Bangalore School on Statistical Physics XIV, Sept 2023

Statistical Mechanics of Complex Networks

Tutorial: Algorithms for network metrics

Sitabhra Sinha

The Institute of Mathematical Sciences, Chennai

How do you represent a network in a computer ?

Information about a network can be stored in computer memory in a number of possible formats

How you store the information about the vertices and edges can affect speed of computation & memory usage !

- The vertices are represented by unique labels, viz., 1, 2, 3, …, N
- To represent edges, one can use different possible representations, e.g.,
  - Adjacency matrix
    Simple (stored as 2-dimensional array of integers) and fast in finding/removing edges (O(1)) but for **sparse graphs** is inefficient in terms of use of memory and takes O(N) operations for neighbour enumeration
  - Adjacency list [Most popular data storage format]
    List containing labels of other vertices to which each vertex is connected: economical in terms of memory usage and takes O(L/N) operations for neighbour enumeration in **sparse graphs** but also for finding/removing edges
  - Adjacency tree
    Like adjacency list, but list of neighbors of each vertex is stored as a binary tree (values stored in left child of node i and its descendants are less than value stored in node i) takes O(log(L/N)) for adding/finding/removing edges

Calculating degree distribution

In adjacency list, information about neighbors for each vertex is maintained
To obtain degree for each node, we need simply count the number of entries in the neighbor set

For adjacency matrix, we need to sum together all the entries of i-th row or column to find the degree of the i-the node

Once the degree of all nodes $\{k_1, k_2, \ldots, k_N\}$ are known, create a histogram

- Construct an array comprising k_{max} "bins" – each bin storing the number of vertices of a specific degree (up to the maximum degree).
- Set all array elements initially to zero.
- Run through each vertex in turn, find its degree q (say) and add 1 to the q-th bin.
- Once all N vertices have been gone through divide all array elements by N to obtain p_k.

Problem: For small bin widths, may look extremely non-uniform, but with larger bins we lose resolution

Solution: Construct complementary cumulative degree distribution $p_{k>K}$ by sorting the degrees in descending order, ranking them from 1 to N and plotting the rank divided by N as a function of the degree

Calculating clustering

The local clustering coefficient of a node i is

$$C_i = \frac{\text{(number of pairs of neighbors of } i \text{ that are connected)}}{\text{(number of pairs of neighbors of } i\text{)}}$$

The denominator is just $\frac{1}{2} k_i (k_i - 1)$, trivial to obtain once degree of node i is known

To calculate the numerator

- ❑ go through every pair of distinct neighbors (p,q) of vertex i (with $p<q$)
- ❑ For each pair we determine whether an edge exists between them
- ❑ count up the number of such edges.

The overall clustering coefficient of a network is

$$C = \frac{3 \text{ (number of triangles)}}{\text{(number of connected triples)}}$$

The denominator is $\frac{1}{2} \sum_i k_i (k_i - 1)$, trivial to obtain once degree of node i is known

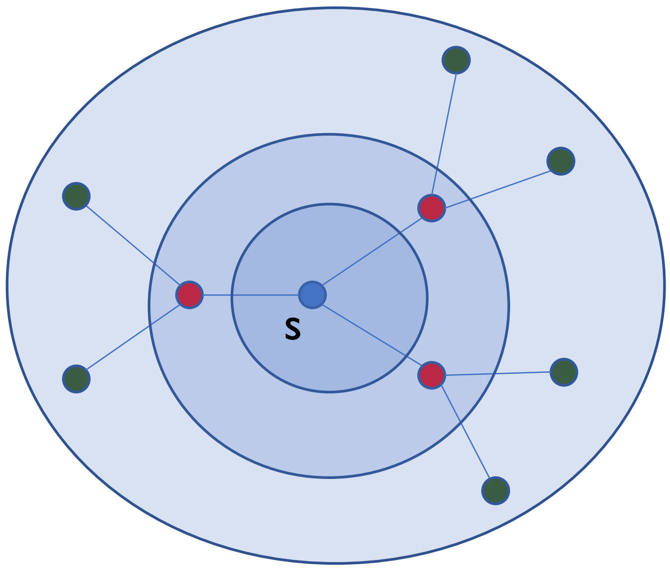
To calculate the numerator

- ❑ consider for every vertex i (=1, 2, …, N) each pair of neighbors (p, q) with $p< q$
- ❑ find whether they are connected by an edge
- ❑ add up the total number of such edges over all vertices

Calculating path length

Breadth-first search algorithm

finds the shortest (geodesic) distance from a single source vertex *s* to *every other* vertex in the network



- ❑ Start from vertex *s*
- ❑ Initially the distances to all other vertices are unknown
- ❑ Find all the neighbors of *s*
  By definition these have distance 1 from *s*.
- ❑ Then find all the neighbors of *those* vertices, excluding those already visited
  These vertices have distance 2 from s

❑ Then find their neighbors, excluding those already visited – these which have distance 3, and so on.

❑ On every iteration, the set of vertices visited grows by one step.

❑ Keep iterating until all nodes are visited