SAPIENZA
UNIVERSITÀ DI ROMA

INGEGNERIA INFORMATICA, AUTOMATICA E GESTIONALE "ANTONIO RUBERTI"

Social Networks and Online Markets

Assignment 1

Professors:
Aris Anagnostopoulos
Stefano Leonardi
Georgios Birmpas

Students:
Vano Mazashvili

Academic Year 2022/2023

Contents

Problem 1.

1.1 Introduction

1.1.1 Problem Statement

The objective of this problem is to implement and investigate various graph models that we have seen and demonstrate fundamental understanding by experimenting with the number of graph model parameters, measuring various properties, and concluding how the changes in parameters affect graph properties.

1.1.2 Graph Models

For this exercise, we are working with Erdős–Rényi random graph model, Watts-Strogatz small-world model, and Barabási–Albert preferential attachment model. Each one provides different type of network, helping us investigate various real-world phenomena: we can study the behaviour of the completely random networks, understand how its structure adapts with local and global connectivity, and demonstrate and investigate temporal evolution of the networks based on the preferential attachment, respectively. We will implement the models, then explore and analyze how different parameter combinations influence networks' behaviour - degree distribution, diameter, and clustering coefficient and other relevant properties.

1.1.3 Used Parameters

Several problem-based parameters are used for each specific model. For Erdős–Rényi random graph model, we use n and p parameters, while Watts-Strogatz small-world model uses n, k, β and Barabási–Albert preferential attachment model n and l parameters.

Under the domain of this homework, the parameters are:
• n - number of nodes
• p - probability of an edge to exist
• k - number of initial edges adjacent to each node
• β - probability of rewiring
• l - number of neighbors that a newly arrived node comes with, it is the exponent of the power-law distribution.

We will discuss and experiment with the various attributes of the given graph properties in depth in the following chapters

1.1.4 Other Interesting Graph Properties

Besides the mentioned parameters, we are working with the graph connectedness and correlation coefficient.

1.2 Methodology and Setup

Graph class is created in order to easily implement three mentioned graph models through inheritance. Because the generation process is different for each types of graph model, node and edge attributes and respectively add_node and add_edge methods are inherited. Graph class is extended with the unique generation methods. For Erdős-Rényi random-graph model, we use the aforementioned n, p parameters and use them to construct a new graph by deleting any previous ones and adding a new node. Through for loop, we populate the graph with the next node and use nested loop to construct an edge through utilizing p parameter with a $random()$ function.
Unlike ER graph model, after resetting nodes and edges attributes, we construct Watts-Strogatz model by adding the nodes first, without constructing any edges. Then we iterate through nodes and use loop from 1 to $k/2 + 1$ to determine the neighbor nodes and then add edges between them. We iterate the same way to determine which edges to rewire using random and β parameter comparison.

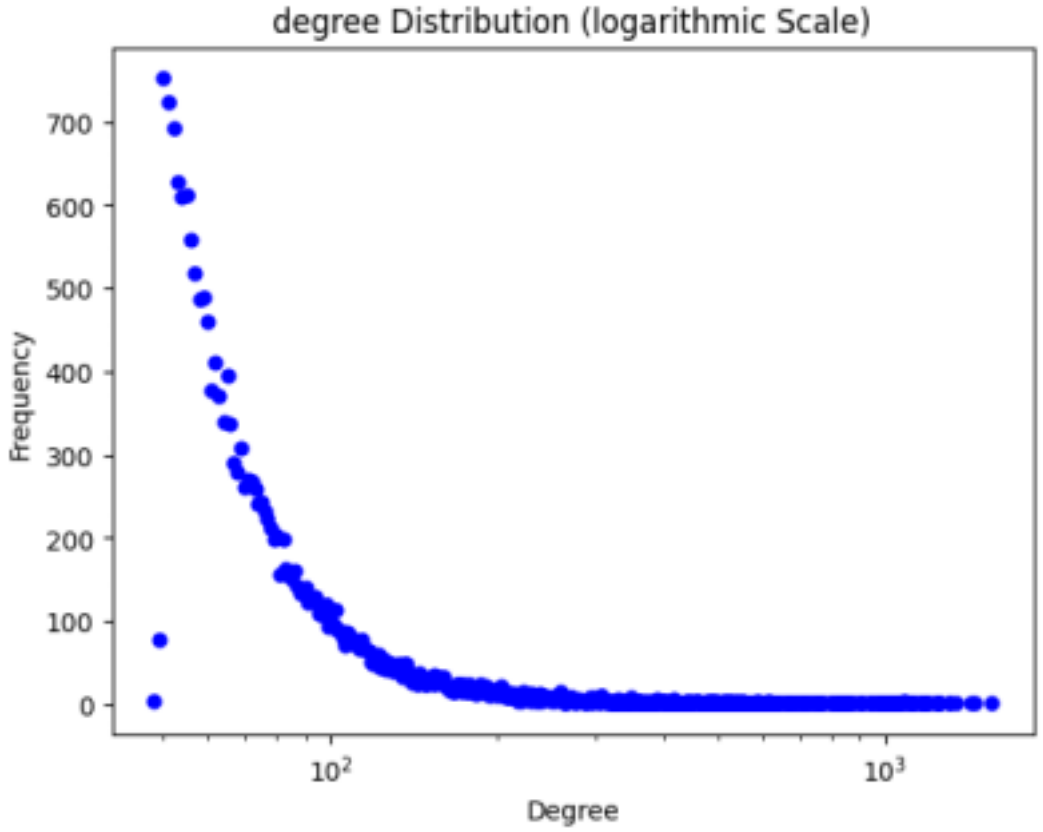
1.3 Results

Degree Distribution For the Erdős-Rényi random-graph model, each pair of nodes has a fixed probability p of being connected by an edge. this means that a given node v has $n-1$ potential edges, each with an independent probability of p. Thus, the degree distribution is a binomial bell-shaped distribution, being approximately normal when the graph is sufficiently large (higher n values).

The degree distribution for Watts-Strogatz small world graph model depends on its parameters. As per original graph, regular ring lattice shape with a high clustering coefficient characterized by the lack of short average inter-node path lengths requires the property of small-worldness, which is addressed by introducing randomly rewiring (replacing) each edge in the network with a given probability. Based on rewiring probability β, the shape of the degree distribution can change from Dirac delta ($\beta = 0$) to more broad, gaussian-like distribution (β 1). With the higher values of β, degree distribution becomes similar to that of a random graph and usually has a pronounced peak at k. Although for the sufficiently high parameter values, the nodes look similar, in particular, theoretically, each node gets to have the same degree.

The problems regarding degree distribution presented in Erdős-Rényi random-graph and Watts-Strogatz small world graph models are addressed by the Barabassi-Albert preferential attachment model, generating heavy-tailed degree distribution. It attempts

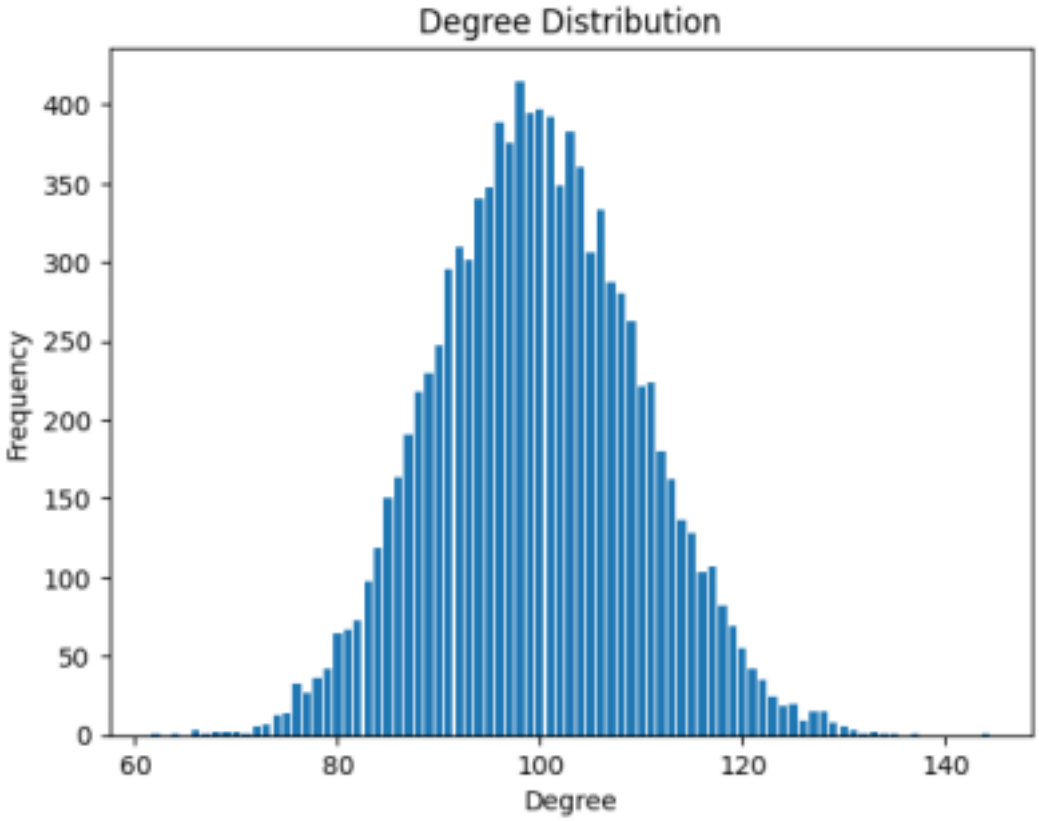
to grow the graph from the initial node by adding subsequent nodes and edges. This allows us to manupulate the probabilities of the new edges being formed, in this case, once the new edge is created, the probability to have node v as an endpoint is proportional to the degree of node v. This property allows a power-law, or heavy-tail distribution, representing real-world network behaviour more precisely.




```
BA Diameter: 3
BA Clustering Coefficient: 0.019026751954177537
```

Diameter As per graph diameter, it tends to be relatively small for the Erdős-Rényi random-graph model when the graph degree is high, creating usually well-connected network. However, we cant say that for Watts-Strogatz small-world model. Due to the nature of the model, we get a regular lattice (or closely resembling) node formation having a relatively large graph diameter, given that β is sufficintly low. As we increase rewiring probability β we should obtain small-world property and thus decrease diameter, while maintaining a high clustering coefficient.
Barabasi-Albert preferential attachment model creates possibility to create highly a few well connected nodes on a "rich-get-richer" basis, indicating relatively small diameter even if n is sufficiently large.
The graph diameter scales logarithmically for all three graph models, although Barabasi-Albert preferential attachment model tends to have lower diameter than Erdős-Rényi random-graph model.

Figure 1: ERdos-Renyi Degree distribution



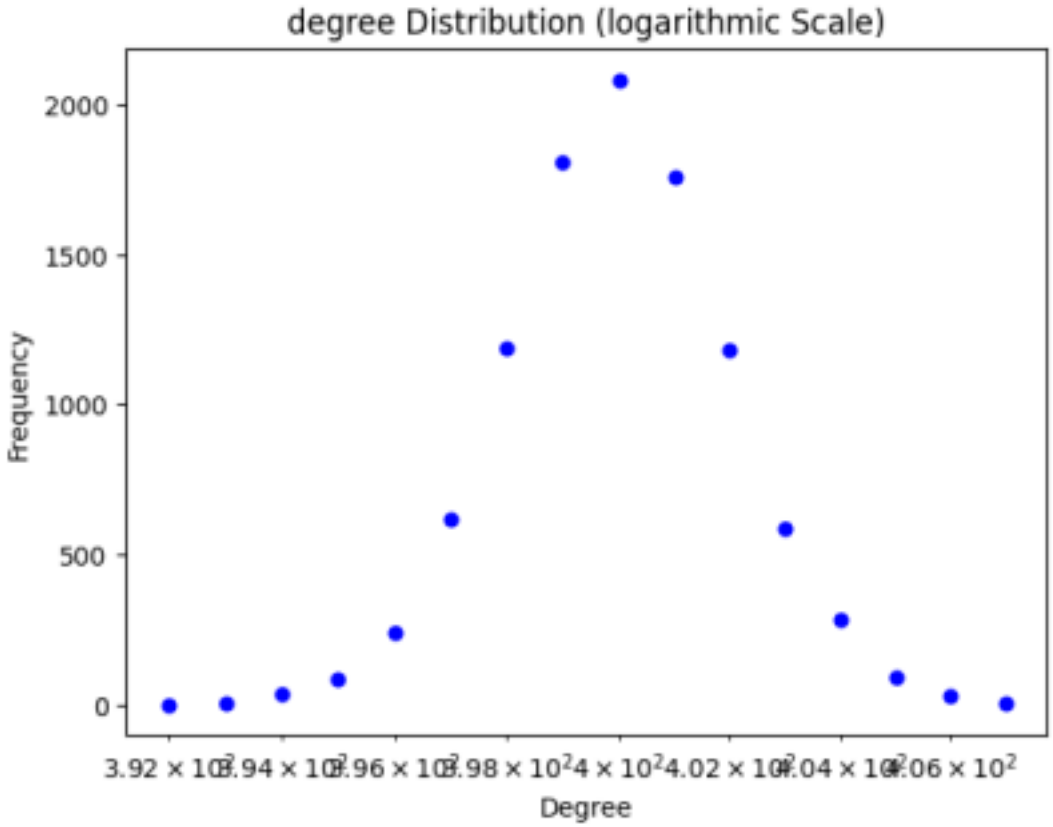
Clustering Coefficient Experimentally, we are going to show difference in clustering coefficient between theese three graph models; Prove that the Erdős-Rényi random-graph model usually has lower clustering coefficient due to the graphs almost pure randomness, while the Watts-Strogatz small-world and Barabasi-Albert preferential attachment models exhibit higher CC values (although CC depends on the graph parameters for the Watts-Strogatz small-world model, it enables us to interpolate between regular and random graphs)

@booklatex2e, author = Leslie Lamport, year = 1994, title = LaTeX: a Document Preparation System, publisher = Addison Wesley, address = Massachusetts, edition

	Erdos-Renyi			Watts-Strogatz			Barabasi-Albert		
Parameters	n =10^5 p =0.01	n =10^5 $p = 0.11$	$n = 10^5$ $p = 0.3$	$n = 2 \cdot 10^5$ $k = 200$ $\beta = 0.11$	$n = 2 \cdot 10^5$ $k = 400$ $\beta = 0.11$	$n = 2 \cdot 10^5$ $k = 200$ $\beta = 0.2$	$n = 2 \cdot 10^5$ $l = 5$	$n = 2 \cdot 10^5$ $p = 50$	$n = 2 \cdot 10^5$ $p = 100$
Clustering Coefficient	0.009978	0.110009	x	0.5277479	0.547843	0.387887	x	0.0190268	0.0311726
Diameter	3	2	x	3	3	3	x	3	3
Connectedness	True	True	True	True	True	True	True	True	True
Correlation Coefficient	0.00098	-0.00109	1.39922	0.000443	-0.000809	-0.001199	-0.02120	0.0014969	0.00547

Table 1: Graph properties according to different parameters

$= 2$



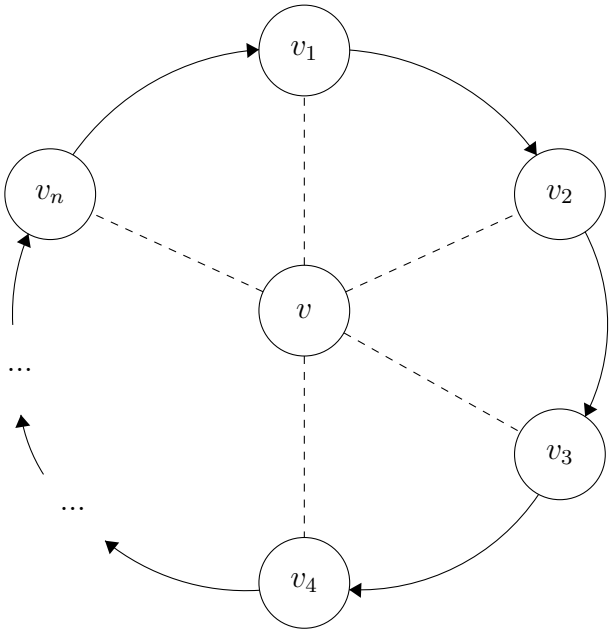

```
WS Diameter: 3
WS Clustering Coefficient: 0.7263757595527361
```

Problem 2.

2.1 Problem description

We construct a cyclic directed graph where v_i connects to v_{i+1} and v_n has an edge with v_1. In the center of the graph, we have a node v. With probability p the undirected edges connect node v to the surrounding ones included in the circular graph. The weights for the undirected edges is 0.5, while for directed - 1. That means that with a probability of p^2 we can take a shortcut from one node to another.
In this problem, we are required to find

1. Probability $P(l, k)$ that shortest path between v_i and v_j is exactly l.
2. Distribution $P(l)$ of the shortest path between the nodes on the cycle.



2.2 Methodology

It is apparent that we can "skip" the nodes from the circular graph by taking a shortcut, shortening the distance between two nodes. Although, using a shortcut is only viable once, because if we "jump" more than once, creating an undirected edges, the resulting graph will connect initial first node we are jumping from to the last node of the last jump, essentially excluding the "middle man" from the shortest path by rendering it redundant. That means that we can take a shortcut only once, although, we can skip several nodes with one shortcut.
From here, it becomes apparent that for any given node pair, the number of valid probable paths between two nodes $= (k^2 - k + 2)/2$ for $1 \leq l \leq k$. From here, we can derive the formula:

$$P(l,k) = \begin{cases} \dfrac{lp^2}{(k^2-k+2)/2}, & \text{for } 0 < l < k \\ 1 - \dfrac{lp^2}{(k^2-k+2)/2}, & \text{for } l = k \end{cases}$$

As for the second part of the problem, in order to calculate the distribution of the shortest path between two arbitrary nodes on the cycle $P(l)$, first, we calculate the joint probability $P(l,k)$ using the conditional probability formula, after which, we marginalize the result over k and get $P(l)$.

For $0 < l < k$, the conditional probability $P(l,k) = P(l|k) \cdot P(k)$. Given that

$$P(l|k) = \frac{lp^2}{(k^2-k+2)/2},$$

where $P(k)$ has an uniform distribution $P(k) = \frac{1}{n-1}$, plugging the formulae, we get

$$P(l,k) = \frac{lp^2}{(k^2-k+2)/2}\frac{1}{n-1}$$

.

After marginalization over k, we get the distribution

$$\begin{aligned} P(l) &= \sum_{k=1}^{n-1} P(l,k) \\ &= \sum_{k=1}^{n-1} \frac{lp^2}{(k^2-k+2)/2}\frac{1}{n-1} \end{aligned}$$

For marginalization, because we need to get the joint probability when $l = k$, we should add 1/(n-1) as well under the sum, because thats the probability P(l=k),

Problem 3.

3.1 Problem Statement

In this problem, we have a given graph $G = (V, E)$. We are to find:

1. Find the densest subgraph.
2. Find a minimum cut.
3. Demonstrate (by calculating λ_2, $\phi(G)$, etc.) that Cheeger's inequalities hold for this graph.
4. Find the cut that satisfies Part 3 (and show that it does.)

3.2 Methodology and Solution

To find the densest subgraph, it is convinient to use Charikar's greedy approximation algorithm. Basically, we iteratively add vertices to the subgraph in order to maximize its density.
Given a graph $G = (V, E)$, the algorithm has to find a subgraph optimizing density. Meaning that $S \subseteq V$ maximising $|E(S)|/|S|$ where $E(S) \subseteq E$ is a set if the edges with endpoints in S. Although, Charikar's algorithm is not optimal. This is evident in given example - while the optimum is 2, the algorithm will spoil it as soon as it takes the first action by removing a node with 4 edges.

the pseudocode is the following (taken from the lecture):

1.Function GreedyDensestSubgraph (G)
2. Input: $G = (V, E)$: Simple undirected graph
3. Output: A set of nodes $S \subseteq V$
4. $S \leftarrow V$
5. $S^G \leftarrow V$
6. while $|S| > 1$
7. $v = \arg\min_{v \in S} \deg_S(v)$
8. $S \leftarrow S \backslash \{v\}$
9. if $f(S) \geq f\left(S^G\right)$
10. $S^G \leftarrow S$
11. end if
12. end while
13. return S^G

according to which, we get:
1. Start with the input graph G consisting of 10 nodes

2. $S = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$
3. Initialize a copy of the graph S^G
4. Enter the while loop since $|S| > 1$
5. $argmin_{v \in S} deg_s(v)$ is all same for all nodes so we choose to remove a node arbitrarily
6. In this case, $S \leftarrow \backslash\{1\}$, thus $S = \{2, 3, 4, 5, 6, 7, 8, 9, 10\}$
7. Now, because $f(S) \leq f(S^G)$, $S^G \nrightarrow S$
8. Iterating the while loop, several nodes have the same $argmin_{v \in S} deg_s(v)$ value of 3. we choose to remove a node arbitrarily
9. In this case, $S \leftarrow \backslash\{2\}$, thus $S = \{3, 4, 5, 6, 7, 8, 9, 10\}$
10. Now, because $f(S) \leq f(S^G)$, $S^G \nrightarrow S$
11. 2 nodes have the same $argmin_{v \in S} deg_s(v)$ value of 2. we choose to remove a node arbitrarily
13. In this case, $S \leftarrow \backslash\{3\}$, thus $S = \{4, 5, 6, 7, 8, 9, 10\}$
14. Now, because $f(S) \leq f(S^G)$, $S^G \nrightarrow S$
15. Iterating the while loop, $argmin_{v \in S} deg_s(4) = 1$
16. $S \leftarrow \backslash\{4\}$, thus $S = \{5, 6, 7, 8, 9, 10\}$
17. Now, because $f(S) \leq f(S^G)$, $S^G \nrightarrow S$
18. 2 nodes have the same $argmin_{v \in S} deg_s(v)$ value of 2. we choose to remove a node arbitrarily between 7 and 5 19. In this case, $S \leftarrow \backslash\{5\}$, thus $S = \{6, 7, 8, 9, 10\}$
20. Now, because $f(S) \leq f(S^G)$, $S^G \nrightarrow S$

...

We iterate through the graph, until we have one node left.

Figure 2: Graph after first several iterations

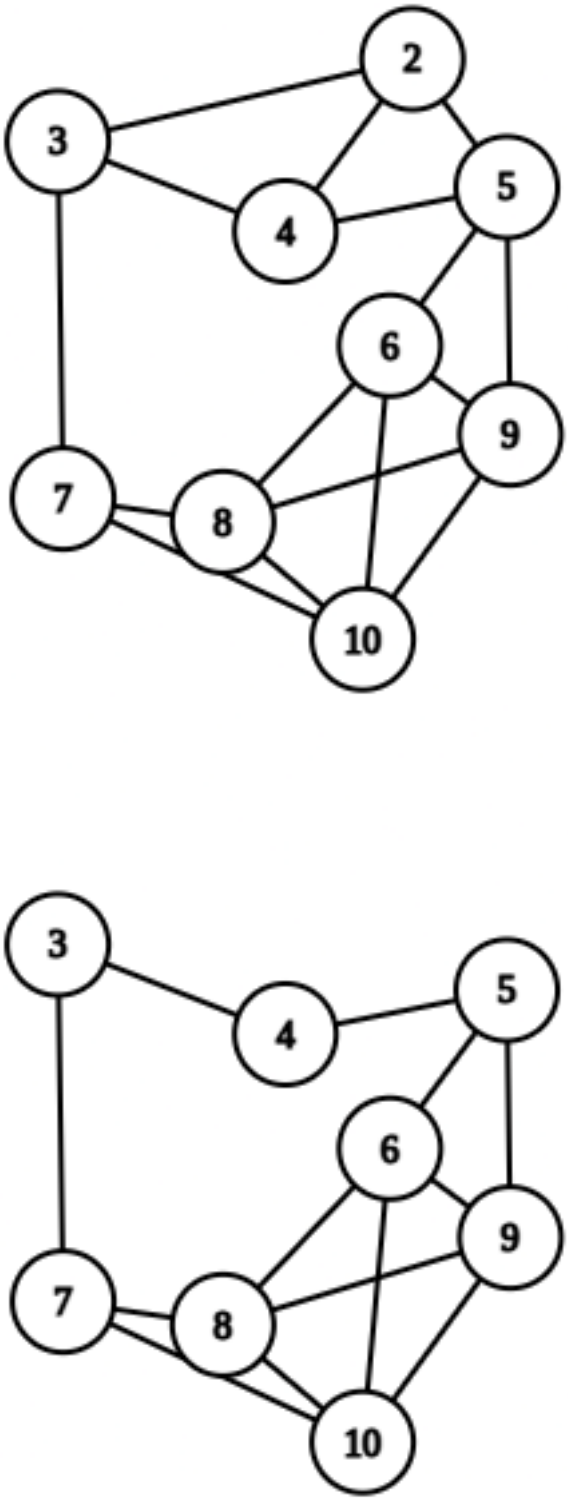
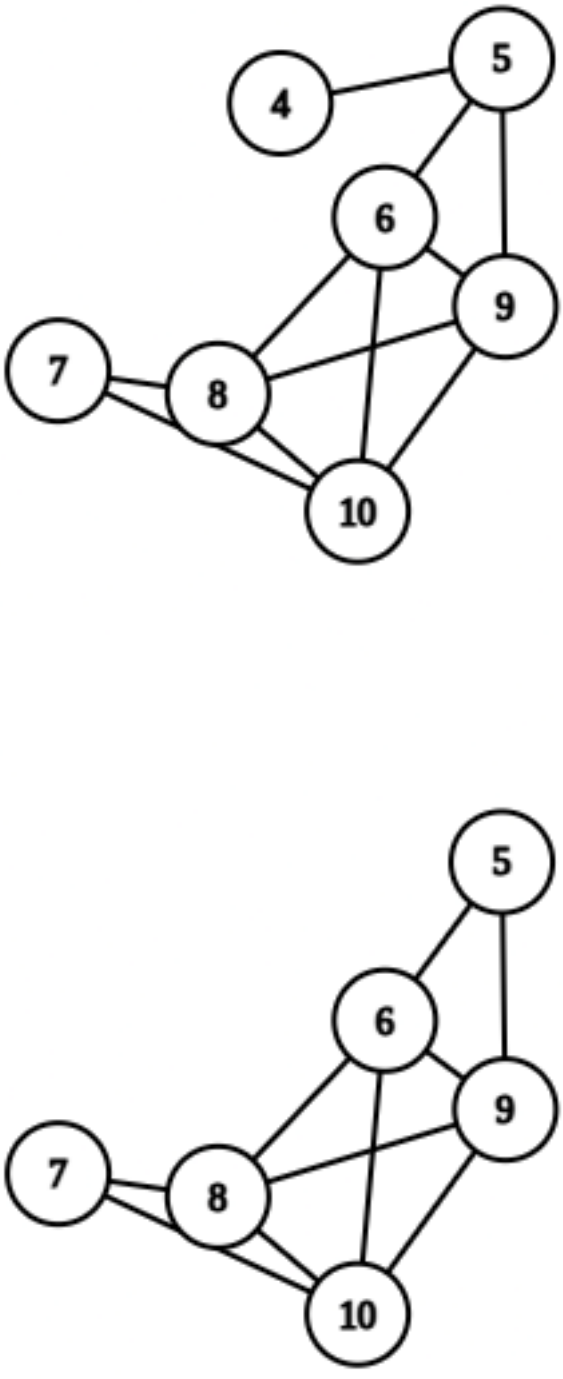


Figure 3: Graph after first several iterations



Problem 5.

5.1 Problem Statement

For this problem, I used the given Snap database to construct the GNN model, To perform a node classification. The database consists of the social media information of several 'egonode' users. This information consists of
$NodeId.edges$ - undirected edges between the given 2 nodes
$NodeId.circles$ - Circles for each ego node. there is a Maximum of 45 circles for the ego node. when we unite the data files, Circles are used as Labels. Not all nodes belong to circles and some nodes are in several circles.
$NodeId.feat$ - features of the user
$NodeId.featnames$ - names for the feature profile dimensions.

5.2 Methodology

I preprocessed the data to use it in the training. First of all, the program reads the edges from all "*.edges* files and loads the values in one array. After that, we do the same for the features. The code iterates over each "*.feat" files' in each line and retrieves the feature vector (all lines but the first value), loading it into 'the features array. Note, That in each file, the size of the feature array is different. Resulting in a jagged matrix. For this, we iterate over ".featnames" files and collect each existing feature. In our case, the length of the feature vector is 1283. We assign a feature vector to each node and populate them with 1, if the node has the correlating feature, otherwise, 0.

```
features = None

for node_id in node_ids:
    result = process_feat_files(data_dir, node_id)
    if features is None:
        features = result
    else:
        features = np.concatenate((features, result), axis=0)
```

The $process_feat_files$ function is defined as

```
    def process_feat_files(data_dir, node_id):
    feat_file_path = os.path.join(data_dir, f'{node_id}.feat')
    featname_file_path = os.path.join(data_dir, f'{node_id}.featnames')

    with open(feat_file_path, 'r') as feat_file,
```

```
    open(featname_file_path, 'r') as featname_file:
        featnames = featname_file.read().strip().split('\n')
        arrays = []

        for line in feat_file:
            line = line.strip()
            if line:
                feat = list(map(int, line.split()[1:]))
                selected_featnames = [featnames[i-1].split()[-1] for i,

                val in enumerate(feat, start=1) if val == 1]

                new_array = np.zeros(1283, dtype=int)
                for featname in selected_featnames:
                    new_array[int(featname)] = 1

                arrays.append(new_array)

        arrays = np.array(arrays)
        return arrays
```

As for labels, the program iterates through all "*.circles" files to retrieve the information regarding circles' groups. As a result, we get a dictionary consisting of the circle entries and the corresponding node numbers for each circle. From here, we can create a num.of.circles size vector for each node and populate them with 0 if the node doesn't belong to any circles. If it does, though, we populate the correlating place with 1 (if node i is in the nth circle, we populate the ith array with 1 in the nth place). We can use labels as is, or take a sum of each array, basically transforming a matrix into an array. Note, that we lose the information, but it is computationally cheaper.

```
    circles_files = [f for f in os.listdir(data_dir) if f.endswith('.circles')]
l = {}
for circles_file in circles_files:
    with open(os.path.join(data_dir, circles_file), 'r') as file:
        for line in file:
            circle = line.strip().split()
            circle_name = circle[0]
            circle_nodes = circle[1:]

            if circle_name in l:
                l[circle_name].extend(circle_nodes)
```

```
            else:
                l[circle_name] = circle_nodes
```

In the end, we get edges, features, and labels to work with, condensed in separate appropriate datatypes.

```
    labels = []
labels = np.sum(label_vectors, axis=1)

num_nodes = len(unique_nodes)
lb = np.zeros((num_nodes, 46), dtype=int)

for node_id, node_list in enumerate(l.values()):
    for circle_name, circle_nodes in l.items():
        circle_index = int(circle_name.replace("circle", "")) - 1
        if str(node_id + 1) in circle_nodes:
            lb[node_id][circle_index] = 1
```

To perform training, first, the GNN model is created with 5 convolutional and one output layer. The message-passing layers leverage ReLu activation and dropout functions. Adam is used as an optimizer. The data is split into training, validation, and testing sets using appropriate masks. We consider test accuracy, precision, recall, and F1 scores to evaluate the performance. The biggest gains in performance are seen when tweaking the Lables data type. Because of summation, a lot of information is lost. However, if we use the dictionary, or the matrix, the accuracy is more than doubled.

References

[1] Leslie Lamport. *LaTeX: a Document Preparation System.* Addison Wesley, Massachusetts, 2 edition, 1994.