Detection of Circular Trading using Node2Vec

Vinayak Nambiar
AI22MTECH13005
Department of Artificial Intelligence
Indian Institute of Technology, Hyderabad
ai22mtech13005@iith.ac.in

Roshin Roy
AI22MTECH13006
Department of Artificial Intelligence
Indian Institute of Technology, Hyderabad
ai22mtech13006@iith.ac.in

SarveshKumar Purohit
AI22MTECH14006
Department of Artificial Intelligence
Indian Institute of Technology, Hyderabad
ai22mtech14006@iith.ac.in

Parth Nitesh Thakkar
CS22MTECH14005
Department of CSE
Indian Institute of Technology, Hyderabad
cs22mtech14005@iith.ac.in

Rahul Vigneswaran K
CS23MTECH02002
Department of CSE
Indian Institute of Technology, Hyderabad
cs23mtech02002@iith.ac.in

Rithik Agarwal
CS22MTECH11004
Department of CSE
Indian Institute of Technology, Hyderabad
cs22mtech11004@iith.ac.in

This Report is made as part of the course Assignment of CS6890 (offered at IITH during Spring 2023), Code is available at Google Collab Link

Abstract

In circular trading, the idea is to create a loop of transactions where parties are involved in buying and selling assets or goods among themselves. They try to fake the invoices and show the money flow as an authentic goods transfer. By issuing a fake invoice, the dealer uses it to minimize his tax liability. In this work, we have shown the algorithm to find circular trading. ***Our algorithm focuses on finding the money flowing in circular trading, mostly through three cycles and two cycles in the transaction graph. We have introduced our own method to find the value describing the intensity of circular trading between dealers and sellers.*** *Then later, with the help of node2vec [1] [2] and DBCAN [4] clustering, we tried to identify the group of dealers and sellers who are performing malpractice with the help of circular trading.*

1. Problem Statement

The objective is to identify the group of people performing circular trading in a network of nodes with edge weights that indicate the strength of the transaction between a set of nodes. Each node can be interpreted as a seller or dealer, and the weight value of the edge will be the transition between them.

2. Description of the Dataset

The dataset has one file -' Iron dealers data.csv.'.

- 'Iron dealers data.csv': Consists of a directed graph where nodes represent Iron dealers and edges represent transactions between them.
  - 'Seller ID': Graph Nodes of the seller.
  - 'Buyer ID': Graph Nodes of the buyer.
  - 'Value': Importance of transaction between the seller and the buyer. Each of the edges represents the importance/strength of the transaction between the seller and the buyer.
  - 'counts':
    - Total no of transactions between buyer and seller (This will be the Number of edges in the graph): 130535
    - Total No of unique sellers, and buyers pair (This will be the Number of nodes in the graph): 799

3. Algorithm Used

We will first walk you through the overall Algorithm used for finding the circular trading and then later in the same section, we will explain each step of the algorithm in detail.

In this work, we have treated the transaction network as a directed weighted graph, and then later, we will convert the directed weighted graph to an undirected weighted graph.While performing a directed weighted graph to the undirected weighted graph, we will focus on the three and two cycle present in the directed weighted graph. **We will also discuss the algorithm for assigning weight in the undirected graph such that the weight value captures the intensity of circular trading.**

3.1. Overview of Complete Algorithm

Algorithm 1 Find_circular_trad( ):

1. G = Directed_MultiWeighted_Graph(Dataset)
   - Construct **Directed multi-weighted graph** from the given dataset. See section 3.2
2. D = **Convert_DiMulti_To_DiSimple(G)**
   - Summing up the same direction invoices. See section 3.3
3. U = **Convert_Directed_To_Undirected(D)**
   - Convert the Simple Directed-Weighted graph G to an Undirected-Weighted Graph U. See section 3.4
4. embeddings = **Node2Vec(U)**
   - Generating node Embeddings using the Node2Vec algorithm. See section 3.5
5. eps = **Elbo_method(embeddings)**
   Elbo method to find eps value for DBSCAN. See section 3.6
6. Cluster = **DBCAN(embeddings)**
   - Using Embeddings generated by Node2Vec, finding the DBCAN Cluster. See section 3.7
7. Plotting and verifying cluster in original MultiDirected Graph

3.2. Constructing Directed weighted MultiGraph (G) from Dataset

We have first constructed the Directed weighted Multi- Graph(G) using NetworkX [3]. With nodes as 'Buyer ID' and 'Seller ID', and the edge weight as 'value' column given in 'Iron dealers data.csv.

3.3. Convert Directed weighted Multi-Graph (G) to Simple Directed Graph (D)

In some cases, where there are multiple edges between the same two nodes, the edges are consolidated by summing up their values

Algorithm 2 Convert_DiMulti_To_DiSimple(Input G):

1.Create an empty_Simple_Directed_Graph(D)
2. **For each** (u,v,w) $\in$ G:
   if(u,v) $\in$ D:
   weight(u,v) = weight(u,v) + w
   End For
3. Return D

.

3.4. Simple Directed Graph (D) to Undirected Weighted graph(U)

Algorithm 3 Simple_directed_to_Undirected(Input D):

1.Directed Simple Graph = D
2.Create an Empty_Undirected_Graph(U) with the same nodes as G
3. For each (u,v,w) $\in$ G:
   if (u,v) $\notin$ any three cycle and two cycle:
   D.add_edge(u,v,0)
   -Three cycles = [Find all the Three length cycles $c_i \in$ G such that (u,v) $\in$ c]
   -Two cycles = [Find all the Two length cycles $d_i \in$ G such that (u,v) $\in$ d]
   -Threecycle_weight = [ ]
   -Twocycle_weight = [ ]

   For each $c_i \in$ Three cycles:
   w=[ ]
   for each (e_i,w_i) $\in c_i$:
   w.append(w_i)
   Threecycle_weight.append(w_i)

   For each $t_i \in$ Two cycles:
   w=[ ]
   for each (e_i,w_i) $\in t_i$:
   w.append(w_i)
   Twocycle_weight.append(w_i)

   weight1 =$\min$(Threecycle_weight)
   weight2 = $\min$(Twocycle_weight)
   Final_weight = $\max$(weight1,weight2)
   D.add_edge(u,v,final_weight)
4. Return U

3.5. Generating Emeddings with Node2vec on Undirected weighted graph (U)

Now, with the help of the Node2Vec algorithm, we have created node embeddings for an Undirected weighted graph (U).

- 'Input to Node2Vec': Undirected weighted graph(D) which we have created in the previous step
- 'Output of Node2Vec': Node Embeddings of Undirected weighted graph(D)

After a lot of Hyper Parameter tuning experiments, We have decided to go with the Node2Vec parameter described below:

Hyper Parameter	Value
dimensions	29
walk_length	30
num_walks	100
workers	4
p	0.9
q	0.5

Table 1. Parameters values for Node2Vec

3.6. Elbow Method: Eps and Minpoint calculation

Eps and midpoint are two important parameter for DBCAN clustering We have used Elbow Method as suggested by a professor for calculating eps and minpoint. From the graph of points vs Distance, we can see the sudden rise at (592,0.125). So for DBSCAN, eps value will be 0.125

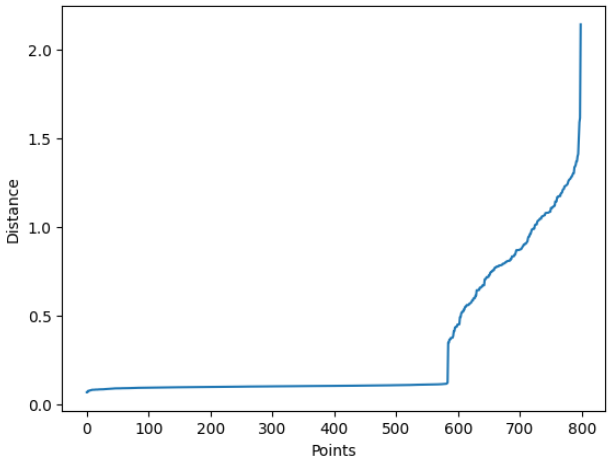

Figure 1. Eps value calculation using Elbo method

3.7. DBCAN- clustering algorithm

Now, with the help of the DBCAN clustering method, we have performed clustering on the embedding generated by Node2Vec

- 'Input to DBCAN': Node embeddings generated by Node2Vec
- 'Output of DBCAN': Cluster

eps and midpoints are two important parameters used in DBCAN. We have used elbomethod as suggested by the professor to find their optimal values.

Hyper Parameter	Value
eps	0.125
min_samples	1

Table 2. Parameters values for DBSCAN

4. Results

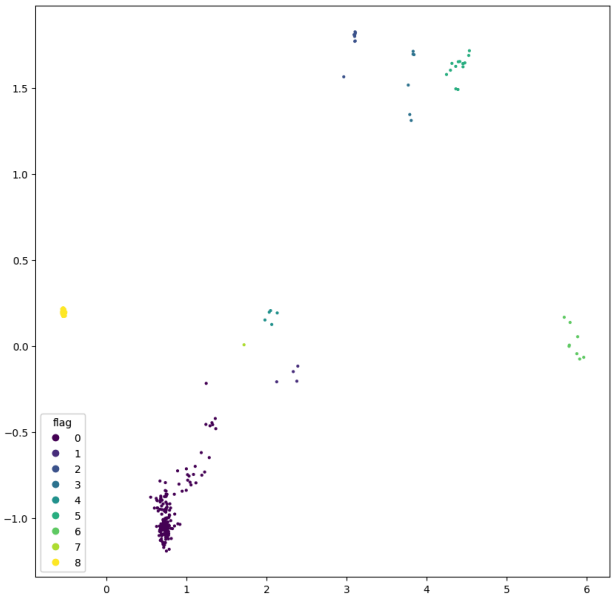

Figure 2. DBSCAN Cluster

We are getting a total **eight clusters** in the result of DBSCAN.

Cluster Information:
In cluster 1 number of elements is 4
In cluster 2 number of elements is 9
In cluster 3 number of elements is 6
In cluster 4 number of elements is 5
In cluster 5 number of elements is 14
In cluster 6 number of elements is 8

Seller ID and Dealer ID for each cluster:

Cluster 1 – [1222, 1535, 1944, 2166]
Cluster 2 – [1003, 1146, 1350, 1659, 1875, 1939, 1961, 2073, 2088]
Cluster 3 – [1023, 1060, 1105, 1264, 1623, 2085]
Cluster 4 – [1002, 1087, 1327, 1449, 1962]
Cluster 5 – [1051, 1079, 1101, 1132, 1135, 1136, 1500, 1656, 1690, 1747, 1809, 1839, 1930, 1990]
Cluster 6 – [1037, 1104, 1148, 1172, 1243, 1293, 1774, 2184]

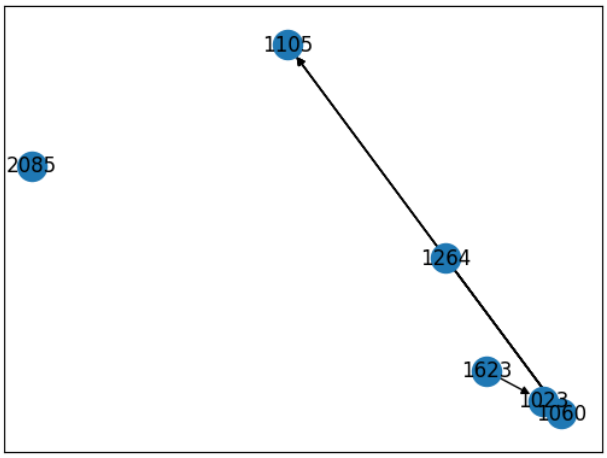

Figure 3. Verification of cluster 3

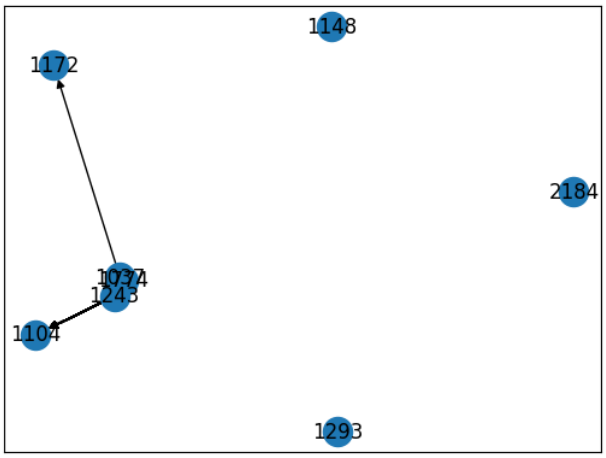

Figure 4. Verification of cluster 6

References

[1] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. *CoRR*, abs/1607.00653, 2016. 1

[2] https://github.com/eliorc/node2vec. 1

[3] https://networkx.org/documentation/stable/install.html. 2

[4] https://scikit learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html. 1