Assignment 3A: Community Structure Analysis

Date of issue: 14.03.2019
Akansha Bhardwaj: `akansha.bhardwaj@unifr.ch`

1 Community Detection

Task 1 Consider the network of *email conversations*[1] from a company. Assuming that Figure 1 is a subgraph of the complete network, perform the following:

1. Find all maximal cliques with degree $k \geq 4$.
2. Find all communities using Clique Percolation Method (CPM) with $k = 3$.

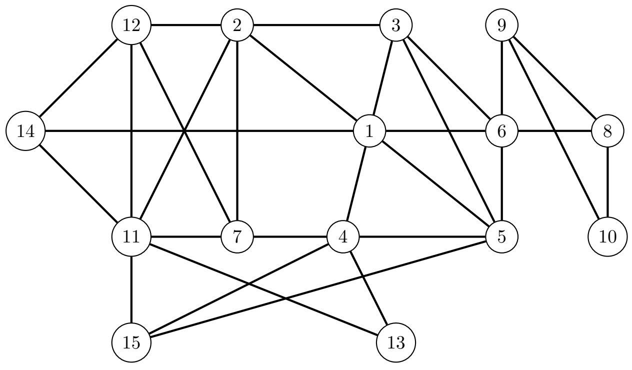

Figure 1

3. Using *python-networkx*[2] library, load the given *email conversations* data. Each row in this file represents edge information. For example, if first row entry is '116 101', this implies that there is an edge from node '116' to node '101'. For the generated graph, perform the following:
   (a) Find the total number of cliques in the graph.
   (b) Find the number of maximal 8-cliques.
   (c) Find the largest clique(s) and report the time it takes for this calculation.

Task 2 Clique Percolation Method.

1. Consider the incomplete implementation of CPM shown below. Finish the implementation of CPM by writing the code of the method $return\_X(clique, node\_cliqueinfo)$.

[1] `https://drive.google.com/file/d/1WWIo2eMykSsjXCv8S6AsU6MpnoxMm6oN/view?usp=sharing`
[2] `https://networkx.github.io`

```
import networkx as nx
from collections import defaultdict

def percolated_cliques(G, k):
    percGraph = nx.Graph()
    cliques_set = [frozenset(c) for c in nx.find_cliques(G) if len(c)>=k]
    #store all information of cliques in a clique set

    percGraph.add_nodes_from(cliques_set)
    #Generate a graph with nodes containing clique information

    #Generate a dictionary with node as 'key' and cliques
    #in which they occur as 'value'
    node_cliqueinfo = defaultdict(list)
    for clique in cliques_set:
        for node in clique:
            node_cliqueinfo[node].append(clique)

    # For each clique, see which adjacent cliques percolate
    for clique in cliques_set:
        for neighbour_Clique in return_X(clique,node_cliqueinfo):
            if len(clique.intersection(neighbour_Clique)) >= (k - 1):
                percGraph.add_edge(clique, neighbour_Clique)

    # Connected components of clique graph with perc edges
    # are the desired percolated cliques
    for component in nx.connected_components(percGraph):
        yield(frozenset.union(*component))
        print component
    return nx.connected_components(percGraph)

def return_X(clique, node_cliqueinfo):

    #Complete the function
```

2. Using $k = 12$, find the communities in the *email conversation* graph using the above completed implementation.

Task 3 Girvan-Newman clustering algorithm.

1. Consider the small network in Figure 2.

   (a) Calculate the edge-betweenness centrality for all the edges. Which edge should be first removed during Girvan-Newman clustering? Explain your answer.

2. Load the *email conversation* network dataset, and perform the following:

   (a) Find the largest connected component of the graph.

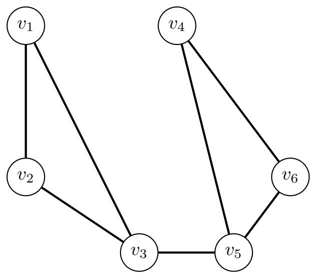

Figure 2

(b) Apply Girvan-Newman method from *python-networkx* library for the first iteration level and report the number of resulting communities.

(c) Compute the size of each community after second iteration.