Social Networks and Online Markets

Homework 1

Due: 9/5/2021, 23:59

Instructions

You must hand in the homeworks electronically and before the due date and time.

The first homework has to be done by each **person individually**.

Handing in: You must hand in the homeworks by the due date and time by an email to `aris@diag.uniroma1.it` that will contain as attachment **(not links to some file-uploading server!)** a .zip or .pdf file with your answers.
After you submit, you will receive an acknowledgement email that your homework has been received and at what date and time. If you have not received an acknowledgement email within 2 days after you submit then contact Aris.

The solutions for the theoretical exercises must contain your answers either typed up or hand written clearly and scanned.

If you need any technical help, you can email Andrea Mastropietro: `mastropietro@diag.uniroma1.it`.

For information about collaboration, and about being late check the web page.

Problem 1. Consider the following modification of the Barabassi–Albert preferential attachement model that we did in class: When a new node arrives at time t again it comes with ℓ edges. However, this time each edge selects a node v with probability proportional to the degree d_v plus a constant c, that is, the probability equals

$$\frac{d_v + c}{(t-1)(2\ell + c)},$$

where $c \geq -\ell$, as we describe at the end of Chapter 4 in the notes (so for $c = 0$ this is the Barabassi-Albert model). Show that the degree distribution that we obtain as $t \to \infty$ is approximately a power law with exponent $3 + c/\ell$.

Problem 2. During the lectures we proved Cheeger's inequality for regular graphs. The inequality can be generalized to general (not necessarily regular) undirected graphs. Furthermore, the same algorithm that we used for proving the "hard direction" of Cheeger's inequality for regular graphs, can be used for general graphs. Let's call this algorithm $\mathcal{A}$.

We define the normalized Laplacian $\tilde{L}$ of a general graph as:

$$\tilde{L}_{ij} = \begin{cases} 1, & \text{if } i = j \\ -\frac{1}{\sqrt{d_i d_j}}, & \text{if } i \text{ and } j \text{ are adjacent, and} \\ 0, & \text{otherwise,} \end{cases}$$

where d_i is the degree of node i.

Implement algorithm $\mathcal{A}$ in Python. (You can use other programming language if you prefer, but probably Python is the easiest.) Make sure to use sparse-matrix representation, and when you calculate an eigevector corresponding to λ_2, use an option that calculates this eigenvector, and not the entire set of eigenvalues and eigenvectors.

After implementing the algorithm apply it on some of the datasets at: `http://snap.stanford.edu/data/index.html`.

When attempting to make the partition, choose the partitioning that minimizes the sparsity. Apply it recursively to find more than two communities. Automate the process so that you can run it in various datasets.

Experiment with different graphs and different sizes. First work with some small ones (if needed you can sample a subgraph out of one of the graphs in the website) and then try it for larger ones. We leave it up to you to decide how many communities to create or, equivalently, when to stop this process. There are not a single correct way to proceed but we ask you to justify your approach.

For each dataset that you try, please report:

- The dataset that you use
- Number of nodes and number of edges
- The number of communities that you have found
- The size of each community
- The number of nodes in each community and the number of outgoing edges.

Return a short document with your approach, with the results, as asked above, the code, and, for each dataset, the communities (i.e., sets of nodes) that you have found.