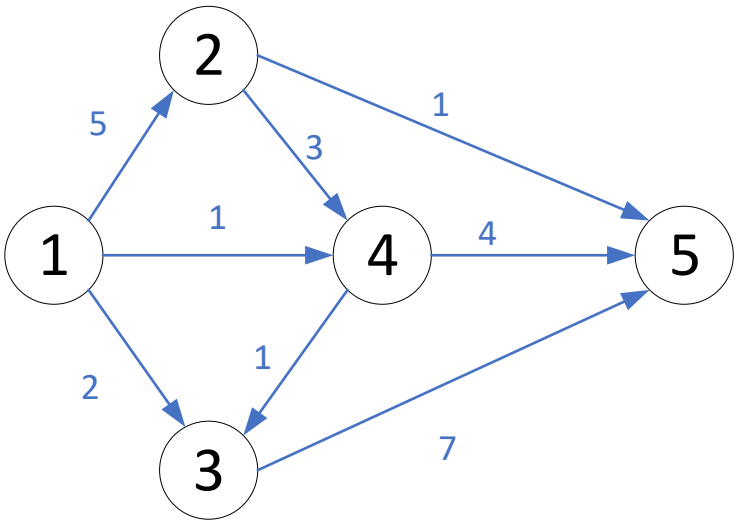
This method of generating constraints on the fly is suitable when the number of constraints are exponential in the problem size. The dual version of this is called column generation.

c. (5 points) Plot the resulting tour on the scatter plot. The optimal tour should look like this:



3. (10 points)

a. Formulate the problem of sending the maximum amount of flow from Node 1 to Node 5 in the following network (the edges have capacities as shown in the figure) as a linear program



b. Formulate and solve the dual using a linear programming solver (say Excel; *Hint: do you really need to solve the dual, or solving the primal is enough?*)

Define a s-t *Cut* as a partition of V into disjoint sets S, T (i.e., S $\cup$T= V and S$\cap$ T = $\varnothing$) with s in S and t in T (*cuts will reappear in Community Detection*)

Define *capacity of the cut*, Cap(S, T) = sum of the capacities of edges from S to T
Define *Flow across the cut*, Flow(S,T): net flow out of S = sum of flows out of S - sum of flows into S.

c. A very famous and central theorem in combinatorial optimization states: Max-flow = Value of Min-cut (Max-flow/Min-cut theorem). Proving *Max-flow* $\leq$ *Value of Min-cut* is easy---give a reason why this should be so.
d. (*challenging*) From the dual values (i.e. the shadow prices) of the linear program corresponding to the optimal extreme point (basic feasible) solution, find a minimum-capacity cut. (*The cut itself would be easy to find by inspection, but you have to come up with a connection between the dual values and the cut; examining the solution values would give you a clue*).