**Computational Complexity**

**Social Network Analysis**

**Final Project**

**Name: Pavan Kumar Ghantasala**

**PUID: 0033740791**

**Table of Contents:**

1. Problem Description
2. Data Description
3. Graph Analysis and Algorithms
4. Information Flow Cost Minimization
5. Link Prediction and Network Growth
6. Conclusion
7. References
8. **Problem Description:**

Many real-world applications use or involve graphical networks in their models. Link Prediction or predicting the future connections between nodes is an important task for these applications. Social Networks, Online Marketplaces use link prediction to identify and target customers. Other applications include detecting collaborations between criminals and terrorist organizations. The retrieval of hidden connection will help in extracting vital information.

Network architecture can easily become complex with each addition of nodes and linkages between them. This makes graph analysis, traversal and identifying patterns among its constituents’ difficult problems. Using Graph Algorithms, Optimization using Integer Programming and Machine Learning to predict future connections, this attempt is to solve the three problems as graphs become more complex.

1. **Data Description**

In this project, a dataset of Facebook, a popular social network, is used. The data contains a list of nodes and list of connections between the nodes. Nodes are the Facebook pages of individuals or organizations in the network and connections are the mutual likes between the pages. The architecture of this design will help us design the data structure for analysis as an undirected graph with V vertices and E edges, G (V, E).

**Exploratory Data Analysis:**

Text

Description automatically generated

Nodes contain the Facebook pages data having the id and name of the page and the newly assigned id which represents the number of the node.

Graphical user interface, application

Description automatically generated

Edges data contain the number of the node and the nodes it is connected to as shown above. Using the nodes and edges data, graph network is created using the network package in python to analyze and understand the architecture of the graph. The resulting graph looks like the following.

A picture containing sky, tree, outdoor, snow

Description automatically generated

**Social Network with nodes and edges**

Text, letter

Description automatically generated

By analyzing the graph, we can see that it has 620 nodes and 2102 connections between the nodes. Network density is less suggesting not everyone is connected to each other. Top by nodes by degree and centrality gives consistent results hinting most connected nodes are important in the network.

A picture containing sky, wire

Description automatically generated

**Spring Layout Chart**

This diagram shows the spring layout form of the anti-gravity simulation of the original graph. This is a force-directed representation of the network treating edges as springs holding nodes close, while treating nodes as repelling objects, sometimes called an anti-gravity force. Simulation continues until the positions are close to an equilibrium.

1. **Graph Analysis and Algorithms:**

As the data are represented in an undirected graph, we can use graph algorithms like Breadth First Search and Depth First Search to find search and sort the entire graph. Developing an effective algorithm for this is vital as graphs become bigger and deeper.

For example, running the Breadth First Search algorithm in our dataset from initial ‘0’ node gives the following output.

Table

Description automatically generated

It gives the level of the child node from the initial node from where we are searching. This gives insights about closest neighbors and farthest connections. For example, Node ‘132’ is an immediate neighbor to the Node ‘0’ whereas node ‘182’ is 603 levels away from the node ‘0’.

Breadth First Search Algorithm also has wide range of applications including finding the shortest path in navigation, cycle detection in undirected graphs, minimum spanning tree and ford Fulkerson algorithm to find maximum flow in a network.

We have also used breadth first search to determine number of connected components in this graph dataset of Facebook and data of interest is a fully connected graph giving the total number of components to be 1.

1. **Information Flow Cost:**

Information Flow is an important problem in graphical networks, the vital part of which is identifying the critical nodes to pass the information such that it flows through the entire network. It can have varied applications in advertising, media, entertainment, and in online marketplaces.

To solve this problem, Minimum Vertex Cover algorithm gives the parallels for solving this problem. Minimum Vertex Cover problem identifies the nodes which covers all the edges in the network. So, to maximize information flow, our hypothesis is to minimize the cost (the number of nodes to traverse) assuming all the edges have equal cost (weights).

We can solve this minimum vertex cover problem using integer programming in the following manner.

Given an undirected graph G (V, E) which has |V| vertices and |E| edges and suppose wi be the weight of each edge, the minimum vertex cover problem can be formulated as following.

Text, letter

Description automatically generated

Where yi is a binary variable having 1 or 0 value if the vertex is selected or not. In this scenario, the weight is assumed to be 1 i.e., all the edges have equal cost of information flow. In this case our objective is to find those minimum nodes that will cover the entire graph.

Graphical user interface, text, application

Description automatically generated

The above picture is a partial output for minimum vertex graph. We can see that the size of the minimum vertex cover is 286. This indicates that to maximize the information flow or to reach the complete network, these 286 nodes out of total 620 nodes need to be targeted. This is a valuable insight for targeting and segmentation teams trying to spread the information among the networks.

Thus, to maximize the information flow in the network, we can convert the maximization problem as minimizing cost problem and assuming the equal weight to each edge we can say that covering the 286 nodes will complete all the edges in the network.

1. **Link Prediction:**

In network theory, link prediction is the problem of predicting the existence of a link between two entities in a network. It has many applications including predicting the friendship links among users in social network, gene-protein interactions, recommendation systems in retail, and predicting co-authorship relationships in citation networks.

In this case, our problem can be thought of having a temporal aspect. Given a snapshot of the set of links at a given time t, the goal is to predict the links at time t+1 or in general (t+n). There are many methods of predicting links including supervised approaches on graphical models and deep learning and unsupervised approaches based on similarity measures computed on random walk and matrix factorization.

Graph Embedding are simple yet powerful and offer a convenient way to predict links. Node2vec is one such module that learns a embedding space in which neighboring nodes are represented as vectors. Once the nodes are represented as vectors, we can use machine learning techniques taught in class to predict edges based on similarity.

**Hypothesis**:

Key Hypothesis is that given a core structure of the graph, Number of Connected Components, the network grows in a sequential manner and future connections between unconnected nodes can be predicted using the redundant links that are present in the data without changing the core structure.

**Approach:**

Proving this Hypothesis include two key components of the problem.

1. Extracting the redundant links in the graph which do not change the structure.
2. Preparing a train and test dataset from the set of connected and unconnected nodes to train and validate the model.

**Data Preparation and Model Development:**

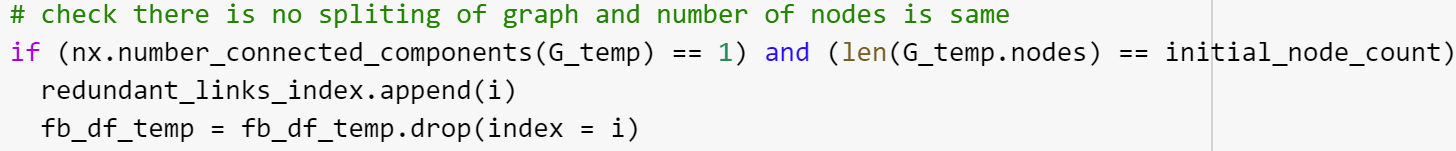
To perform the above two steps, we need to obtain unconnected nodes from the graphs. For this, we can use the adjacency matrix representations of the graphs explained in the class.

Calendar

Description automatically generated

**Adjacency Matrix**

In adjacency matrix, if a value is 0 then the node in row and node in column are unconnected. Combination of all these pairs forms the unconnected pairs of the graph.



We find the redundant links based on removing each link in the graph and checking if the core structure (number of connected components) has changed and if total number of nodes is present.

Graphical user interface, text, application

Description automatically generated

After data preparation, the data contains 19,018 pairs of unconnected nodes and 2966 pairs of connected nodes which makes it reasonable for applying machine learning models.

Here we want to find whether there is a future connection between two unconnected nodes. This is a classification problem which is a supervised machine learning technique. After applying Node2Vec module and vector representation of the nodes, four machine learning algorithms are applied.

1. Logistic Regression
2. Support Vector Machines
3. Decision Tree
4. Neural Network
5. **Results:**

Following are the ROC AUC curves for four different models applied. We can see that ROC AUC curve performs better than the remaining three models with SVM Classifier not performing better than a random guess classifier.

Chart, line chart

Description automatically generated Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated Chart, line chart

Description automatically generated

**ROC Curves of different Models**

|  |  |  |
| --- | --- | --- |
| **Algorithm** | **ROC AUC Score** | **Accuracy** |
| Logistic Regression | 0.7869 | 72.8% |
| Decision Tree Classification | 0.6892 | 93.49% |
| Support Vector Machine Classification | 0.5165 | 93.30% |
| Neural Network | 0.7292 | 94.64% |

We can also see the accuracy scores of all the classifiers. From the results we can say that although the accuracy scores of SVM, Decision tree and Neural Network is high, their predictive power is very less as compared to ROC AUC score. For classification models, ROC AUC score is robust metric and gives a clear picture of predictions at different levels of threshold.

Our Hypothesis that redundant connections, which form the basis for network growth, can be used as training for prediction of future connections holds true.

1. **Conclusion:**

Graph Networks have great potential in solving problems and applications in real world problems. In this project we have used techniques discussed in computational complexity, namely Graph Algorithms, Linear and Integer Programming and Machine Learning, to apply for social network connections data.

* Breadth First Search is useful in identified the closest connected neighbors and identify the shortest path of the graph
* Number of Connected Components algorithms reveals that the graph is fully connected
* Information flow problem can be solved using formulation of Minimum Vertex Cover Algorithm and identify the important nodes in the network using Integer Programming
* Adjacency Matrix is used to identify the architecture of the graph and retrieve the unconnected and redundant connections from the graph
* Machine Learning techniques are used to predict future connections in the graph given a set of connections proving out hypothesis.

**References:**

[**https://networkrepository.com/fb-pages-food.php**](https://networkrepository.com/fb-pages-food.php)

[**https://arxiv.org/pdf/1507.05739.pdf**](https://arxiv.org/pdf/1507.05739.pdf)

[**https://www.kaggle.com/roshansharma/facebook-social-network-analysis**](https://www.kaggle.com/roshansharma/facebook-social-network-analysis)

[**https://en.wikipedia.org/wiki/Link\_prediction**](https://en.wikipedia.org/wiki/Link_prediction)