MA214 Network Analysis Group Assignment

Group report on Analysis of Co-purchasing Network

Team Members

Praveen Kumar Neranike (2310926)

Jasper Varun Thinagaraj (2310238)

Anris Roque Da Costa (2312127)

Bathala Nagadithya (2310244)

Rishwanth Mithra (2311566)

School of Mathematics, Statistics and Actuarial Sciences

University of Essex, UK.

ABSTRACT

A supply chain is a network of linked stages that coordinate to convert raw materials into completed products and deliver those products to the buyers. Firms need effective supply chain management to optimize operations, reduce costs, and enhance product quality, which plays a critical role in meeting customers' demands and keeping up with the market dynamics. However, in a standard supply chain, management faces some common issues, such as high costs, supplier dependence, lack of new markets, and varying demands. To address similar problems, there are multiple techniques to overcome these issues.

Co-purchasing is when individuals band together to make joint purchases. This collaborative approach leverages consumers with collective buying power, cost savings, exclusive deals, increased bargaining margins and saves time. Consider a network of four products A, B, C, and D, where D is the parent node, A is the sub-parent node, and B and C are the child nodes. This implies that the users purchased products B and C together with item A and purchased products A and C with item D. This makes product C a co-purchasing of items A and D.

Multiple strategies have been put forth with a co-purchasing network as a base in network analysis. One is where a recommender system is built using centrality measurement based on multi-objective optimization problems and Pareto optimal recommendations, influencing others' purchasing decisions [1]. The second one is based on the ERGM exponential random graph model which elaborates the product involvement based on timestamp and visit depth through which one can reveal how the recommendation network affects product involvement [2]. The third one is based on the co-purchase network, where the network is analyzed using a machine learning models like linear SVM binary classifier considering different structural properties to identify how a particular node (product) is positioned in this network [3]. They have analyzed the nature and occurrences of motifs present in a network, which can be used along with other community detection techniques to analyze the co-purchase network efficiently and gain insights out of it [4]. The authors have analyzed the co-purchasing network. They have drawn some central entities from the frequent item sets and grouped them from the network using Clauset-Newman_Moore (CNM) [5] method and Jaccard co-efficient to find the relation between the products and boost the sales of the chosen items.

Keywords: Supply Chain Management, Co-purchasing Networks, Network Analysis, Machine Learning, Communities.

---

1. Literature Review

This research paper focuses on a recommender system based on multi-objective optimization problem and pareto optimal recommendations concerning rating and rankings of the products. This research paper [1] also involves developing a product co-purchasing network for the recommender system using centrality measurement which influences other's purchasing

decisions. A model-based recommender system – A behavioral user model is created using clustering approach on user's preferences. This technique helps with recommendations by understanding and clustering users with similar preferences. This process involves analyzing the history of item co-purchases as a directed graph and applying topological analysis for item centrality. This process evaluates predicted ratings and item rankings to recommend optimal products to interested users. The algorithm used in this technique is hierarchical clustering with an agglomerative approach while creating the model. Hierarchical clustering with an agglomerative approach creates a behavioral user model. The products recommended are identified using the Pareto optimal concept. The above results imply that closeness centrality leads to better recommendations than Degree Centrality. The items recommended satisfies the stated goals highlights how successful the strategy is in optimizing item recommendations.

According to the research paper [2] recommendation networks are used in many economic sectors that allow users to find their choices based on certain reactions. Therefore, it is considered that the co-purchase recommendation system is the main cause of harm to the sale diversity. Also, the data consists of individual interactions from the users which are namely those individuals who have clicked, tagged, and added the product to the cart while also those who removed it from the cart and those who purchased the product. An important piece of information is by creating a directed adjacency matrix, one can conclude that the diagonal entries are all zeros and entries can be zeros and ones where zero means that repurchase behavior was excluded. From the research paper, the measurement for outer node attributes is based on product involvement which states that the product is revealed by seeking, evaluating, the exponential random graph model from the research paper explains the statistical inference. One can cypher out from this research paper that it helps in understanding the economic values of the co-purchase network which includes the endogenous co-purchase relationship and exogenous product involvement.

Researchers in this paper [3] built a co-purchase network treating the individual products as nodes, with edges in between if two products were bought with one another. Network-assisted sales rank is determined by analyzing a co-purchase network using "also-bought" metadata. This method is more effective in classifying a popular product and the properties such as Clustering Coefficient, Betweenness, Closeness, Eigenvector, PageRank, and Community Membership can distinguish a popular product from an unpopular one. Authors used a linear SVM binary classifier to predict product popularity based on network features. They trained on pre-2013 products and predicted sales rank of new products launched in 2013. A product's popularity can be determined by its position in a co-purchase network. They proposed a network-based classifier that performs better than the baseline by using co-purchase count, reviews, and rating features. In the future, they aim to apply network-based classifier method to other e-commerce platforms and explore its potential for studying market characteristics like customer behavior, seller behavior, and sales speed.

In this paper [4] they have analyzed the Amazon co-purchase network dataset to understand the significance of nodes with high in-degree and high out-degree and they have subsequently analyzed the evolution of communities in the network. They have made use of the Motifs technique in this work on the product co-purchasing network to gain insights into relations between products. Community structures have also been used in this paper to identify different communities and the various products preferred by these communities. Analysis of 3 node motif in network and analysis of 4 node motif in network were performed in this work and conclusions were derived on which motif ID has the most occurrences and by seeing that they

have predicted and analyzed the pattern in which customer buys products. They have also discussed community detection algorithms such as the Girvan-Neuman algorithm and modularity maximization method. The conclusions drawn from this paper are that motifs alone cannot be used to efficiently analyze the network, they must be combined with any other community detection method for efficient analysis of the network and with respect to the co purchase network they have been able to draw the following conclusions just by making use of motifs, prediction of product demand, purchasing trends in product categories and relations between different products. The future works suggested in this paper is to analyze the motifs and find out different patterns present in the motif which can be later used to effectively improve the marketing and advertising of various products by a retailing company to increase their sales and profit more deeply.

In this paper [5] researchers studied network data to understand the significance of nodes with high in-degree and high out-degree. The chosen dataset consists of a network and its snapshots over time intervals which helped to understand how network changes over time and finding the co-purchasing patterns. The important aspect of analyzing dynamic networks is to track the evolution of communities. To track the evolution and to find the disjoint community structures, they have used Clauset-Newman_Moore method. Correlation between all the communities of one timestamp is calculated from the subsequent timestamp to understand whether a node in specific community is same at different time stamps. They made use of Jaccard co-efficient to measure how similar two communities are from two different time stamps. Frequent item sets based on the sub-category have been found by making use of FP-Growth Algorithm. They concluded that finding the association between the nodes will help to build a prediction model where if a pattern is frequent in ‘n-1 ’time stamps, then there is a possibility that it will repeat in nth time stamp. In future work, they stated that we can further use the Markov chain model for implementing a generalized recommendation system to build a prediction model.

2. Network Data Analysis

2.1. Source and Description of Dataset

We gathered the Amazon product co-purchasing network dataset from the following website https://snap.stanford.edu/data/com-Amazon.html. It is based on a feature where a product is co-purchased with another. The Amazon product co-purchasing network dataset is a rich and complex dataset that offers significant information on the behaviors of customers' buying patterns on Amazon's website. The network has 334863 nodes and 925872 edges where we can perform analysis of network data.

2.2. Preprocessing

Initially, we loaded the downloaded dataset text file to Jupyter Notebook (Python IDE) and converted the text file to a CSV file for analysis purposes. We removed the first four records as they were header information. Ideally, the dataset nodes were concatenated in a single column, and the data is split into two columns. Later, the concatenated column is dropped. We have taken a subset of 10,000 records to reduce the computational complexity and stored it in

a CSV file. The analysis is carried out with the CSV file having a subset of records. We have drawn the graph showing the subset of records having 11,468 nodes and 10,000 edges. The image of the co-purchasing network is shown below.

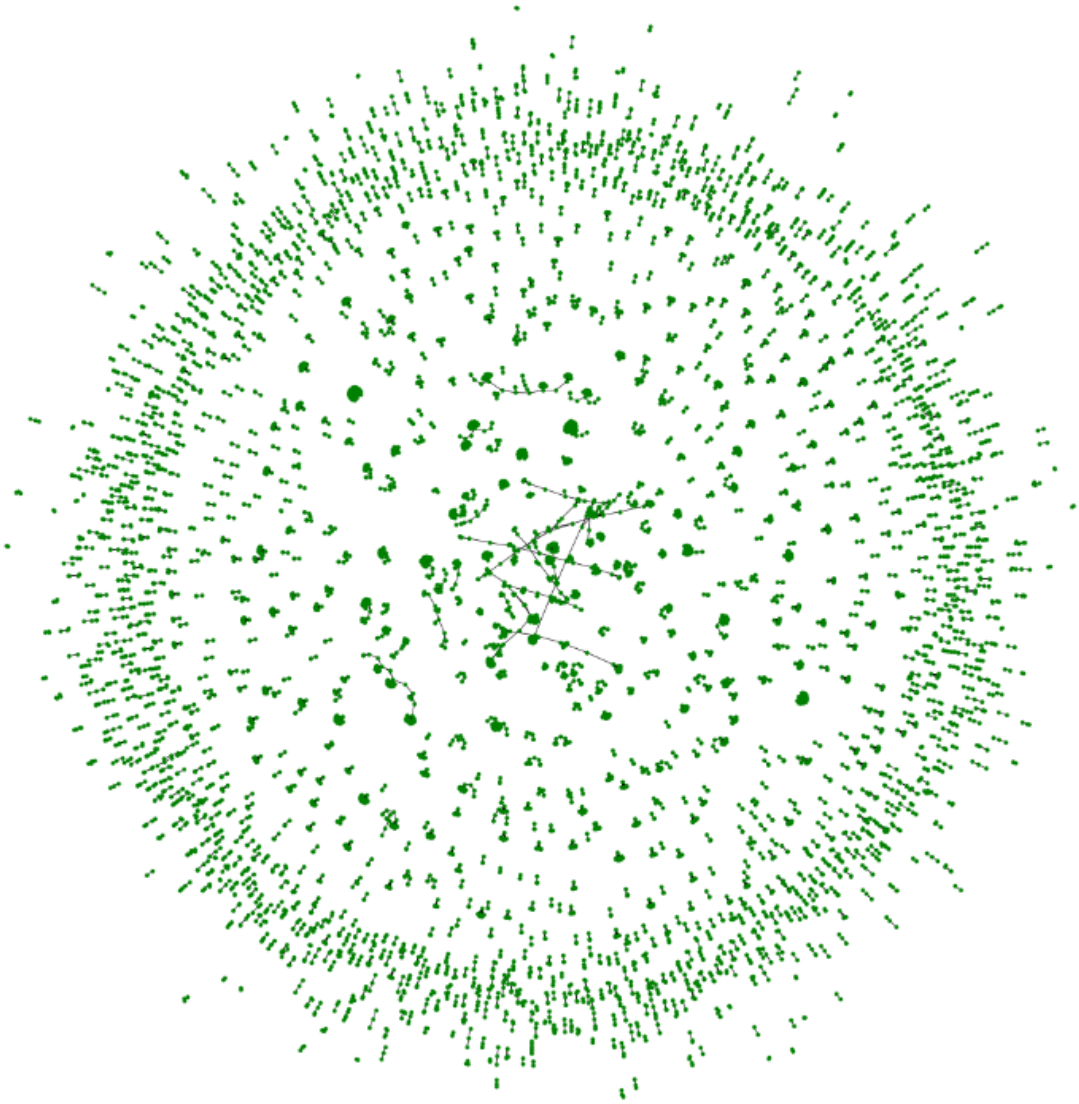

Figure 1: *Co-purchasing Network.*

2.3. Analyzing Network

We draw the edges from one node to another by iterating the dataset one row at a time, and then we print the different nodes present in the network and the edges formed. We are finding the adjacency matrix of the product co-purchase network using the adjacency matrix function, which is present in the network library. The adjacency matrix obtained is of the shape [11468 x 11468]. In the adjacency matrix A, Aij = 1, when there is an edge between node 'I' to node 'j', else Aij = 0, when there is no connection between the nodes 'I' and 'j'.

From the constructed network, we randomly chose two nodes and checked if there was an edge between the two nodes by making use of the has path function present in the network library; this function returns true; if there is an edge between the nodes else, it returns false. Similarly, by using the shortest_path function, we have found the shortest path from the randomly selected node from the network to all other nodes connected to that node. From the co-purchased network, we found the shortest path from node 500 as a source node to all other nodes connected to node 500 and obtained the below results.

```
Shortest paths from source node 500 :

{500: [500],
 94156: [500, 94156],
 179049: [500, 179049],
 377514: [500, 377514],
 468226: [500, 468226]}
```

Figure 2: *Shortest paths from a source node*

From the above result, we can see four nodes connected to node 500, and the shortest paths between node 500 and the other four nodes are as displayed above. Similarly, we have selected a random node from the network and found the length of the shortest path to that node as a target node from all other connected nodes. And the results are shown below:

```
Shortest paths to target node 447165 :

{447165: [447165],
 1: [1, 447165],
 88160: [88160, 1, 447165],
 118052: [118052, 1, 447165],
 161555: [161555, 1, 447165],
 244916: [244916, 1, 447165],
 346495: [346495, 1, 447165],
 444232: [444232, 1, 447165],
 500600: [500600, 1, 447165]}
```

Figure 3: *Shortest paths to a target node*

From the above result, we can see that the shortest path length from node 1 to node 1 itself is zero, and from node 1 to node 88160 is 1. We have used the is_connected function to check if the network is connected, but the function returned false as the co-purchased network is not connected. The number of connected components present in the network is 1624 which is determined by using the number_connected_components function.

The density of a network is the ratio of the actual connections in the network to the maximum number of connections. The density of the co-purchased network is found to be 0.00015209, which is near zero, which indicates that the network is not a complete graph. The degree of a node indicates the number of edges connected to that node, the degree of each node in the co-purchase network is found out, The maximum degree of a node in the network is 72, and the mean degree is found as well, the mean degree of the product co-purchase network is 2, and a frequency distribution of the degrees of the product co-purchase network is as shown below.

<Axes: xlabel='degree'>

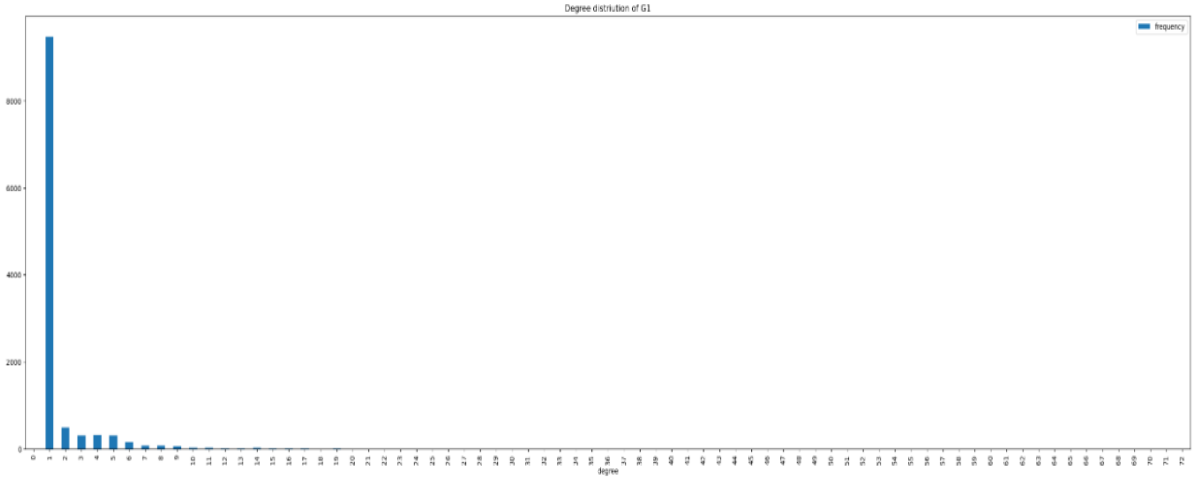

Figure 4: *Frequency Distribution Plot of Degree*

From the above frequency distribution plot, we can conclude that 9477 nodes have a degree of 1, 488 nodes have a degree of 2, and one node has a degree of 72.

2.4. Closeness centrality, Betweenness Centrality, K-Cores and K-Components:

The closeness centrality function indicates how close all other nodes are to a particular node which is connected. Considering the co-purchase network, most nodes have a closeness centrality near zero. The betweenness centrality measure evaluates a node's position in the network by analyzing how frequently it serves as a bridge on the shortest path between two other nodes. Considering the co-purchase network, most nodes have a betweenness centrality near zero. This indicates that most nodes do not act as a bridge between the other nodes. The number of cliques which are present in the network is 9875. The minimum clique size of this network is 1; this was found by making use of the enumerate_all_cliques function. The function find_cliques return an iterator over cliques, each of which is a list of nodes.

For dividing the network into groups, we are making use of another technique called k cores, which divides the network based on k core values. We got the k values 1, 2, 3 and 4 in the network we analyzed. In the first core, there are 11468 nodes; in the second core, there are 330 nodes; in the third core, there are 16 nodes; and in the fourth core, which is the main core, there are 5 nodes. The nodes which are present in the main core are 862, 865, 866, 868, 152810. By making use of the k_components function, we found out the different components that are present in each core and, we found out the main core, which is 4 and the nodes that are present in the main core. For example, in this particular co-purchasing dataset, in core 1, one of the components which are present in it is {1, 88160, 118052, 161555, 244916, 346495, 444232, 447165, 500600}.

2.5. Community Detection

Community Detection using greedy modularity optimization:

We have made use of modularity to measure the strength of the division of networks into groups. The high modularity value indicates strong connection between the nodes present in a group and weak connection between nodes present in different groups and the low modularity value indicates weak connection between the nodes present in a group and strong connection between nodes present in different groups. The modularity value obtained for the co-purchase network used in this project is 0.9985 which indicates that the groups are divided strongly and there is a very strong connection between the nodes present within each group and very weak connection between nodes present in different groups. The number of groups obtained from the network are 1624 which is having 11468 nodes and the different groups obtained are shown below:

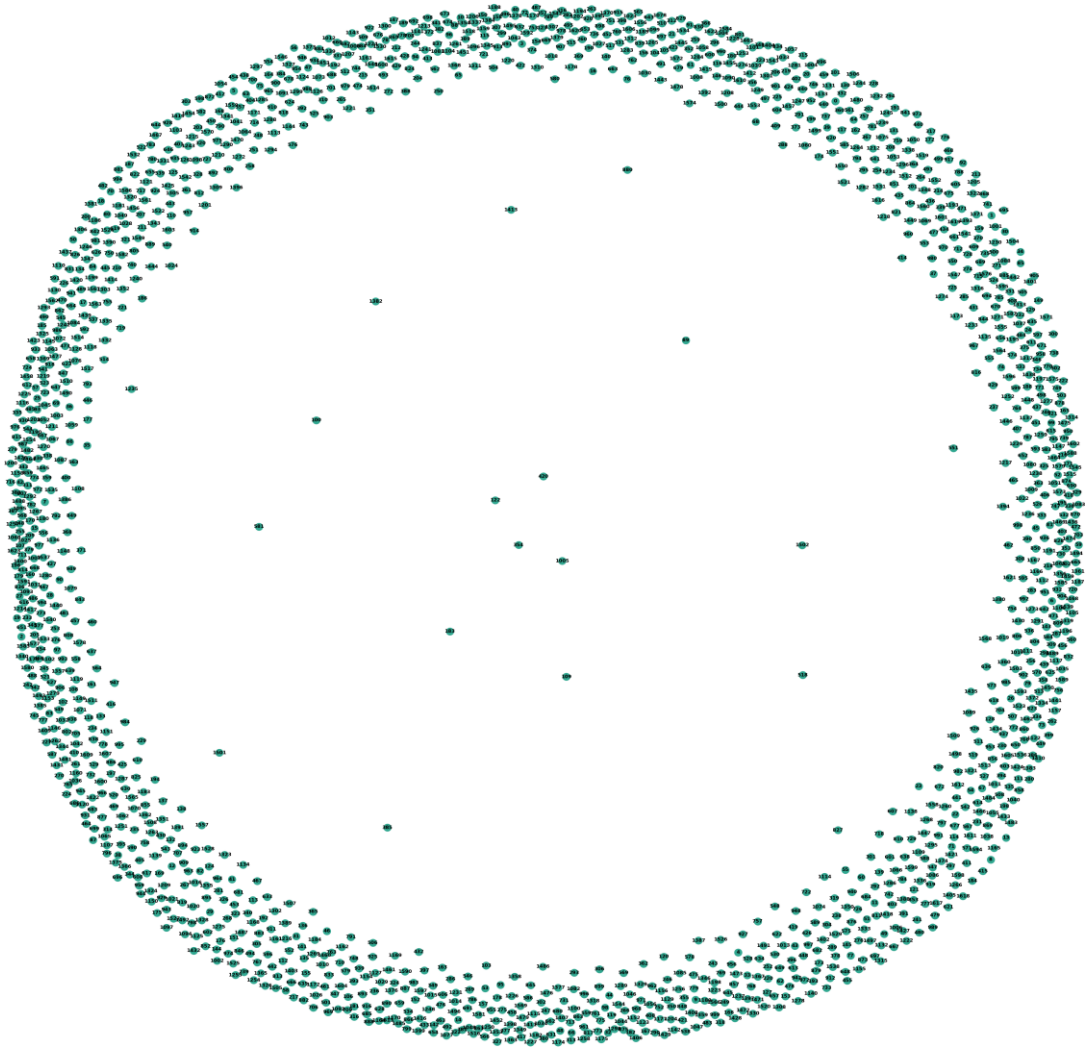

Figure 5: *Different Communities in the Network*

Community Detection using Girvan-Newman Algorithm:

The Girvan-Newman Algorithm is a technique used to detect community; it works by iteratively removing edges with high betweenness centrality. Number of Communities detected using Girvan-Newman algorithm is 1654. The steps involved in the algorithm:

- The edge betweenness is calculated for each edge in the network.
- The edge with the highest betweenness is removed.
- Edge betweenness is calculated for edges affected in step2.
- Repeat from Step until no edges are left.

Community Detection using InfoMap Algorithm:

Additionally, we made use of InfoMap Algorithm. It is a technique used to detect communities based on information flow. This algorithm works on the technique of random walk. Steps included in this algorithm are an initial node is selected and then we move one step along the edge which is randomly chosen from among the edges connected to that node and this process is repeated. The number of Communities detected by InfoMap Algorithm is 1692.

2.6. Visualizing Communities

A random community was selected from the network and the nodes present in the community are [2215, 223703, 413504, 540136]. On analyzing this community, we found that the following edge connections present in this community are [(2215, 223703), (2215, 413504), (2215, 540136)]. Below figure visually represents how nodes were connected to each other.

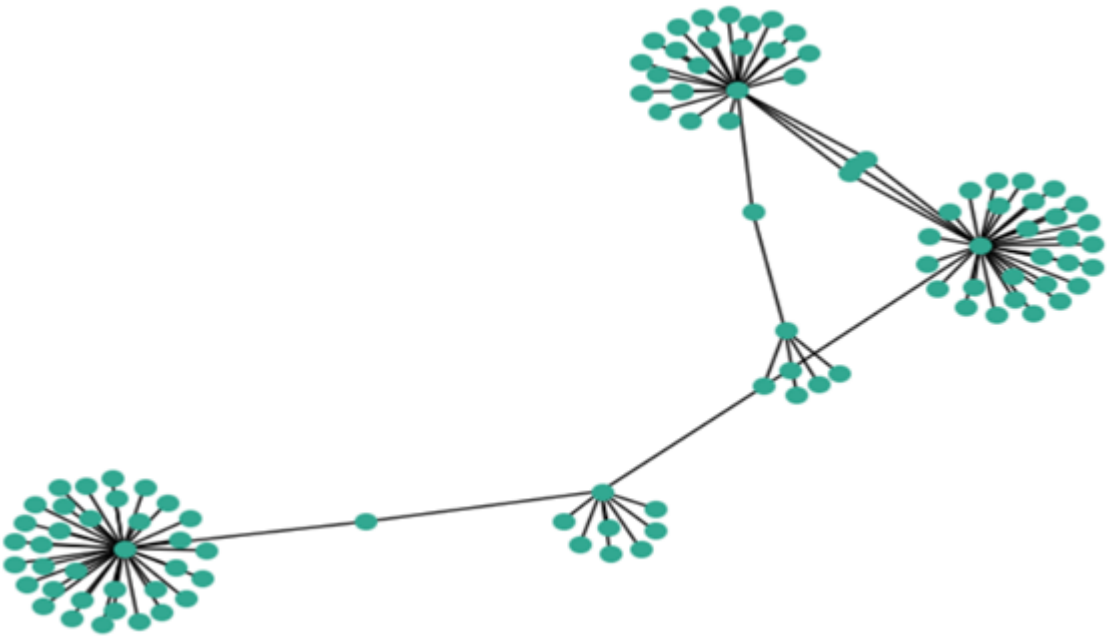

Figure 6: *Community of a Network*

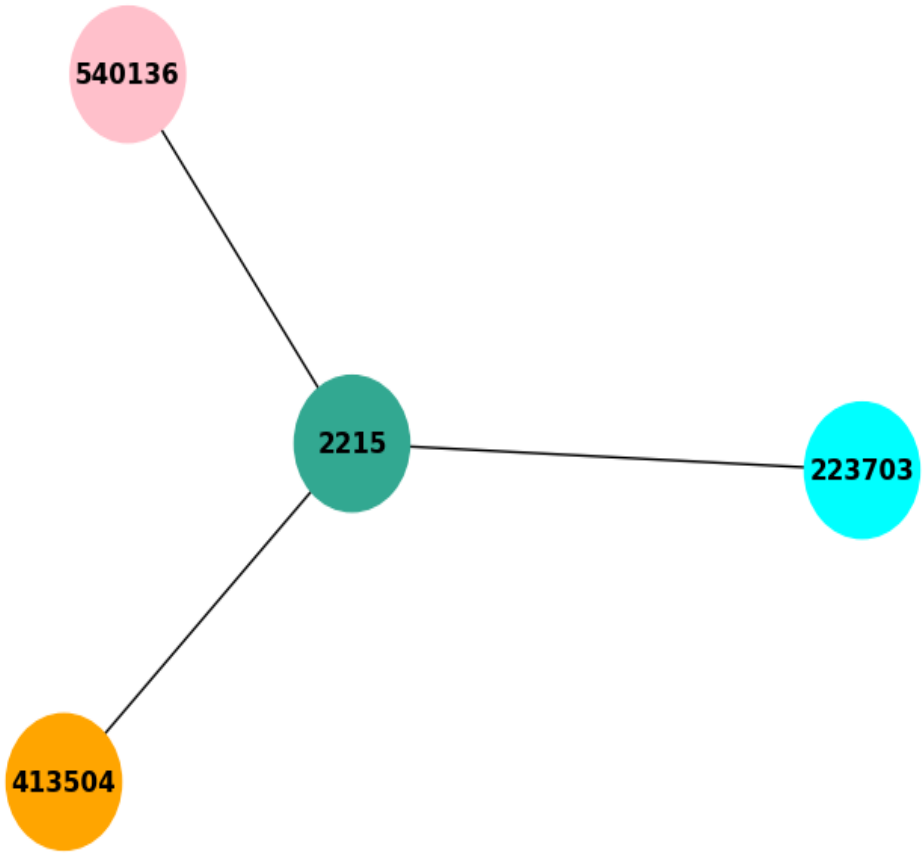

Figure 7: *Second Community of a Network*

If we consider each node as individual products, then if a consumer wants to buy product 2215 then based this community Products 540136, 223703, 413502 are recommended to consumer as most consumers who bought product 2215 also purchased either of the three products along with it and if the consumer wants to buy product 540136 then it is most likely he would buy product 2215 and 223703 or product 2215 and 413504 or both.

Above considered sample community consists of only four nodes, we have considered a smaller sample for ease of analysis. The Largest community obtained by our analysis of Product Co-Purchase Network Dataset consists of 73 Nodes. On analyzing larger communities, we will be able to obtain broader insights on which product is frequently co-purchased with another product and same can suggested to the consumer which would in-tern raise sales of the particulars products and help in generating more significant revenue.

3. Conclusion

The Aim of the research on Product Co-Purchase Network Data is to apply various community Detection Algorithm or Method on the Network Dataset and to find the internal Communities present within the entire Network. In this work we have successfully implemented Cliques, K-Core, K-Component, Modularity Maximization, Girvan-Newman Algorithm and InfoMap Algorithm to detect communities present in the Product Co-Purchase Network Dataset. Following were the findings the number of cliques which are present in the network is 9875 and for K-Core, we got the k values 1, 2, 3 and 4 in the network we analyzed. In the first core, there are 11468 nodes; in the second core, there are 330 nodes; in the third core, there are 16 nodes; and in the fourth core, which is the main core, there are 5 nodes.

In K-Component we found out the different components that are present in each core and also we found out the main core, which is 4 and the nodes that are present in the main core are 5 and in Modularity Maximization the modularity value obtained for the co-purchase network used in this project is 0.9985 and communities detected were 1624. InfoMap Algorithm detected 1692 Communities. In the future we can expand this work by taking more complex Dataset and implementing more efficient community detection algorithms like Louvain Modularity, Walk trap Algorithm and Fast Greedy Algorithm which would lead to more enhanced detection of the communities in a Network.

4. References

[1] Sunantha Sodsee and Maytiyanin Komkhao, “Item Recommending by Item Co-purchasing Network and User Preference”, The Twelfth International Conference on Digital Information Management (ICDIM 2017) September 12- 14, 2017, Kyushu University, Fukuoka, Japan.

[2] Hongming Gao, Hongwei Liu, Minqi Yi, “Inferring values of recommendation links: Analysis of co-purchase network based on ERGM and product involvement”, 2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE 2021), 2021.

[3] Utpal Prasad, Nikky Kumari, Niloy Ganguly & Animesh Mukherjee, “Analysis of the Co-purchase Network of Products to Predict Amazon Sales-Rank”, International Conference on Big Data Analytics BDA, 2017.

[4] Abhishek Srivastava, “Motif Analysis in the Amazon Product Co-Purchasing Network”, COMS6998- Network Theory- HW3, Computer Science Department, Columbia University, 2010.

[5] Partha Basuchowdhuri, Manoj Kumar Shekhawat, Sanjoy Kumar Saha, “Analysis of Product Purchase Patterns in a Co-purchase Network”, Fourth International Conference of Emerging Applications of Information Technology, 2014.

5. Appendix

```
#Importing libraries
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import networkx.algorithms.community as nx_comm
import matplotlib.colors as mcolors
import statistics as st
import cdlib
from cdlib import algorithms, readwrite
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = 'all'
```

```
# Cleaning Data
#Loading dataframe into a variable
df = pd.read_csv('com-amazon.ungraph.txt', header = None)
df.head()
#Ingoring first four rows of unwanted data
df = df.iloc[4:]
df.head()
#Data Info
df.info()
#Splitting the data column into two separate columns i.e., Node1 and Node2
df[["Node1", "Node2"]] = df[0].str.split("\t", expand = True)
df.head()
#Dropping the unwanted column
df.drop(0,axis=1, inplace = True)
```

```
df.head()
#Writing the processed data to csv file
df.to_csv("co_purchase_data.csv", index = None)
#Reading data
data = pd.read_csv("co_purchase_data.csv")
data.head()
#Subset of the dataset
data_subset = data[:10000]
#Shape
print("Shape of chosen subset : ",data_subset.shape)
data_subset.head()
#Writing subset to csv file
data_subset.to_csv("co_purchased_data_subset_10000.csv", index = None)
# Visualizing the Network
#Reading subset data to variable
data=pd.read_csv("co_purchased_data_subset_10000.csv")
#Plotting graph
plt.figure(figsize=(25,25))
G1=nx.Graph()
G1.add_edges_from([(row['Node1'],row['Node2']) for i,row in data.iterrows()])
nx.draw(G1, with_labels = False, node_size = 25, node_color = 'Green', font_size = 10)
```

Analyzing the Network

```
#Nodes and Edges
print("Number of Nodes in the network : ", G1.number_of_nodes())
print("Number of Edges in the network : ", G1.number_of_edges())
#List of nodes and edges
print("List of Nodes : ",list(G1.nodes))
print("List of Edges : ",list(G1.edges))
```

```
#Adjacency matrix
Adj_mat = nx.adjacency_matrix(G1)
print("ADJACENCY MATRIX (A) : \n\n",Adj_mat.todense())
#Adjacency matrix shape
print("Shape of adjacency matrix : ",Adj_mat.shape)
#Checking if path exists
nx.has_path(G1, 10, 80558)
nx.has_path(G1, 500, 168895)
#Checking for the shortest paths from a given source node
print("Shortest paths from source node 1 :")
nx.shortest_path(G1, source = 1)
print("\nShortest paths from source node 500 : ")
nx.shortest_path(G1, source = 500)
#Checking for the shortest paths to a given target node
print("Shortest paths to target node 94156 : ")
nx.shortest_path(G1, target = 94156)
print("\nShortest paths to target node 447165 : ")
nx.shortest_path(G1, target = 447165)
#Shortest paths for each node
print("Shortest paths for each node in the network : ")
nx.shortest_path(G1)
#Length of the shortest paths for each node
print("Length of the shortest paths for each node : ")
dict(nx.shortest_path_length(G1))
#Checking if the network is connected
print(nx.is_connected(G1))
#Checking if the network is planar network
print(nx.check_planarity(G1)[0])
#Finding number of connected components in the network
print("Number of components in the network : ",nx.number_connected_components(G1))
```

```
#Density of the network
print("Density of the network : ",nx.density(G1))
#Degree of each node in the network
d1 = G1.degree()
print("Degree of each node in the network : \n",d1)
#Mean degree of the network
n1=len(list(G1.edges))
mean_degree = sum(dict(d1).values())/n1
print("Mean Degree of the network : ",mean_degree)
#Finding the unique degree values and its count
degree_sequence_G1=sorted([d for n, d in G1.degree()], reverse=True)
degree_unique_G1=np.unique(degree_sequence_G1, return_counts=True)
print("Unique degree values and its count : \n",degree_unique_G1)
#Plotting degree frequency distribution
degree_plot=list(range(0, max(degree_unique_G1[0])+1))
freq_plot=[0] * len(list(range(0, max(degree_unique_G1[0])+1)))
for i in degree_unique_G1[0]:
    freq_plot[i]=degree_unique_G1[-1][list(degree_unique_G1[0]).index(i)]
d = {'degree': degree_plot,
     'frequency': freq_plot}
df_G1 = pd.DataFrame(data=d)
fig, axes = plt.subplots(figsize=(35,10),nrows=1, ncols=1)
df_G1.plot.bar(x='degree', y='frequency', ax=axes)
axes.title.set_text("Degree distriution of G1")
#Closeness centrality of the network
print("Closeness Centrality : ")
nx.closeness_centrality(G1)
#Betweenness Centrality of the network
print("Betweenness Centrality : ")
nx.betweenness_centrality(G1)
```

```
#Enumerate all cliques
enumerate_cliques = list(nx.enumerate_all_cliques(G1))
enumerate_cliques
#Finding Cliques
print("Number of cliques : ",len(list(nx.find_cliques(G1))))
Cliques=list(nx.find_cliques(G1))
print("Cliques in network : ",Cliques)
#Finding code number of each node in the network
print("Core number of each node : ")
nx.core_number(G1)
#Nodes in the Main Core of the network
print("Nodes in the main core : ",list(nx.k_core(G1)))
#Main component and Nodes in the main component
k_comp = nx.k_components(G1)
k_comp_dict = dict(k_comp)
print("Main component : ",max(k_comp_dict))
print("Nodes in main component : ",max(k_comp_dict.values()))
#Average Clustering Coefficient
cluster_coef=nx.clustering(G1, nodes = None, weight = None)
print("Average clustering coefficient : ",st.mean(cluster_coef.values()))
```

Finding communities using networkx community algorithm

```
#Modularity of the network
communities = list(nx_comm.greedy_modularity_communities(G1))
max_modularity=nx_comm.modularity(G1,communities)
print("Number of communities in the network: ",len(communities))
print("Modularity of the network : ",max_modularity)
```

```
#Finding node list and edge list considering each community as one node
neighbors=[]
for i in range(0,len(communities)):
    neighbors.append(set())
for i in range(0, len(communities)):
    for j in range(0, len(communities[i])):
        neighbors[i]=neighbors[i].union(set(G1.neighbors(list(communities[i])[j])))
Indicator=[]
for i in range(0,len(communities)):
    Indicator.append([])
for i in range(0, len(communities)):
    for j in range(0, len(communities)):
        Indicator[i].append(len(neighbors[i].intersection(communities[j])))
nodeslist=list(range(0,len(communities)))
edgeslist=[]
for i in range(0,len(communities)):
    for j in range(i+1,len(communities)):
        if Indicator[i][j]!=0:
            edgeslist.append((i,j))
#Plotting communities' graph
G2=nx.Graph()
G2.add_nodes_from(nodeslist)
G2.add_edges_from(edgeslist)
figure = plt.gcf()
nx.draw(G2, with_labels=True, node_color = '#32A891',node_size=350,
font_weight='bold')
figure.set_size_inches(40, 40)
```

 # Girvan-Newman algorithm to detect communities

#Girvan-Newman Algorithm

```
partition=nx_comm.girvan_newman(G1)
```

#Printing out communites and its length

```
length_of_communities=0
for i in partition:
    length_of_communities=length_of_communities+1
print(length_of_communities)
```

InfoMap Algorithm to detect communities

#InfoMap Algorithm

```
coms = algorithms.infomap(G1)
```

#Writing to csv

```
readwrite.write_community_csv(coms, path="coms.csv")
```

#Reading the communities data to variable

```
coms_df=pd.read_csv("coms.csv", header=None)
coms_df
```

 # Visualizing Communities

 # Node set 1

```
community_nodes1 = {2215,223703,413504,540136}
```

#Node set 2

```
community_nodes2 = {673, 1457, 1476, 2229, 2492, 3237, 4596, 18489, 25769, 43132,
46910, 48954, 61562, 65912, 80630, 90707, 99110, 99366, 101152, 102047, 102541,
102544, 114017, 118741, 123959, 125365, 131706, 132938, 136011, 139245, 141445,
142654, 145550, 148243, 149529, 149806, 156902, 167634, 178113, 179435, 179896,
181892, 184678, 188041, 195711, 200015, 200740, 206060, 209890, 214516, 220266,
220371, 221758, 221810, 225879, 234108, 244470, 244877, 248495, 254401, 276692,
276751, 279859, 280431, 294844, 320219, 324873, 334239, 336977, 342906, 353653,
355473, 359473, 359758, 374219, 375146, 382287, 384396, 389377, 390909, 393252,
404870, 414868, 416882, 436846, 438623, 438726, 444083, 450006, 452164, 462824,
```

```
468285, 479212, 480692, 482469, 494913, 501444, 502784, 508812, 509779, 513668,
514735, 522958, 532299, 539923}

# Initialize a list to store edges in the community

community_edges1 = []

community_edges2 = []

# Iterate through all edges in the network

for u, v in G1.edges():

    if u in community_nodes1 and v in community_nodes1:

        community_edges1.append((u, v))

# Iterate through all edges in the network

for u, v in G1.edges():

    if u in community_nodes2 and v in community_nodes2:

        community_edges2.append((u, v))

# Print the edges present in the community

print("Edges present in the community 1:", community_edges1)

print("\nEdges present in the community 2:", community_edges2)

#Plotting graph of a community

G3=nx.Graph()

G3.add_edges_from(community_edges1)

figure = plt.gcf()

nx.draw(G3, with_labels=True, node_color =
['skyblue','cyan','magenta','peachpuff'],node_size=3000, font_weight='bold')

#Plotting graph of a community

G4=nx.Graph()

G4.add_edges_from(community_edges2)

figure = plt.gcf()

nx.draw(G4, with_labels=False, node_color = '#32A891',node_size=50,
font_weight='bold')
```