AFRICAN SCHOOL OF ECONOMICS

NETWORK PROJECT REPORT

Fraud Detection: Visualization of MTN-Benin Credits Transfer Network

by

Henri Noël Kengne

March 11, 2020

Declaration of Authorship

I, Henri Noël Kengne , declare that this project titled **Fraud Detection: Visualization of MTN-Benin Credits Transfer Network** has been composed by myself and contains original work of my execution.

Contents

The whole is more than the sum of the parts.

Metaphysica
Aristotle

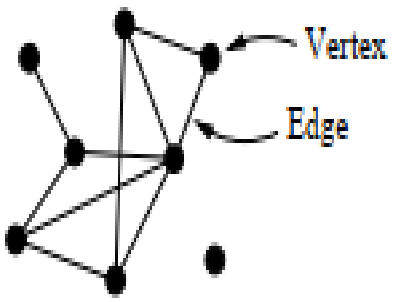
1 introduction

I had the opportunity to work in the revenue insurance department of MTN-Benin[1] as a Data Science intern for about three months. In that role, along with my fellows, our job consisted of retrieving the data via SQL server and analyzing them to detect fraud or inconsistencies in the system that could cause a financial loss for the company or subscribers' dissatisfaction. For instance, when a subscriber sends money to a relative, friend, or a business partner via the MTN mobile money system, if the latter does not receive the exact amount that has been sent, then there is a problem; or if the system did not charge this financial transaction, then there is an inconsistency in the charging of financial transactions. Issues like these are crucial in the sense that they can seriously hinder the financial stability of the company and negatively affect its customers' experiences. The sale of credits in the form of communication packages and internet packages is the mainstay of MTN's business activity. The company has enforced a set of rules that regulate the flow of credits between its different agencies and service centers scattered all over the territory. Therefore, any action that violates the rules or that is unethical is a fraud. Sometimes, the density of the dataset makes it difficult to detect fraudulent activities. A possible solution to this problem consists of constructing the network of credits distribution and performing a kind of links analysis. That is, by visualizing the edges of the network, one can answer questions like "who sent what to who?", "Is this transaction allowed?", "What is the amount of the transaction?". As we will explain in a subsequent section of this document, at MTN, a credit transaction is not allowed between certain types of distributors[2]. In this project, we aim to detect this kind of fraud by using networks. firstly, we used the python language to construct the credit transfer network in three efficient layouts, namely Hive plot, Cerco plot, and arc plot. We chose these layouts because they allow us to distinguish the links between different groups(types) of distributors efficiently. Secondly, we inspect the links between different groups of distributors to detect fraudulent transactions.

2 Netwoks

2.1 What is a network?

A network (or graph in mathematics) is a way of representing the relationships among a collection of objects. Those objects are called *nodes or vertices* and the links between some of them are called *edges*. Some examples of networks are social networks (they model the relationship between people); transportation networks (they model the connectivity between locations as determined by roads or flight paths connecting them); credit transfer network (they model credit transfers between different services); food network (they model predation or feeding habits of animals in their habitats) and, etc. Sometimes, the relationship is not mutual so its direction matters. For instance, in a credit transfer network, we wish to know



[1] MTN (Mobile Telephone Network) is a multinational mobile telecommunication company in Benin
[2] A distributor can be a service center, sub-service center, point of sale,...

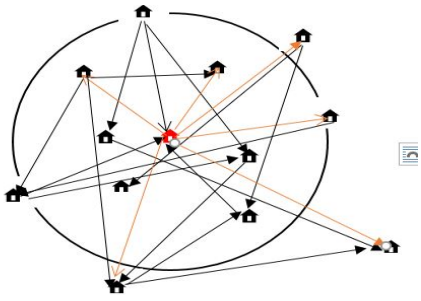
who is the sender and who is the receiver. Such a situation is depicted by a directed network where the edges are represented by arrows. A network is said to be weighted when there is a numerical value associated with each edge. Edge weight may represent a concept such as a similarity, distance, credit, or connection cost. So, networks are convenient for modeling relationships between entities. By modeling a situation by networks, one can end up gaining insight into what entities or vertices are "important".

2.2 Centrality measures of a network

Having constructed a network, say a credit transfer network of MTN-Benin, we want to be able to answer the question **"What is the most important distributor or agency in our network?"**. There are many answers to this question depending on what we mean by "important". In the jargon of the theory of networks, those answers are based on numerical values called *centrality measures*; they aim at capturing particular features of the network topology. There are a wide number of different centrality measures that have been proposed in the literature over the years, many of which were introduced first in the study of social networks, although they are now in wide use in many other areas. Although the main purpose of this project is the visualization of credits distribution, it is worth associating to the network some centrality measures because they convey how well the network is performing as a whole. For this project, we chose the most meaningful centrality measures, namely degree centrality and closeness centrality. In the literature, there are several ways of expressing these concepts mathematically but rather we will explain them intuitively along with possible interpretations of these notions in the context of credit distribution network.

Degree Centrality: The first proposal of a centrality measure for the nodes in a network is the degree. For an undirected network, the degree of a node is the number of connections between that node and the other nodes in the network. For a directed network, we have to distinguish between incoming and outgoing links. The input degree centrality of a node in a directed network is the number of edges that point at that node whereas the output degree centrality of a node is the number of edges that depart from that node. In a credit distribution network, a distributor with high input degree centrality is connected to multiple service centers that transfer credits to it; A distributor with high output degree centrality displays a high distributive power as it transfers credits to a large number of MTN's service centers and agencies.

Closeness Centrality: If all the MTN's credit distributors were within a circle, the distributor at its center would have the property that all the others are at a distance equal or smaller than the radius, while other positions may be as much as twice that distance. This suggests that the credit distributor at (or near) the center of the circle is the central (or most important) node of the network given its distances to the rest of the nodes. The advantage of being central in this sense comes from the possibility of distributing credits, being sure that the time needed to reach the whole network is as short as possible. This is the idea behind closeness centrality: For each node, you calculate the sum of the shortest distances to all the other vertices in the network and define a centrality in which shorter distances imply higher closeness centrality and vice versa. Likewise to degree centrality, closeness centrality also admits output and input version for directed network, depending on whether the distances are computed from or to the reference node, respectively.



3 Credits Distribution Network of MTN-Benin

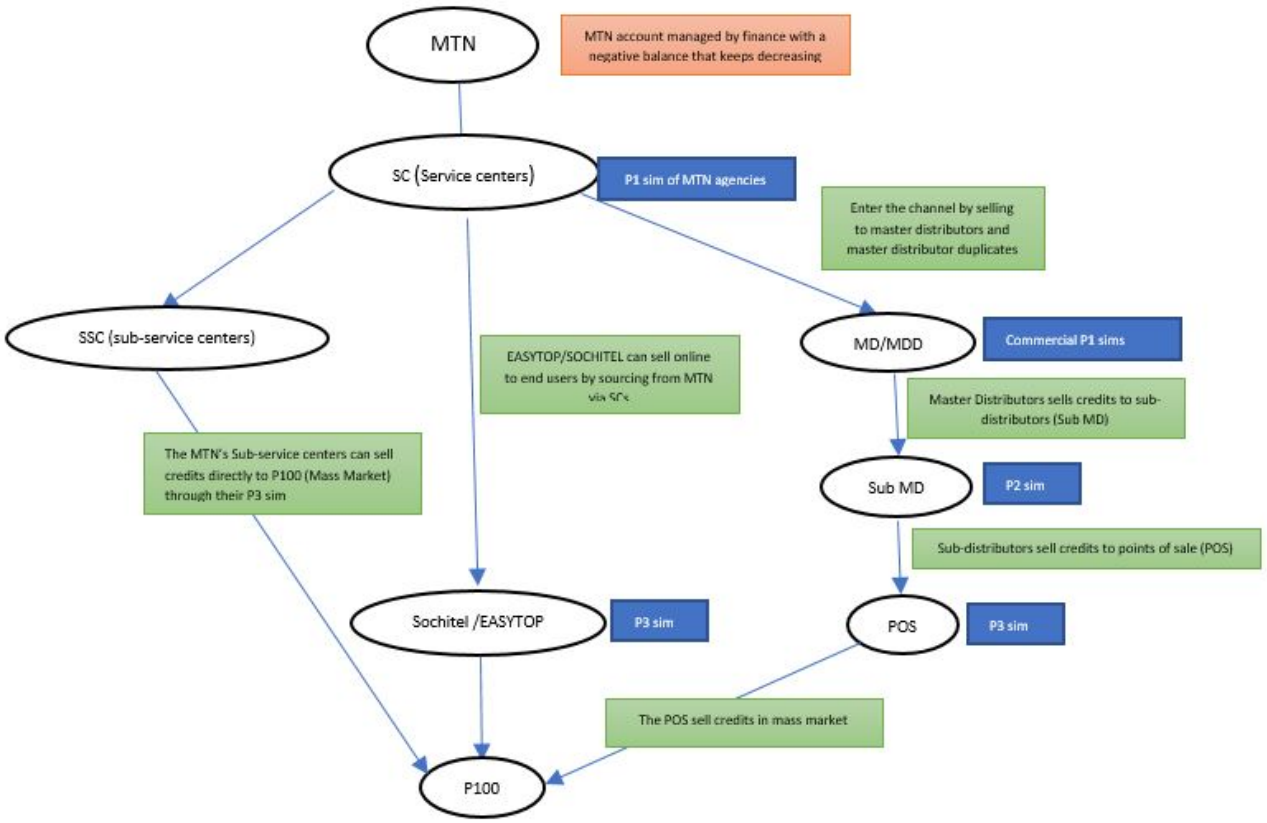


Figure 1: Schematic diagram of Credits Distribution Network of MTN-Benin

The above diagram depicts a simplified scheme of a credit distribution network of MTN-Benin. There are five types of credit distributors, namely service centers (SC), sub-service centers (SSC), master distributors (MD), master distributors duplicates (MDD), Easytop, and point of sales (POS). The last component of the network is the mass market (P100), constituted of people like you and me who buy credits for communication or internet from the POS at the corner of the street. Each distributor uses a special sim card depending on its type. For instance, the SC use sim card P1, Easytop and POS use sim cards P3. When the company wants to inject a certain quantity of credits (up to 2 billion FCFA) in its distribution network, it distributes it to its various service centers which in turn are responsible for selling the credits they have received to SSC, MD, MDD, or EASYTOP. The SSC, POS, and EASYTOP are allowed to sell credits directly to the mass market(P100), except the MDD and MD. The latter can only sell credits to Sub-MD which in turn trade with the POS. The situations that are not depicted in the diagram are situations where the distributors of the same type can trade with each other. Any other form of credit trade that is not depicted in the diagram is prohibited. Case in point, it will a fraud if an SC sells credits directly to a sub-MD or a POS. Thousands of credit transactions are recorded every day, so detecting this kind of fraud from a raw dataset is challenging. We believe that a network visualization is a powerful tool that can make the fraud detection task less painful since it displays different credit transactions between the distributors.

4 Network Visualization

Imagination is more important than knowledge.

Albert Einstein

4.1 Data

	ers_from_partner_id	ers_to_partner_id	To_Partner_Type_ID	Amount_transfered	ers_sender_rs_type	ers_receiver_rs_type
0	22967091118	22966481607	DUOS_RS17854	20000	SD	POS
1	22962513640	22962305093	DUOS_RS13683	6500	SD	POS
2	22996186679	22996880128	DUOS_RS19592	1560	POS	POS
3	22966608245	22969869100	DUOS_RS22339	20000	SD	POS
4	22962301212	22969875744	TEDDY_RS24284	20000	SD	POS

Figure 2: The top five observations

The dataset at our disposal is retrieved via SQL server from the database CDR (Call Details Records). It contains 89,384 observations and 35 variables. For this project, we identified six variables of interest, namely the sender's ID, The receiver's ID, the sender's type, the receiver's type, the amount of the transaction, and the receiver's type id. Despite the large size of the data

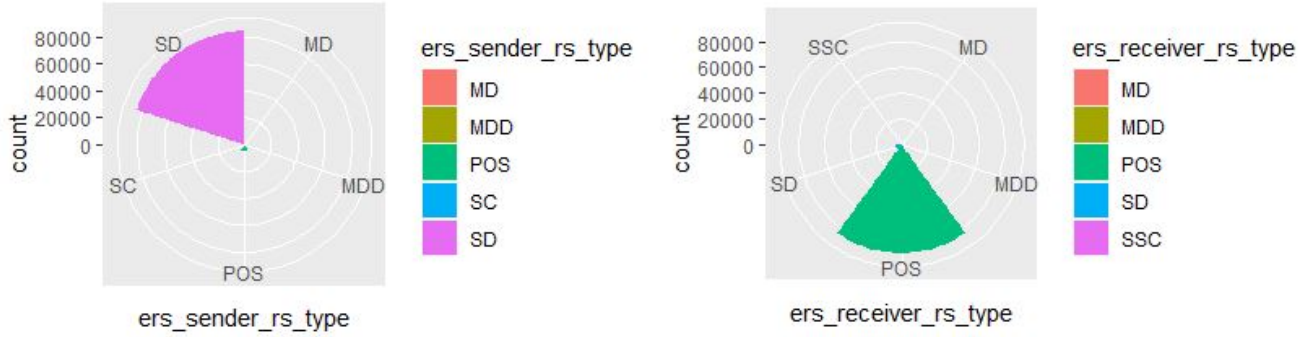


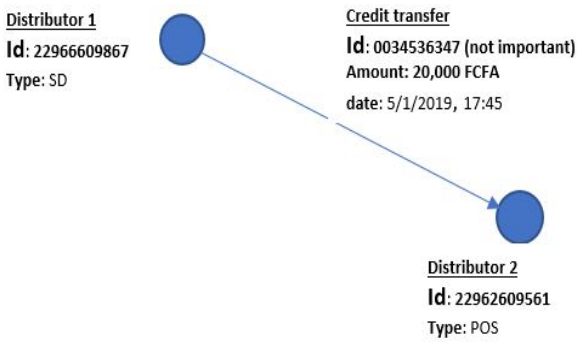
Figure 3: Pie chart of distributors's type

set, on the one hand, most of the senders are of type SD and only a few of them are of type POS; on the other hand, the majority of the receivers are of type POS and the rest are of type SD. The distributors of type MD and MDD are practically absent in the data. This happened by chance because we randomly retrieved a portion of the data from the database. Due to this situation, even though we will use just a fraction of the data set to build the network, we expect to see the bulk of the credit transactions going from SDs to POSs. Furthermore, there may not be any fraudulent activities detectable from our network because SDs are permitted to trade with POSs only.

4.2 Credit Transfer Network Structure

The two most fundamental questions in network science are:

- What is the vertex (or node)?
- When are two vertices connected?



In this project, the vertex is a credit distributor (MTN's service center or agency). Two distributors are connected if there is a credit transfer between them. Nodes and edges can have metadata (also called attributes) associated with them. In python, the attributes are stored in Key-value pair in a dictionary. For instance, the dictionary of the attributes of the distributor 1 is $\{id : 2296660987,\ type :\ 'SD'\}$. id and type are the keys and 2296660987 and 'SD' are the values. The attributes of a credit transfer are the id of the transaction, its amount, and the day on which the transfer was effectuated. The construction of the network requires the python package *networkx*. It allows us to manipulate, model, and analyze graphs. Let us start by building a credit transfer network from a very small dataset that we have modified to include all types of distributors. Also, for confidentiality concerns, the distributors' IDs have been changed. For the moment, the coordinates and the colors of the vertices are specified in the code manually. This becomes cumbersome while working with a large dataset. Therefore, in the main program, we make things more practical as a color is automatically assigned to each node depending on the type of distributor it represents.

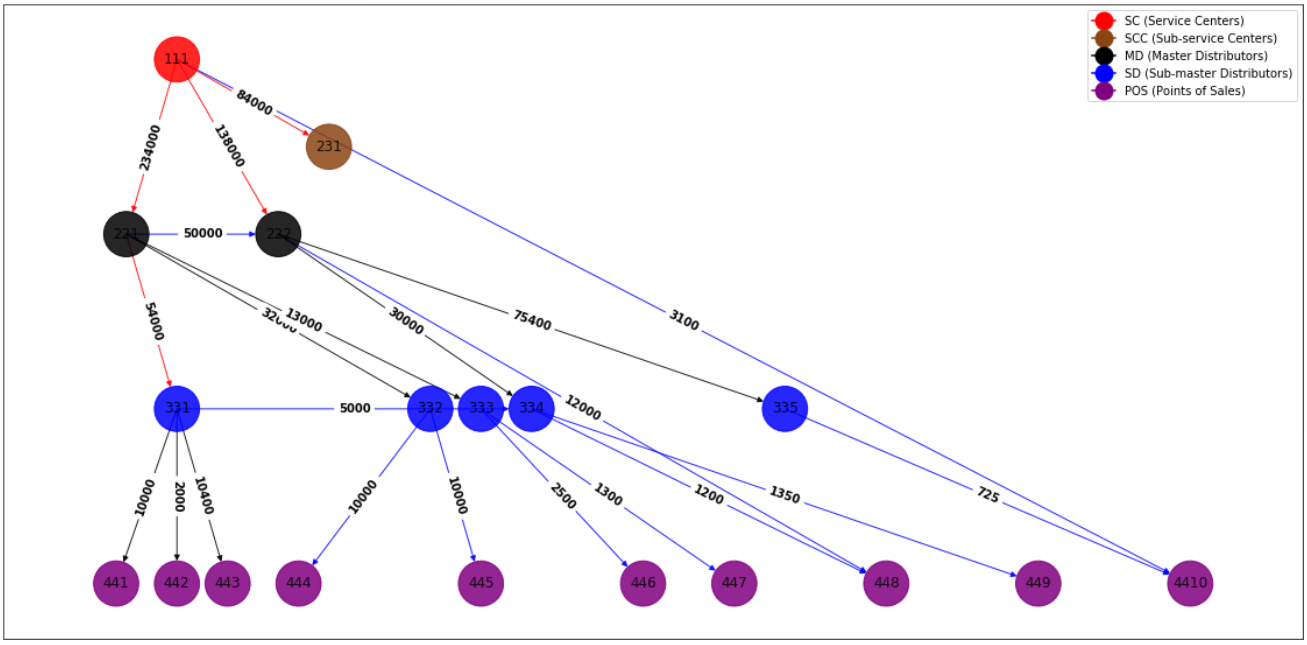
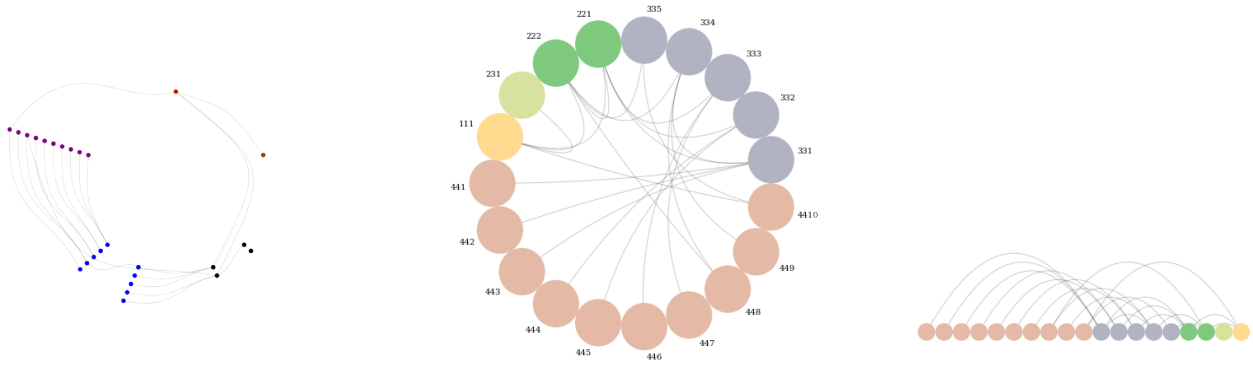


Figure 4: credit transfer network from a small dataset

By visualizing the network depicted in figure 4, we notice that an SC-distributor (in red) sold credits directly to a POS-distributor (in saddle brown) and an MD-distributor (in black) transferred credit to a POS-distributor. These credit transactions are fraudulent as they violate the company's recommendations on credit transactions. The disposition of nodes of a network can make its visualization beautiful and easy to interpret, especially when the network is large and complex. There exist "fancy" graph visualization methods provided by the python packages *nxviz* and *pyveplot*, namely Hive plot, Cerco plot, and Arc plot.

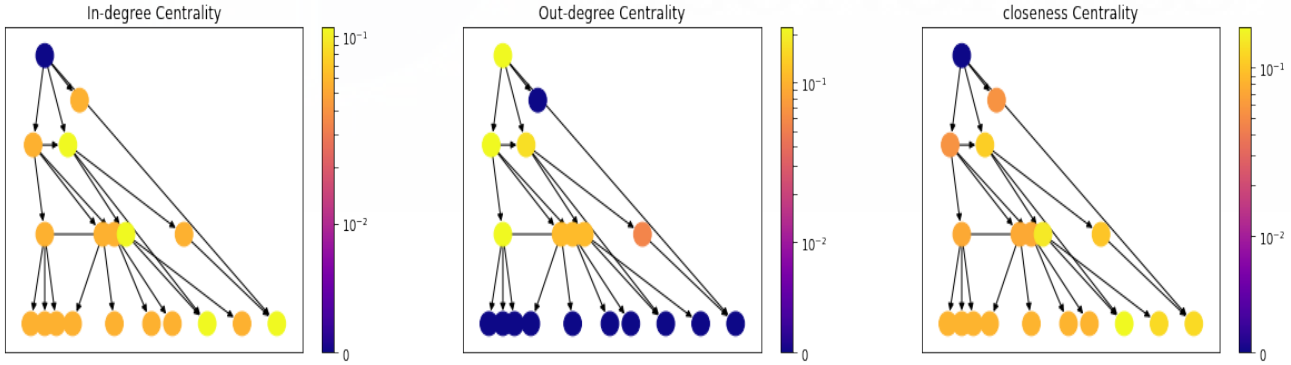


(a) Hive plot (b) Cerco plot (c) Arc plot

Figure 5: hive, cerco and arc plot of the credit transfer network.

Arc plot lays down the nodes of the graph along one axis and groups them according to the types of distributors they represent. The arcs depict credit transfers between them. Unlike arc plot, cerco plot arranges the nodes circularly and also groups them by types. When we examine both layouts, we can clearly distinguish the forbidden credit transaction between an SC distributor and a POS distributor. The hive plot is another representational and computational solution to visualizing complex networks. Hive plot places nodes on "radially oriented" linear axes, like a coordinate system. Unlike other network visualization techniques, hive plots project the network onto a multi-dimensional coordinate system that is based on properties of the network, like clustering or connectivity. This allows us to spot patterns in our credit transfer network. Again, it shows us the fraudulent credit transfer between an SC distributor and a POS distributor.

We can also visualize the degree and closeness centrality of the different nodes in the network.



(a) Indegree Centrality (b) Outdegree Centrality (c) Closeness Centrality

Figure 6: degree and closeness centrality of the network

Interestingly, that the unique SC-distributor has a zero in-degree centrality. It is the source of the credit diffused in the network. The distributors in yellow have the highest in-degree centrality as they receive credits from the highest number of distributors. The SC distributor and the POS distributors are represented by the terminal nodes in the network, so it is obvious that their out-degree centrality is zero. As for the closeness centrality, we can see that the SC type distributor is not "close" to other distributors because it does not receive credits from any distributor.

The network depicted in figure 7 is created from a sample of 52 observations of the data described in section 4.1. As we expected, most of the credit transactions occurred between the

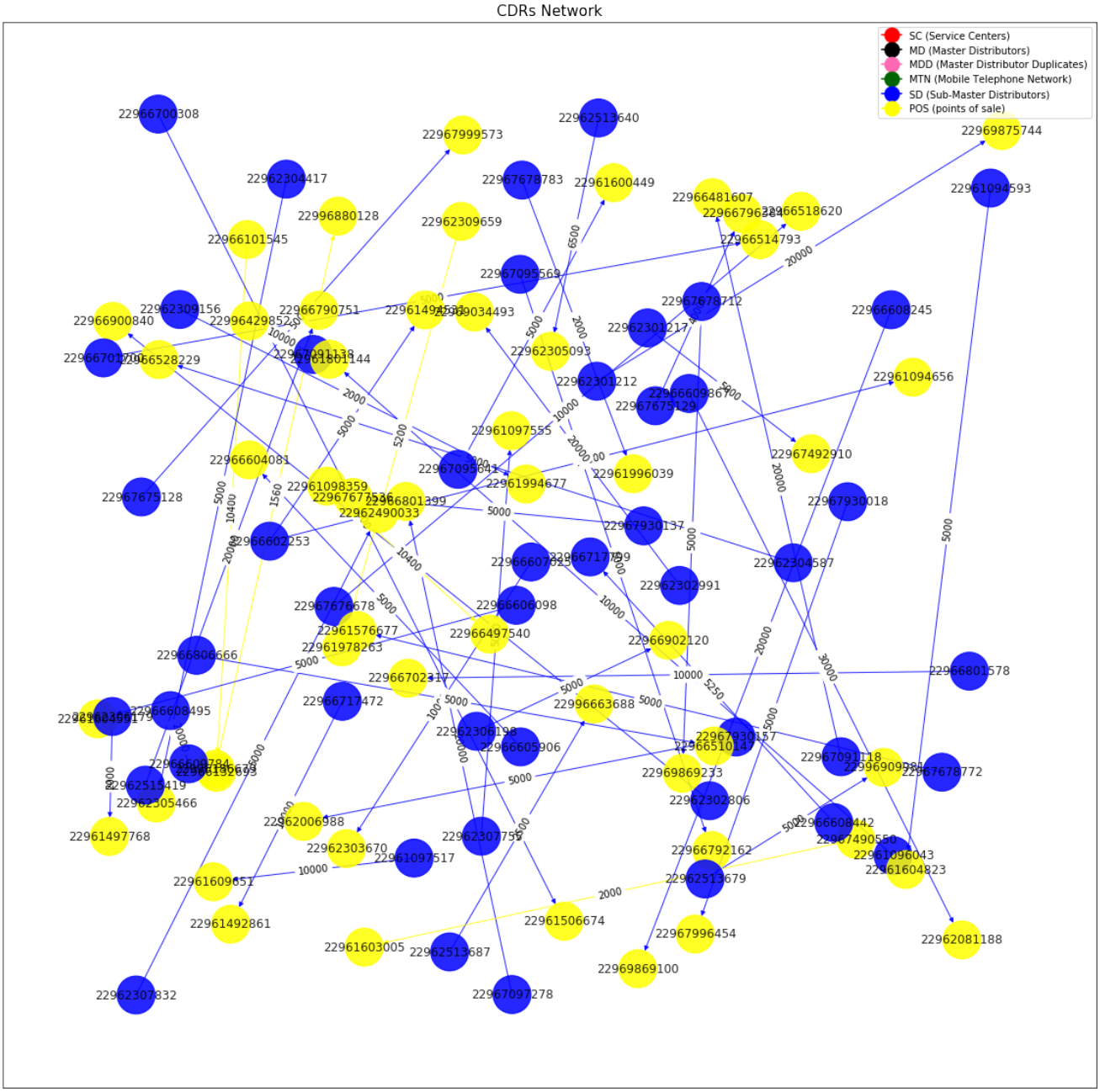
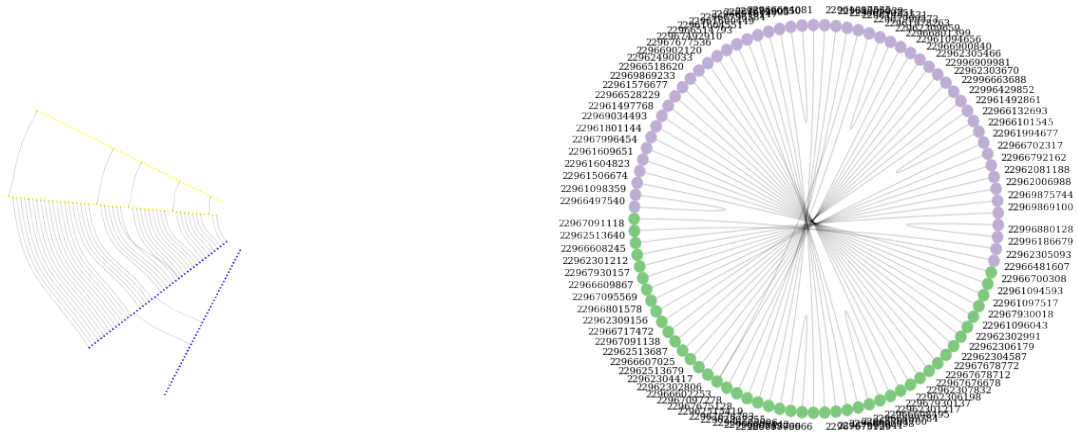


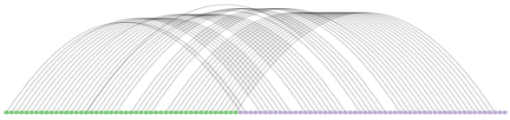
Figure 7: credit transfer network from a sample of 52 observations

SDs and POSs. A careful analysis of this graph shows that there is nothing unusual. This is corroborated by the hive plot, cerco plot, and arc plots of the network depicted in figure 8. All three layouts show that most of the credit transactions happened between SD-distributors and POS-distributors. The Hive plot shows that there are two credit transfers between the SDs and 5 credit transfers between the POSs.

A visualization of the degree and closeness centralities of the network depicted in figure 7 (see figure 9) shows that most of the POS-distributors display the highest in-degree centrality while the majority of SD-distributors have a zero in-degree centrality. Most of the POS distributors have zero out-degree centrality. Only the SD-distributor with the id 22966602253 has the highest out-degree centrality with two credit transfers done because each of the distributors initiated at most one credit transfer.

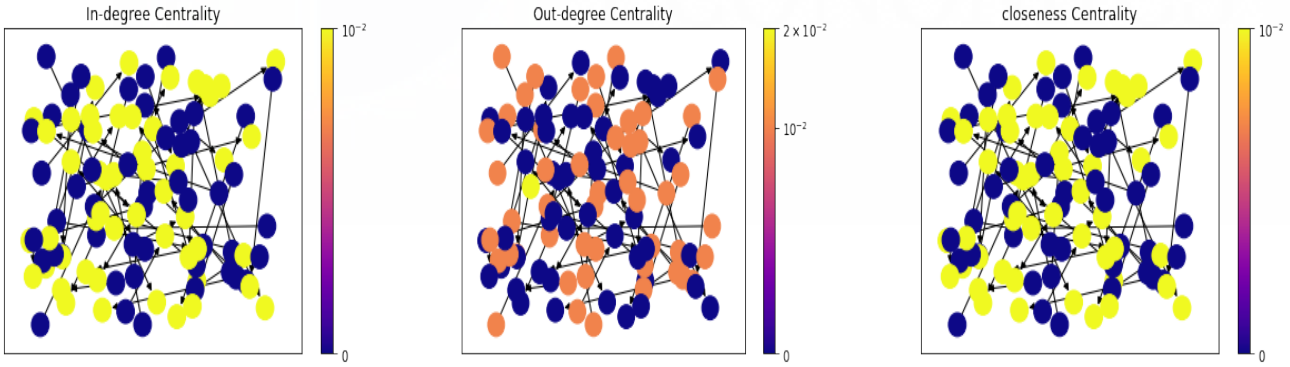


(a) Hive plot (b) Cerco plot



(c) Arc plot

Figure 8: hive, cerco and arc plot of the credit transfer network.



(a) Indegree Centrality (b) Outdegree Centrality (c) Closeness Centrality

Figure 9: degree and closeness centrality of the network depicted in figure 8

The network depicted in figure 10 is what we got when we increased the sample size to 300. The network is dense and its visualization is extremely difficult. Let us look at its hive, arc and cerco plot (see figure 11). We can see that most of the credits transfers go from the SD type distributors to POS type distributors and there is no fraudulent transfer. Again, the hive plot seems to be more efficient in detecting fraud than the cerco plot and the arc plot.

When the network is dense, it is sometimes difficult to visualize it. One way to solve this issue is to represent the network in 3D (see figure 11)

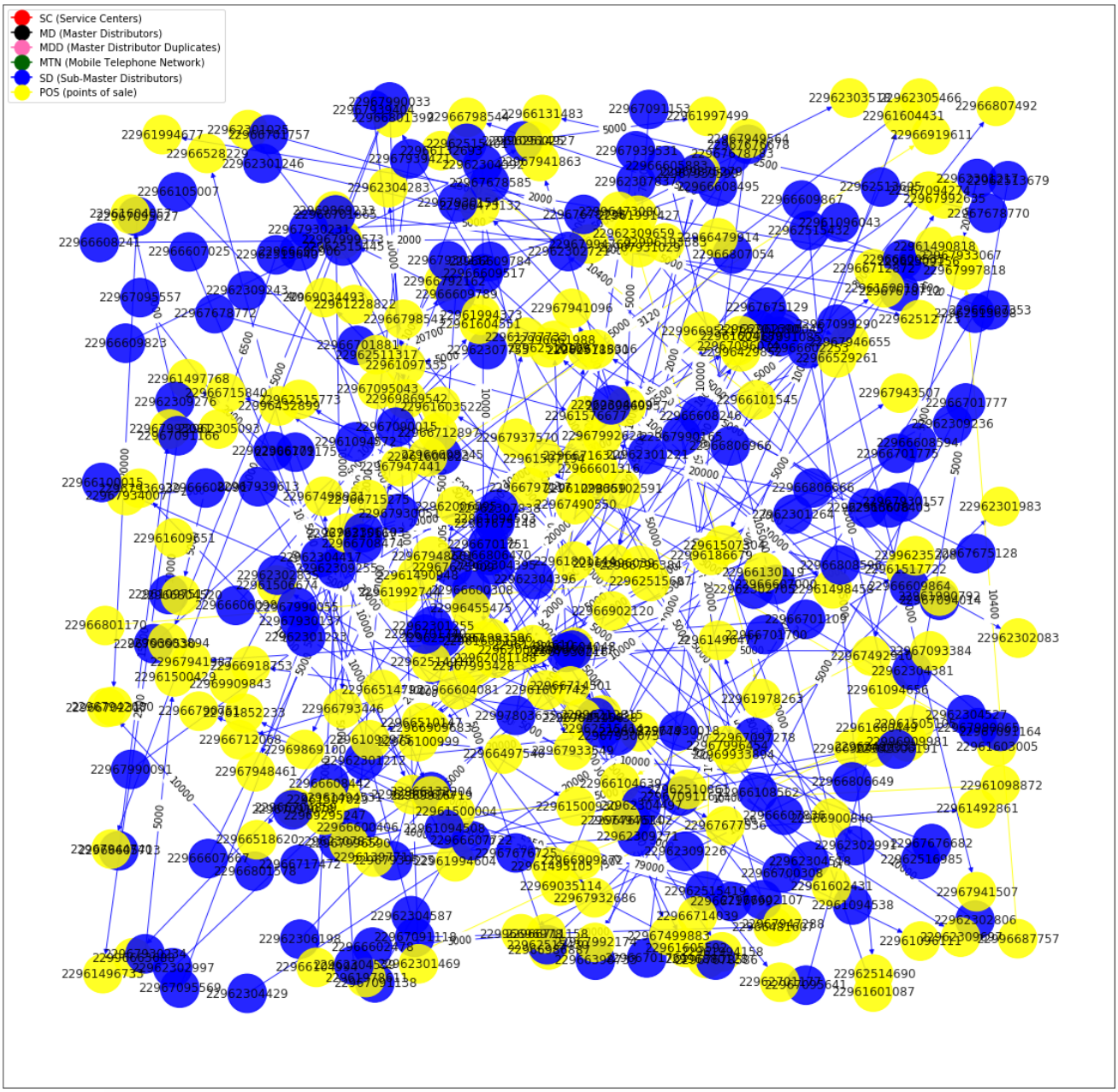
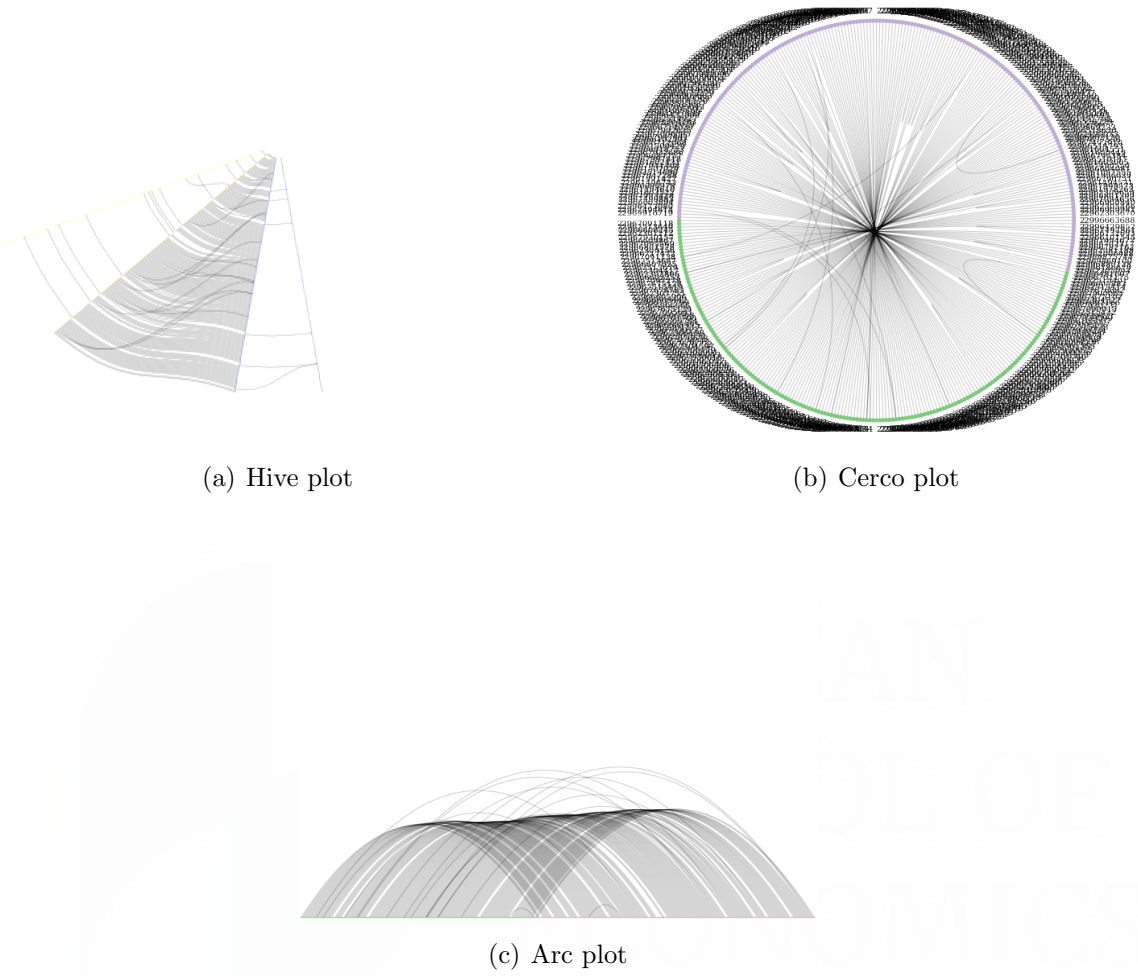


Figure 10: credit transfer network from a sample of 300 observations

5 Conclusion

In this project, we have shown how networks can be powerful tools for fraud detection. The main drawback is that when the sample size is too large, the visualization of the resulting network becomes cumbersome. One idea to overcome this difficulty will be to write a program capable of extracting from the data only fraudulent links. Those links will then constitute what we can call a "Fraudulent Network".



(a) Hive plot

(b) Cerco plot

(c) Arc plot

Figure 11: hive, cerco and arc plot of the credit transfer network depicted in figure 11.

6 Python Codes

```
##############################################################################
#    ========================================================================#
#     Fraud Detection: Visualization of MTN-Benin Credits Transfer Networt   #
#    ========================================================================#
#                                                                            #
#   Author: Henri Noel Kengne        Instructor: Dr. Daniel Felix Ahelegbey  #
#                                                                            #
#              Institution: African Schoool of Economics                     #
##############################################################################



# THE NECESSARY LIBRARIES
import networkx as nx
import pandas as pd
import plotly
import matplotlib.pyplot as plt
from matplotlib.patches import Patch
from matplotlib.lines import Line2D
```

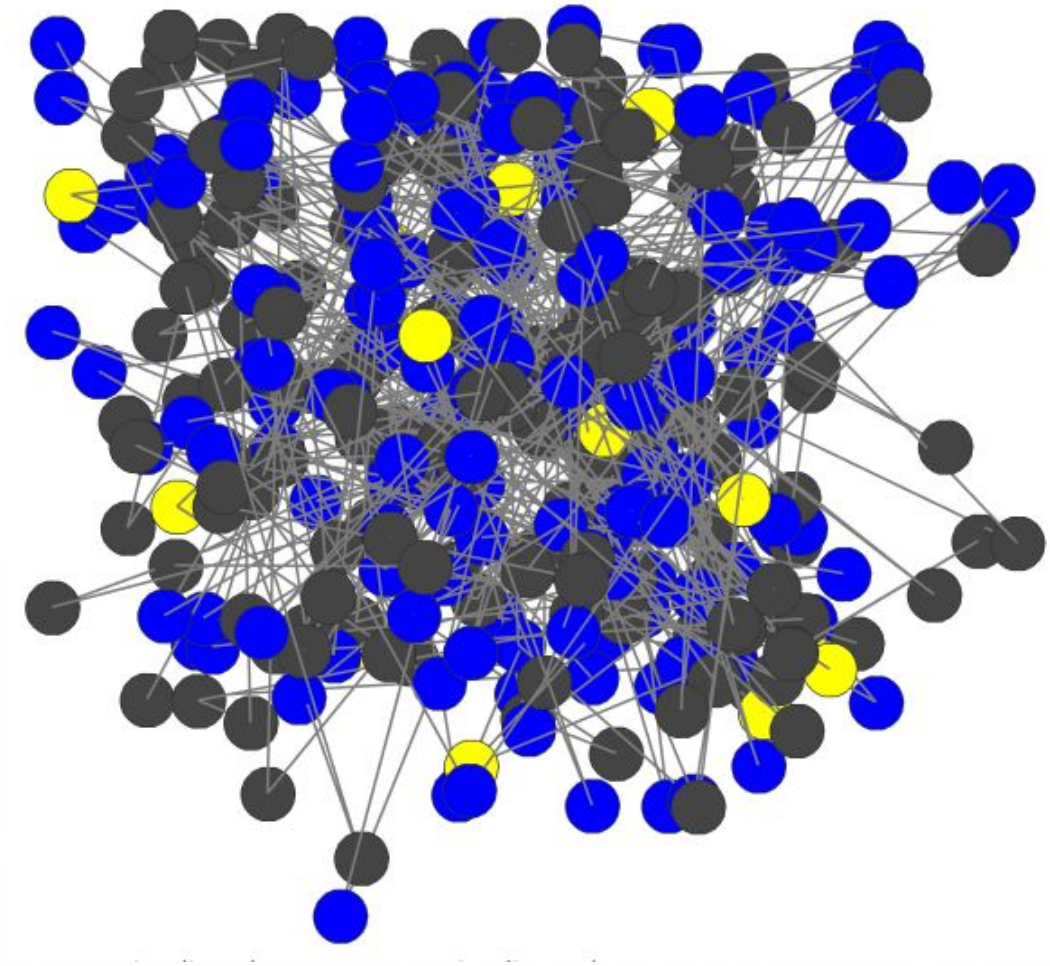


Figure 12: Dynamic Credit transfer network in 3D

```
import matplotlib.colors as mcolors

# for Notebook
# %matplotlib inline

# First, we are defining a simple method to draw the graph and the centrality
    metrics of nodes with a heat map.
def draw(G, pos, measures, measure_name):

  nodes = nx.draw_networkx_nodes(G, pos, node_size=250, cmap=plt.cm.plasma,
  node_color=list(measures.values()),
  nodelist= measures.keys())
  nodes.set_norm(mcolors.SymLogNorm(linthresh=0.01, linscale=1))

  #labels = nx.draw_networkx_labels(G, pos)
  edges = nx.draw_networkx_edges(G, pos)

  plt.title(measure_name)
  plt.colorbar(nodes)
  #plt.axis('off')
  plt.show()

# Data
df = pd.read_csv("C:\\Users\\PT WORLD\\Documents\\MYWORKS\\intership-MTN\\MTN-
    project\\CDRs-data\\trydata.csv") # Our dataset is turned into a pandas
    dataframe
```

```
43
44 G=nx.DiGraph()   # Our network should be a directed network
45
46 # We can now build the edges of the network
47 for i,elrow in df.iterrows():
48   G.add_edge(elrow[0],elrow[1], attr_dict=elrow[4:].to_dict(), weight=elrow['
     Amount'])
49
50 # The color of the edges depends on the sender's type.
51 color_map = []
52 for node in G.nodes():
53     for fa  in df.FromAccountID :
54         if node == fa:
55         if df.ers_sender_rs_type[df.loc[df.FromAccountID  == fa].index[0]]=='SC':
56         color_map.append('red')
57
58         if df.ers_sender_rs_type[df.loc[df.FromAccountID  == fa].index[0]]=='SCC':
59         color_map.append('saddlebrown')
60
61         if df.ers_sender_rs_type[df.loc[df.FromAccountID  == fa].index[0]]=='MD':
62         color_map.append('black')
63
64         if df.ers_sender_rs_type[df.loc[df.FromAccountID  == fa].index[0]]=='SD':
65         color_map.append('blue')
66
67         if df.ers_sender_rs_type[df.loc[df.FromAccountID  == fa].index[0]]=='POS':
68         color_map.append('green')
69
70 # We can manualy enter the cordinates of the different nodes
71 xpos=[0,-50,100,150,0,250,300,350,600,-60,0,50,120,300,460,550,700,850,1000]
72 ypos=[200,150,150,175,100,100,100,100,100,50,50,50,50,50,50,50,50,50,50]
73
74 # Now that we have give colors to the edges, we add an additional column that
     contains colors to our dataframe
75 color= pd.DataFrame(data=color_map)
76 df['color']=color
77
78 # We rebuild the edges of our network and specify their weight.
79 for i,elrow in df.iterrows():
80     G.add_edge(elrow[0],elrow[1], attr_dict=elrow[4:].to_dict(), weight=elrow['
     Amount'])
81
82 # Let's see what our dataframe looks like
83 df
84
85 # We create a dictionary that contains the nodes and their positions
86 node_col=['red','black','black','saddlebrown','blue','blue','blue','blue','blue','
     purple','purple','purple','purple','purple','purple','purple','purple','purple'
     ,'purple']
87 nodeList = {'NodeName': G.nodes(), 'X':xpos, 'Y':ypos, 'C':node_col}
88 nodeFrame = pd.DataFrame(data=nodeList)
89
90 # add node properties
91 for i,nlrow in nodeFrame.iterrows():
92     G.node[nlrow[0]].update(nlrow[1:].to_dict())
93
94 #Plot network
95 node_pos = {node[0]: (node[1]['X'],node[1]['Y']) for node in G.nodes(data=True)}
96 edge_col= [e[2]['attr_dict']['color'] for e in G.edges(data=True)]
97 fig = plt.figure(figsize=(20, 20))
```

```
node_col=['red','black','black','saddlebrown','blue','blue','blue','blue','blue','purple','purple','purple','purple','purple','purple','purple','purple','purple','purple']

legend_elements = [Line2D([0], [0],marker='o', color='red',label='SC (Service Centers)', markersize=15),
   Line2D([0], [0],marker='o', color='saddlebrown',label='SCC (Sub-service Centers)', markersize=15),
   Line2D([0], [0],marker='o', color='black',label='MD (Master Distributors)', markersize=15),
   Line2D([0], [0],marker='o', color='blue',label='SD (Sub-master Distributors)', markersize=15),
   Line2D([0], [0],marker='o', color='purple',label='POS (Points of Sales)', markersize=15)]

# Create the figure
fig,ax = plt.subplots(figsize=(20, 10))
ax.legend(handles=legend_elements)
nx.draw_networkx(G, pos=node_pos, arrows=True, edge_color= edge_col, node_size=1500, alpha = .85, node_color=node_col,with_labels=True)
labels = nx.get_edge_attributes(G, 'weight')
nx.draw_networkx_edge_labels(G, pos=node_pos, edge_labels=labels,font_color='k', alpha=2,font_weight='bold') # edge_labels=labels,
#plt.title('', size=15)

#plt.axis("off") # Uncomment this line to remove axes
plt.show

# VISUALIZE THE HIVE PLOT
from hiveplot import HivePlot
nodes = dict()
nodes['red'] = [n for n,d in G.nodes(data=True) if d['C'] == 'red']
nodes['saddlebrown'] = [n for n,d in G.nodes(data=True) if d['C'] == 'saddlebrown']
nodes['black'] = [n for n,d in G.nodes(data=True) if d['C'] == 'black']
nodes['blue'] = [n for n,d in G.nodes(data=True) if d['C'] == 'blue']
nodes['purple'] = [n for n,d in G.nodes(data=True) if d['C'] == 'purple']
edges = dict()
edges['group1'] = G.edges(data=True)

nodes_cmap = dict()
nodes_cmap['blue'] = 'blue'
nodes_cmap['red'] = 'red'
nodes_cmap['black'] = 'black'
nodes_cmap['purple'] = 'purple'
nodes_cmap['saddlebrown'] = 'saddlebrown'
edges_cmap = dict()
edges_cmap['group1'] = 'darkgreen'
h = HivePlot(nodes, edges, nodes_cmap, edges_cmap)
h.draw()

G.nodes(data=True)


# VISUALIZE A GRAPH BY USING MATRIX PLOT
import nxviz as nv

# VISUALIZE A GRAPH BY USING CIRCOS PLOTS
from nxviz import CircosPlot
```

```
147 Mat = CircosPlot(G , node_labels=True,node_order='Y',font_size=32, node_color='C',
    figsize=(8,8),font_weight='bold') # This creates a circosplot object Mat.
148 #plt.figure(figsize=(15, 15))
149 Mat.draw()              # Draw to the screen
150 plt.show()              # Display the plot
151 plt.savefig('Cercopplot1.png', dpi=320)
152
153
154 # VISUALIZE A GRAPH BY USING ARC PLOTS
155 from nxviz import ArcPlot
156 Mat = ArcPlot(G ,node_labels=True,font_size=25,node_order='Y', node_color='C',
    figsize=(8,8)) # This creates a circosplot object Mat.
157 #plt.figure(figsize=(15, 15))
158 Mat.draw()              # Draw to the screen
159 plt.show()              # Display the plot
160
161
162 pos = node_pos
163
164 # DEGREE CENTRALITY
165 draw(G, pos, nx.in_degree_centrality(G), 'In-degree Centrality')
166 draw(G, pos, nx.out_degree_centrality(G), 'Out-degree Centrality')
167
168 # CLOSENESS CENTRALITY
169 #draw(G, pos, nx.closeness_centrality(G, distance='weight'), 'Closeness Centrality
    ')
170 draw(G, pos, nx.closeness_centrality(G, distance=None, wf_improved=True) ,'
    closeness Centrality')
171
172 ## MAIN PROGRAM AND LARGE DATASET
173
174 import missingno as mn
175 #df = pd.read_csv("C:\\Users\\PT WORLD\\Documents\\MYWORKS\\Machine Learning\\My
    Python-Codes\\testdata4.csv")
176 #df = pd.read_csv("C:\\Users\\PT WORLD\\Documents\\MYWORKS\\intership-MTN\\MTN-
    project\\CDRs-data\\Mysmalltestdata2.csv")
177 #df = pd.read_csv("C:\\Users\\PT WORLD\\Documents\\MYWORKS\\intership-MTN\\MTN-
    project\\CDRs-data\\CDRtestdata2.csv")
178 df = pd.read_csv("C:\\Users\\PT WORLD\\Documents\\MYWORKS\\intership-MTN\\MTN-
    project\\CDRs-data\\CDRtestdata.csv")
179 #df = pd.read_csv("CDRdata.csv")
180 mn.matrix(df)
181 Graph=nx.DiGraph()
182
183 for i,elrow in df.iterrows():
184     Graph.add_edge(elrow[0],elrow[1],attr_dict=elrow[0:].to_dict())
185
186 # Here our program assigns color to nodes base on their type
187 node_col = []
188 NodeSet = list(Graph.nodes())
189 for node in NodeSet:
190     if (node in list(df.ers_from_partner_id) and df.ers_sender_rs_type[df.loc[df.
    ers_from_partner_id  == node].index[0]]=='SC') or (node in list(df.
    ers_to_partner_id) and df.ers_receiver_rs_type[df.loc[df.ers_to_partner_id  ==
    node].index[0]]=='SC'):
191     node_col.append('red')
192
193     if (node in list(df.ers_from_partner_id) and df.ers_sender_rs_type[df.loc[df.
    ers_from_partner_id  == node].index[0]]=='SCC') or (node in list(df.
    ers_to_partner_id) and df.ers_receiver_rs_type[df.loc[df.ers_to_partner_id  ==
```

```
    node].index[0]]=='SCC'):
194     node_col.append('saddlebrown')
195 
196     if (node in list(df.ers_from_partner_id) and df.ers_sender_rs_type[df.loc[df.
    ers_from_partner_id  == node].index[0]]=='MD') or (node in list(df.
    ers_to_partner_id) and df.ers_receiver_rs_type[df.loc[df.ers_to_partner_id  ==
    node].index[0]]=='MD'):
197     node_col.append('black')
198 
199     if (node in list(df.ers_from_partner_id) and df.ers_sender_rs_type[df.loc[df.
    ers_from_partner_id  == node].index[0]]=='MDD') or (node in list(df.
    ers_to_partner_id) and df.ers_receiver_rs_type[df.loc[df.ers_to_partner_id  ==
    node].index[0]]=='MDD'):
200     node_col.append('hotpink')
201 
202     if (node in list(df.ers_from_partner_id) and df.ers_sender_rs_type[df.loc[df.
    ers_from_partner_id  == node].index[0]]=='SD') or (node in list(df.
    ers_to_partner_id) and df.ers_receiver_rs_type[df.loc[df.ers_to_partner_id  ==
    node].index[0]]=='SD'):
203     node_col.append('blue')
204 
205     if (node in list(df.ers_from_partner_id) and df.ers_sender_rs_type[df.loc[df.
    ers_from_partner_id  == node].index[0]]=='MIN') or (node in list(df.
    ers_to_partner_id) and df.ers_receiver_rs_type[df.loc[df.ers_to_partner_id  ==
    node].index[0]]=='MIN'):
206     node_col.append('darkgreen')
207 
208     if (node in list(df.ers_from_partner_id) and df.ers_sender_rs_type[df.loc[df.
    ers_from_partner_id  == node].index[0]]=='POS') or (node in list(df.
    ers_to_partner_id) and df.ers_receiver_rs_type[df.loc[df.ers_to_partner_id  ==
    node].index[0]]=='POS'):
209     node_col.append('yellow')
210 
211 # Here the program assigns colors to edges or links base on the type of the node
    they depart from.
212 
213 edge_col = [0 for i in range(len(NodeSet))]
214 for start in NodeSet:
215     for i in list(df.loc[df.ers_from_partner_id  == start].index):
216     if df.ers_sender_rs_type[i]=='SC':
217     edge_col[i] ='red'
218     if df.ers_sender_rs_type[i]=='MD':
219     edge_col[i] ='black'
220     if df.ers_sender_rs_type[i]=='MDD':
221     edge_col[i] ='hotpink'
222     if df.ers_sender_rs_type[i]=='SD':
223     edge_col[i] ='blue'
224     if df.ers_sender_rs_type[i]=='MIN':
225     edge_col[i] ='darkgreen'
226     if df.ers_sender_rs_type[i]=='POS': # uncomment this line if there are POS in
    your dataset.
227     edge_col[i] ='yellow'
228 
229 import random
230 
231 # We can assign random positions to the nodes
232 xpos = [random.randrange(0,7000) for i in range(len(NodeSet))]
233 ypos = [random.randrange(0,7000) for i in range(len(NodeSet))]
234 
235 # The Graph edges and theirs weight can be built.
```

```
color= pd.DataFrame(data=edge_col)
df['color']=color
df.columns = [col.strip() for col in df.columns]
df.columns

for i,elrow in df.iterrows():
Graph.add_edge(elrow[0],elrow[1], attr_dict=elrow[2:].to_dict(), weight=elrow['Amount_transfered'])

df.head() # For large dataset we just need to have a look at the top five observations
nodeList = {'NodeName': Graph.nodes(), 'X':xpos, 'Y':ypos, 'C':node_col}
nodeFrame = pd.DataFrame(data=nodeList)

# add node properties
for i,nlrow in nodeFrame.iterrows():
Graph.node[nlrow[0]].update(nlrow[1:].to_dict())

# Plot network
node_pos = {node[0]: (node[1]['X'],node[1]['Y']) for node in Graph.nodes(data=True)}
edge_col= [e[2]['attr_dict']['color'] for e in Graph.edges(data=True)]
fig = plt.figure(figsize=(20, 20))
legend_elements = [Line2D([0], [0],marker='o', color='red',label='SC (Service Centers)', markersize=15),
    Line2D([0], [0],marker='o', color='black',label='MD (Master Distributors)', markersize=15),
    Line2D([0], [0],marker='o', color='hotpink',label='MDD (Master Distributor Duplicates)', markersize=15),
    Line2D([0], [0],marker='o', color='darkgreen',label='MTN (Mobile Telephone Network)', markersize=15),
    Line2D([0], [0],marker='o', color='blue',label='SD (Sub-Master Distributors)', markersize=15),
    Line2D([0], [0],marker='o', color='yellow',label='POS (points of sale)', markersize=15)]

# Create the figure
fig , ax= plt.subplots(figsize=(20,20))
ax.legend(handles=legend_elements)
nx.draw_networkx(Graph, pos=node_pos, arrows=True, edge_color= edge_col, node_size=1500, alpha = .85, node_color=node_col,with_labels=True)
#nx.draw_networkx(G, pos=node_pos, arrows=True, edge_color= edge_col, node_size=1500, alpha = .85, node_color=node_col,labels)
labels = nx.get_edge_attributes(Graph, 'weight')
nx.draw_networkx_edge_labels(Graph, pos=node_pos,edge_labels=labels,font_color='k', alpha=1)
plt.title('CDRs Network', size=15)
#plt.axis("off") # Uncomment this line to remove the axis
plt.show

# NETWORK REPRESENTATION IN SD

import random
import plotly.graph_objs as go
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)

# To have a similar graph in 3D, we will keep previous value of 2D position and add a third one
Xn = xpos #or Xn = [random.randrange(0,7000) for i in range(len(NodeSet))]
```

```
Yn = ypos #or Yn = [random.randrange(0,7000) for i in range(len(NodeSet))]
Zn = [random.randrange(0,7000) for i in range(len(NodeSet))]
nodeList = {'NodeName': Graph.nodes(), 'X':Xn, 'Y':Yn, 'Z':Zn}
nodeFrame = pd.DataFrame(data=nodeList)

#add node properties
for i,nlrow in nodeFrame.iterrows():
Graph.node[nlrow[0]].update(nlrow[1:].to_dict())
amount = []
for i in labels:
amount.append(labels[i])
group = edge_col
position = {node[0]: (node[1]['X'],node[1]['Y'],node[1]['Z']) for node in Graph.
    nodes(data=True)}
name = [x for x in Graph.nodes()]


#edge
trace1=go.Scatter3d(x=Xn,
       y=Yn,
       z=Zn,
       mode='lines',
       line=dict(color='rgb(125,125,125)', width=3), #width represent the weight of
     the line
       text=amount,
       hoverinfo='text'
       )
# node
trace2=go.Scatter3d(x=Xn,
       y=Yn,
       z=Zn,
       mode='markers',
       name='actors',
       marker=dict(symbol='circle',
       size=17,
       color=group,
       colorscale='Viridis',
       line=dict(color='rgb(50,50,50)', width=0.5)
       ),
       text=name,
       hoverinfo='text'
       )

axis=dict(showbackground=False,
       showline=False,
       zeroline=False,
       showgrid=False,
       showticklabels=False,
       title=''
       )

layout = go.Layout(
       title="Visualization of Credits Distribution Network of MTN-Benin (3D
    visualization)",
       width=1000,
       height=1000,
       showlegend=True,
       scene=dict(
       xaxis=dict(axis),
       yaxis=dict(axis),
```

```
340         zaxis=dict(axis),
341         ),
342     margin=dict(
343     t=100
344     ),
345     hovermode='closest',
346     annotations=[
347     dict(
348     showarrow=False,
349     text="Important: click on trace0 to visualize edges or actors to visualize
        nodes",
350     xref='paper',
351     yref='paper',
352     x=0,
353     y=0.1,
354     xanchor='left',
355     yanchor='bottom',
356         font=dict(
357         size=14
358         )
359         )
360         ],     )
361
362
363 data=[trace1, trace2]
364 fig=go.Figure(data=data, layout=layout)
365
366 iplot(fig, filename='CDRs Network')   # visualize inside the notebook
367 plot(fig, filename='CDRs Network.html')    # visualize outside the notebook
368
369
370
371
372 #-------------------------------END-------------------------------------------------
```

Listing 1: Phyton code used for this project