HW2

Network Analytics, MSc BA, 2017_18

This is a group HW. It might look long, but I expect you to split the programming part equally amongst yourselves. It will be worth 10% of your grade. We will be covering the material of questions 2 and 4 on 27 Nov.

First things first: Due by **25 Nov, 12 noon – this part is not graded as such, but you will receive a 0 on this HW if you don't submit this or if your row in the submission is blank.**

Who-talks-to whom network is the basis for most social network analytics and workforce analytics. What is often glossed over are the challenges involved in collecting this data.

Every member of your group should fill out **class_2017.xlsx** with the number of communications you sent and received (separate sheets in the Excel file) from every other member of the cohort (not just your group). I deliberately am leaving out specifying how (chat over coffee, emails, text messages, whatsapp?). I ask you to limit the time period however, to the previous one week. Discuss with your group how you would capture this. This data will be anonymized and posted back for Question 1 below

Discuss with your group the problems and challenges in collecting the data --- for class discussion on 27 Nov.

Submission by **6 Dec 2017 12 noon (the HW will be graded for 30 points)**

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 on your own.

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

Programming part (15 points or 20 points if you attempt the challenge part)

1. (5 points) (You will get the data for this on 25 Nov 2017 8PM): The data collected will invariably be fraught with problems. We proceed nevertheless. Use built-in functions in NetworkX on the data **HW2_ who_talks_to_whom.txt** 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 your cohort.

2. [10 or 15 points] The data **HW2_tsp.txt** contains latitude and longitude data of 38 cities in a country in Africa (Djibouti). 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. 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.

DO EITHER (b) OR (c) --- no need to do both; if you submit both only (c) will be graded

b. (5 points) Use the or-tools traveling salesman routine to find a tour of the 38 cities. If it is not finding a tour fast enough, try a subset of 10 cities (then 15 and so on). My suggestion is to try to understand the solved example given there and try to recreate it for this data.

c. [Challenge problem., 10 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,

   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. Compare the optimal solution with the or-tools solution.

The optimal tour is here.

d. Plot the resulting tour on the scatter plot

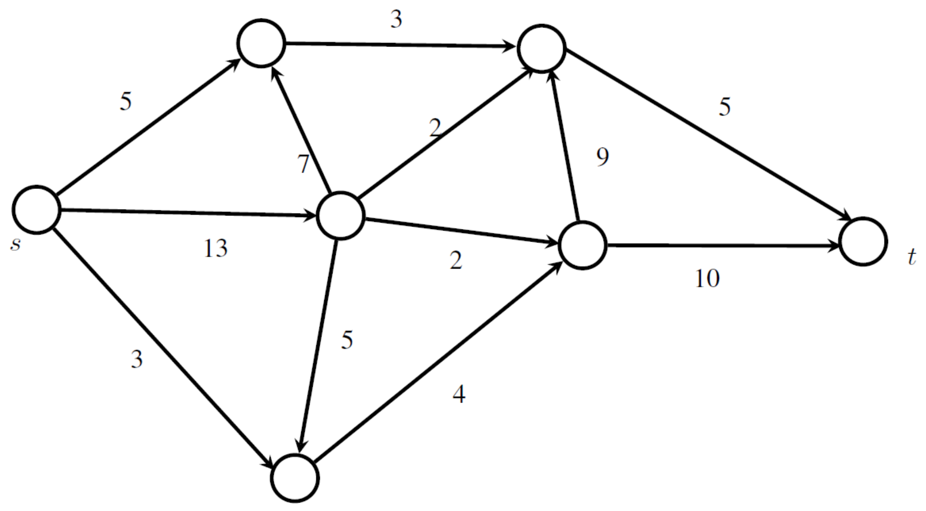
Theory (10 points)

3. (5 points) Show that the greedy algorithm works in finding the optimal (min or max) weighted spanning tree. i.e., you have to show (a) it terminates in finite time---better if you can show polynomial time (b) the tree that it finds is an optimal tree.

   For your benefit, I attach a chapter from a textbook (`greedy.pdf`*) giving you the reasoning, as this is standard introductory material in most combinatorial optimization books. However, I want you to digest the material and write the answer in your own words (max one page).*

4. (5 points)

   a. Formulate the problem of sending the maximum amount of flow on the following network (the edges have capacities as shown) as a linear program



   b. Formulate the dual of the linear program as well.
   c. Solve using a linear programming solver (say Excel)
   d. Define a s-t *Cut* as a partition of V into disjoint sets S, T (i.e., $S \cup T = V$ and $S \cap T = \emptyset$) with s in S and t in T.
   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. 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 general way of extracting the cut from the optimal dual variables).