NBA ASSIST NETWORK ANALYSIS

A project report submitted for the Final Review

CSE 3021

Social and Information Networks

Submitted by:

Pratyush Manandhar(20BCE2643)

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Winter Semester -2023

ACKNOWLEDGEMENT

I sincerely thank our Chancellor - Dr. G. Viswanathan, VIT University, for giving me the opportunity to pursue this course of Social and information networks and

introduce J-component as part of our academic curriculum. We also thank PROF.

Rajkumar R – School of Computer Science and Engineering, VIT University, for giving us the opportunity to do this project. We also thank the Dean and HOD of School of Computer Science and Engineering (SCOPE), for their continued support and

encouragement.

ABSTRACT

The NBA is one of the world's most popular sports leagues. It amasses a following from all over the world and of all demographics. Because of its high pace and constant action, a large amount of data is generated which is very accurately stored in depth by the amazing data analysis group of the NBA. All this data is publicly available, and a lot of sport statistics junkies have devised new and creative ways to analyze the game in a way that has never been done before. The teams are also aware of this data and constantly use advanced statistics metrics to optimize their team and strategy. This has resulted in the NBA becoming one of the most radically changing sports leagues. Assists are a good metric to study a team's offense and find underlying patterns. In this project I wanted to take the large amount of available data and represent a team's offense using social network graphs.

This is a great fit for using social network tools and getting a crude understanding of how each team's offense works. Using networkx and pyplot for plotting and requests for data scraping we can achieve the desired goal.

Keywords: NBA, python, data scraping, social network analysis and visualization.

Testing dataset

The dataset that we will use is from https://www.nba.com/stats. They record play by play data accurately and are completely free to use. The data that we are going to use for analysis is assists of individual players and the receiver of the assists. We will only consider the stats of player who recorded more than 50 assists in the given season to remove bench players that do not paint the picture of the actual team's offensive system.

Procedure

1. The python script takes in 3 parameters(team abbreviation, season date and assists or passes) these to scrape the NBA stats website for required data. The data is cleaned by filtering players that rarely play and also standardizing the names of players.
2. The code iterates through a list of players and calculates passing statistics between them. It creates a matrix of passing statistics, where each row represents the passer, and each column represents the receiver. The passing statistics are obtained by filtering a data frame and then summing the values, or assigning the first value if only one value exists.
3. We then use NetworkX a data modelling python package. The code uses NetworkX to create and visualize a graph based on the data stored in a matrix. It creates a graph where each node represents a player, and the edges represent the passing statistics between players. The size and layout of the graph are customized using various parameters.
4. We create offset for the name label network and finally use the weighted edges and number of assists to map the assists network of a team for that specific season.
5. The size of the node is proportional to the number of assists the players gets and the colour of the edges denote the connection of two players.

Code

```
import matplotlib.pyplot as plt
import networkx as nx
import pandas as pd
import numpy as np
import requests
import json

def pass_web(team,ssn,web):

    ### the 'team' parameter takes in an nba team's three letter abbreviation. 'GSW' for the golden state warriors, for example
    ### the 'ssn' parameter takes in an nba season (2013-14 at the earliest). '2015-16' for example
    ### the 'web' parameter takes in 'AST' or 'pass' (or other variations of the two).
    ### the 'web' parameter exists because you can create networks in which the lines represent either assists or passes -- the parameter is used to tell which one of the two you'd like to do
    ### example of usage: pass_web('GSW','2016-17','AST')

    if (web in ['ast','AST','assist','assists','assisting','ASSIST','ASSISTS']):
        web = 'assist'
    elif (web in ['pass','passes','passing','PASS','PASSES']):
        web = 'pass'
    else:
        return print('Error: Third parameter requires "assist" or "pass" input')

    headers = {'Host': 'stats.nba.com','Accept': 'application/json, text/plain, */*','x-nba-stats-token': 'true','User-Agent': 'Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/109.0.0.0 Mobile Safari/537.36','x-nba-stats-origin': 'stats','Origin': 'https://www.nba.com','Referer': 'https://www.nba.com/','Accept-Encoding': 'gzip, deflate, br','Accept-Language': 'en-US,en;q=0.9}'}
    ### scrape stats.nba.com to find the team_id for the inputted team

    url = 'https://stats.nba.com/stats/leaguedashptstats?College=&Conference=&Country=&DateFrom=&DateTo=&Division=&DraftPick=&DraftYear=&GameScope=&GameSegment=&Height=&LastNGames=0&LeagueID=00&Location=&Month=0&OpponentTeamID=0&Outcome=&PORound=0&PerMode=PerGame&Period=0&PlayerExperience=&PlayerOrTeam=Team&PlayerPosition=&PtMeasureType=Drives&Season=' + str(ssn) + '&SeasonSegment=&SeasonType=Regular+Season&StarterBench=&TeamID=0&VsConference=&VsDivision=&Weight='

    json = requests.get(url, headers=headers).json()

    data = json['resultSets'][0]['rowSet']
    columns = json['resultSets'][0]['headers']
```

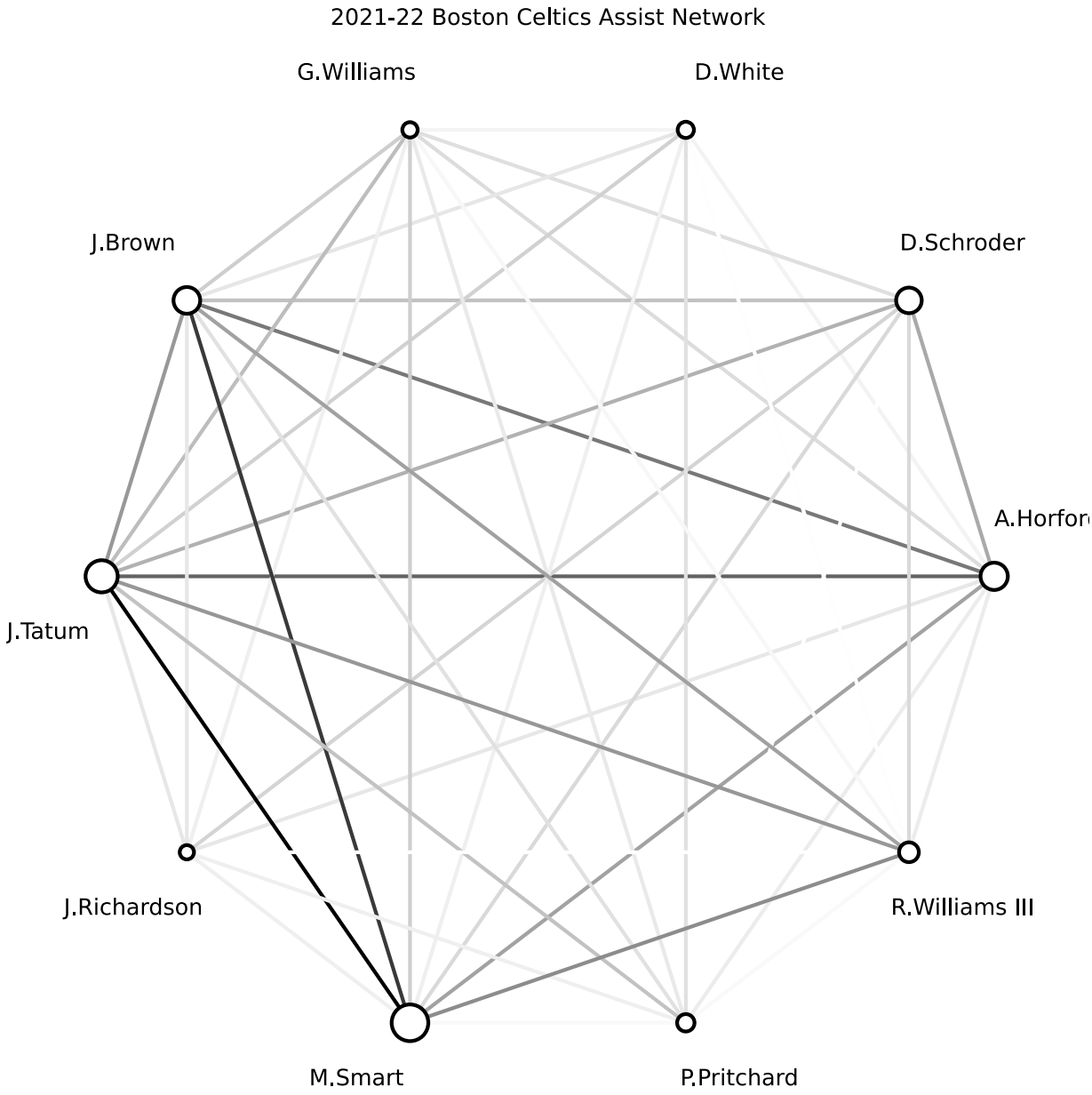
```
    tms = pd.DataFrame.from_records(data, columns=columns)

    team_id = tms[tms.TEAM_ABBREVIATION == team].reset_index(drop=True).TEAM_ID[0]
    team_name = tms[tms.TEAM_ABBREVIATION ==
team].reset_index(drop=True).TEAM_NAME[0]

    ### using the scraped team_id, find all players who accumulated at least 10 assists with that
team in the inputted season

    url =
'https://stats.nba.com/stats/leaguedashplayerstats?College=&Conference=&Country=&DateFrom
=&DateTo=&Division=&DraftPick=&DraftYear=&GameScope=&GameSegment=&Height=&L
astNGames=0&LeagueID=00&Location=&MeasureType=Base&Month=0&OpponentTeamID=
0&Outcome=&PORound=0&PaceAdjust=N&PerMode=Totals&Period=0&PlayerExperience=&
PlayerPosition=&PlusMinus=N&Rank=N&Season=' + str(ssn) +
'&SeasonSegment=&SeasonType=Regular+Season&ShotClockRange=&StarterBench=&TeamI
D=' + str(team_id) + '&TwoWay=0&VsConference=&VsDivision=&Weight='

    json = requests.get(url, headers=headers).json()

    data = json['resultSets'][0]['rowSet']
    columns = json['resultSets'][0]['headers']

    df = pd.DataFrame.from_records(data, columns=columns)

    df = df[df.AST > 50]

    players = df.PLAYER_ID.unique()

    ### using the players previously found, record all of their pass connections while on the team

    df_list = []

    for x in players:

        url =
'https://stats.nba.com/stats/playerdashptpass?DateFrom=&DateTo=&GameSegment=&LastNGa
mes=0&LeagueID=00&Location=&Month=0&OpponentTeamID=0&Outcome=&PORound=0
&PerMode=Totals&Period=0&PlayerID=' + str(x) + '&Season=' + str(ssn) +
'&SeasonSegment=&SeasonType=Regular+Season&TeamID=0&VsConference=&VsDivision='

        json = requests.get(url, headers=headers).json()

        data = json['resultSets'][0]['rowSet']
        columns = json['resultSets'][0]['headers']
```

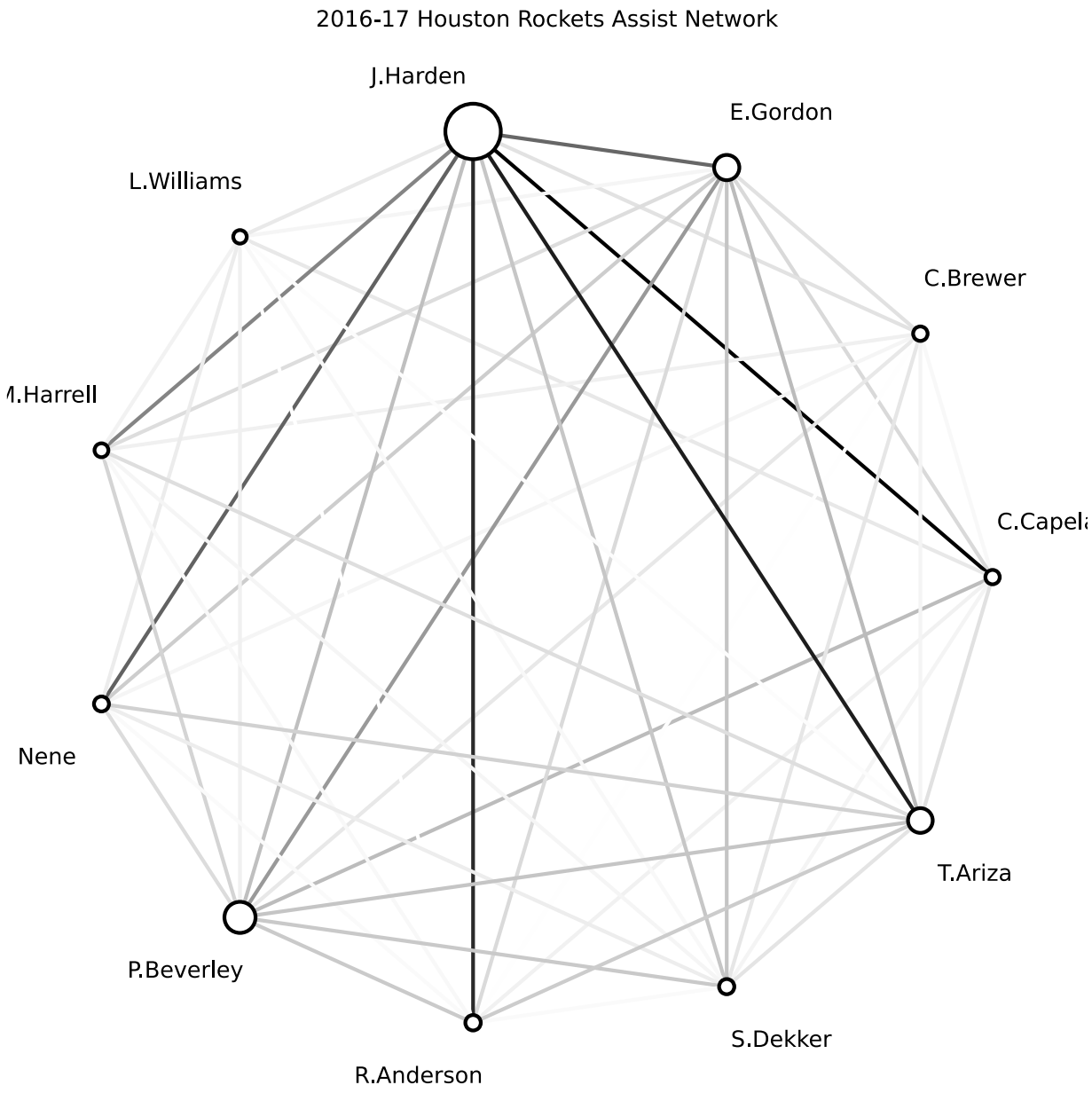
```
        df_list.append(pd.DataFrame.from_records(data, columns=columns))

    df = pd.concat(df_list)

    df =
df[['TEAM_ABBREVIATION','PLAYER_NAME_LAST_FIRST','PASS_TO','PASS','AST']]
    df.columns = ['tm','passer','receiver','passes','ast']

    ### clean up the format for the name

    def fix_name(name):
        if ',' in name:
            return name.split(", ")[1][:1] + "." + name.split(", ")[0]
        else:
            return name
    df.passer = np.vectorize(fix_name)(df.passer)
    df.receiver = np.vectorize(fix_name)(df.receiver)

    df = df[df.receiver.isin(df.passer.unique())].reset_index(drop=True)

    players = df.passer.unique()

    ### making list with assist totals for each player (will be used as input for node size)

    ast_list = []
    for x in players:
        tf = df[df.passer == x].reset_index(drop=True)
        ast_list.append(tf.ast.sum())

    ### creating adjacency matrix

    adf = pd.DataFrame(index=players,columns=players)

    adf.values[[np.arange(adf.shape[0])]*2] = 0

    if web == 'pass':
        for x in players:
            for y in range(0,len(players)):
                tf1 = df[(df.passer == x) & (df.receiver == adf.columns[y])].reset_index(drop=True)
                tf2 = df[(df.passer == adf.columns[y]) & (df.receiver == x)].reset_index(drop=True)
                if (tf1.shape[0] == 1) & (tf2.shape[0] == 0):
                    adf.at[x,adf.columns[y]] = tf1.passes[0]
                elif (tf1.shape[0] == 0) & (tf2.shape[0] == 1):
                    adf.at[x,adf.columns[y]] = tf2.passes[0]
                elif (tf1.shape[0] == 1) & (tf2.shape[0] == 1):
```

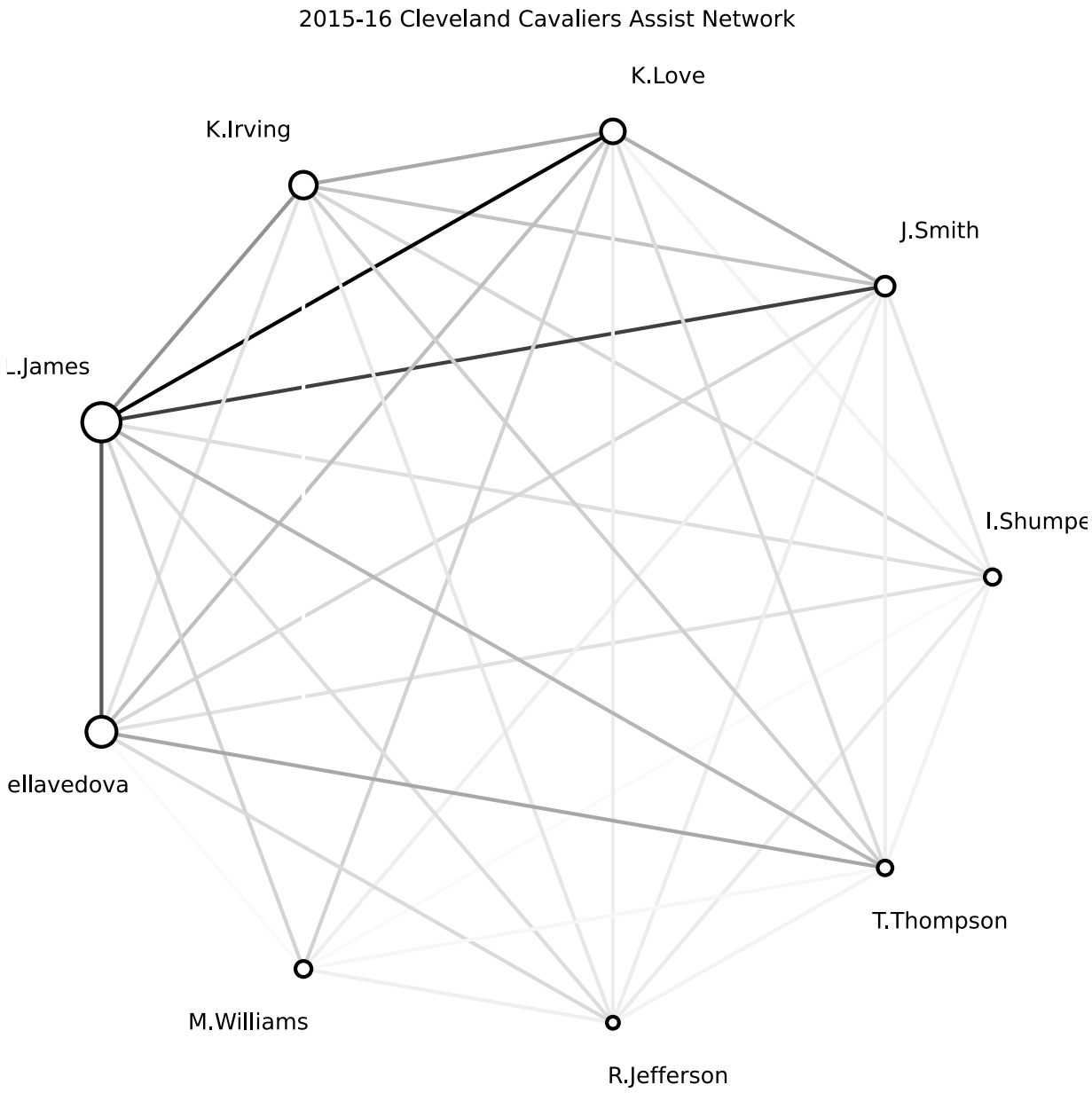
```
                adf.at[x,adf.columns[y]] = tf1.passes[0] + tf2.passes[0]
            else:
                adf.at[x,adf.columns[y]] = 0

    else:
        for x in players:
            for y in range(0,len(players)):
                tf1 = df[(df.passer == x) & (df.receiver == adf.columns[y])].reset_index(drop=True)
                tf2 = df[(df.passer == adf.columns[y]) & (df.receiver == x)].reset_index(drop=True)
                if (tf1.shape[0] == 1) & (tf2.shape[0] == 0):
                    adf.at[x,adf.columns[y]] = tf1.ast[0]
                elif (tf1.shape[0] == 0) & (tf2.shape[0] == 1):
                    adf.at[x,adf.columns[y]] = tf2.ast[0]
                elif (tf1.shape[0] == 1) & (tf2.shape[0] == 1):
                    adf.at[x,adf.columns[y]] = tf1.ast[0] + tf2.ast[0]
                else:
                    adf.at[x,adf.columns[y]] = 0

    ### graphing the network

    plt.figure(figsize=(8,8))

    G = nx.from_numpy_matrix(np.array(adf.values, dtype=int), parallel_edges=True)

    d = dict(G.degree)

    edges,weights = zip(*nx.get_edge_attributes(G,'weight').items())

    pos = nx.circular_layout(G)

    labeldict = {}
    for n in range(0,len(adf.index)):
        labeldict[n] = adf.index[n]

    # nx.draw_circular(G,labels=labeldict,with_labels=True, nodelist=d.keys(), node_size=[v for
v in ast_list],linewidths=2, edgelist=edges, edge_color=weights, width=2,
edge_cmap=plt.cm.binary, node_color='white')
    nx.draw_circular(G, nodelist=d.keys(), node_size=[v for v in ast_list],linewidths=2,
edgelist=edges, edge_color=weights, width=2, edge_cmap=plt.cm.binary, node_color='white')

    for p in pos.keys():
        cord = pos[p]
        offset_x, offset_y = 0.12, 0.12
        if (cord[0] < 0):
            offset_x *= -1
        if (cord[1] < 0):
```

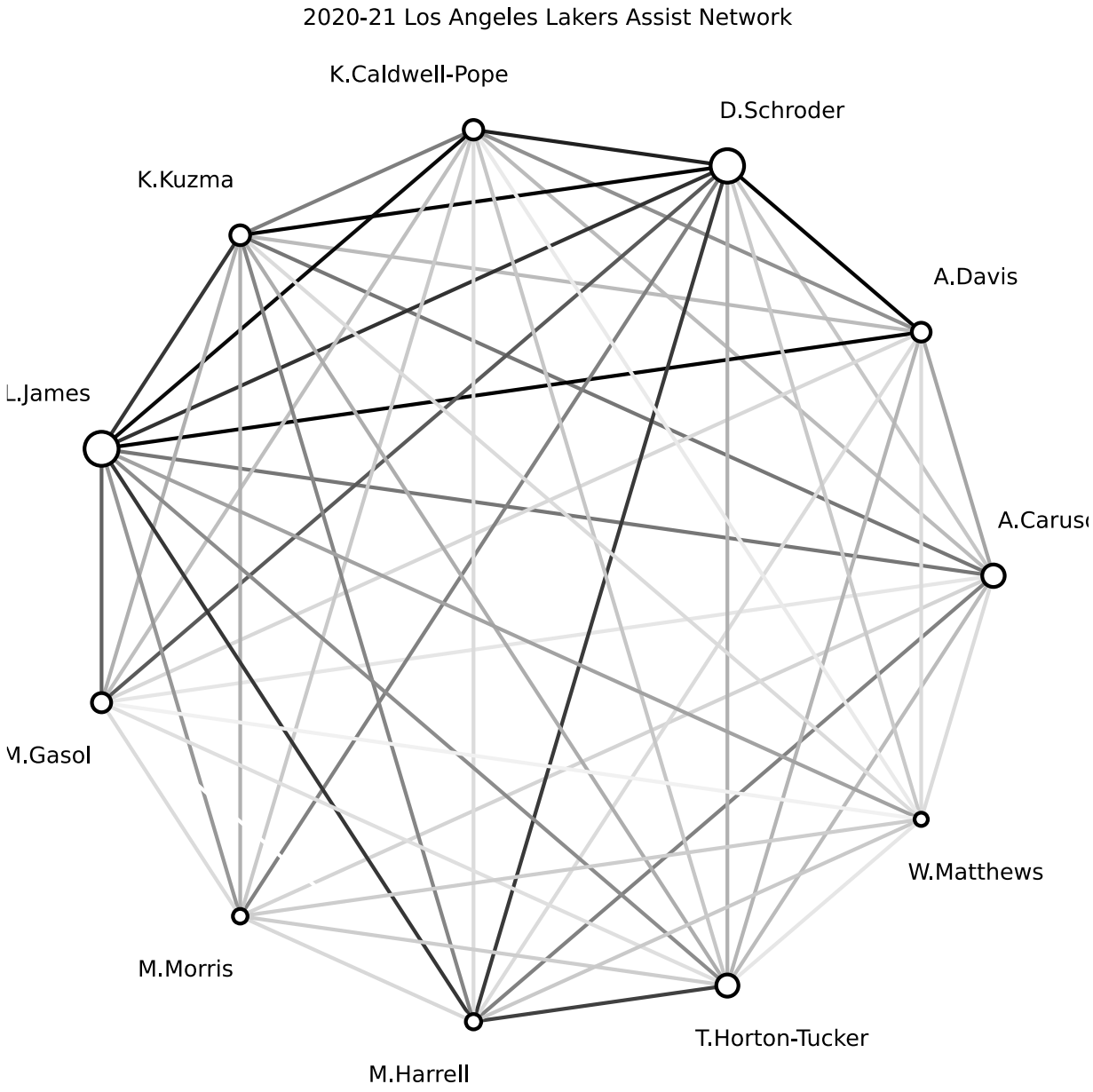
```
        offset_y *= -1
    cord[0] += offset_x
    cord[1] += offset_y

    pos[p] = cord
nx.draw_networkx_labels(G, pos, labels=labeldict, horizontalalignment='center')

ax = plt.gca()
ax.collections[0].set_edgecolor("#000000")

if web == 'pass':
    plt.title(ssn + ' ' + team_name + ' Passing Network')
else:
    plt.title(ssn + ' ' + team_name + ' Assist Network')

yr = ssn[:2] + ssn[5:] # converts '2016-17' to '2017', for instance
plt.savefig(team + yr + web  + '.svg',dpi=300,bbox_inches='tight')
```

Outputs

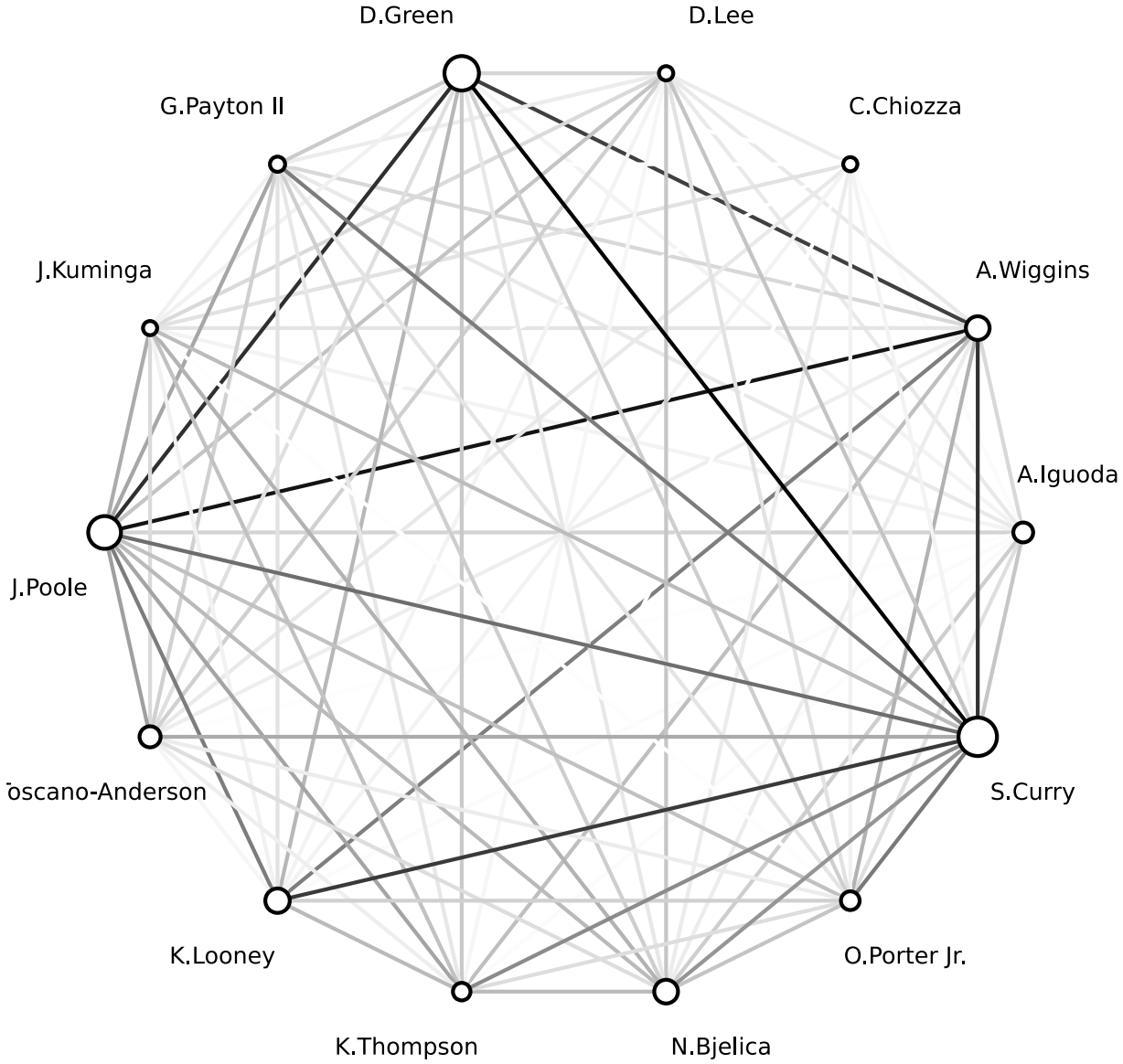

2021-22 Boston Celtics Assist Network
G.Williams
D.White
J.Brown
D.Schroder
A.Horfor
J.Tatum
J.Richardson
R.Williams III
M.Smart
P.Pritchard


2016-17 Houston Rockets Assist Network
J.Harden
E.Gordon
L.Williams
C.Brewer
M.Harrell
C.Capela
Nene
T.Ariza
P.Beverley
S.Dekker
R.Anderson

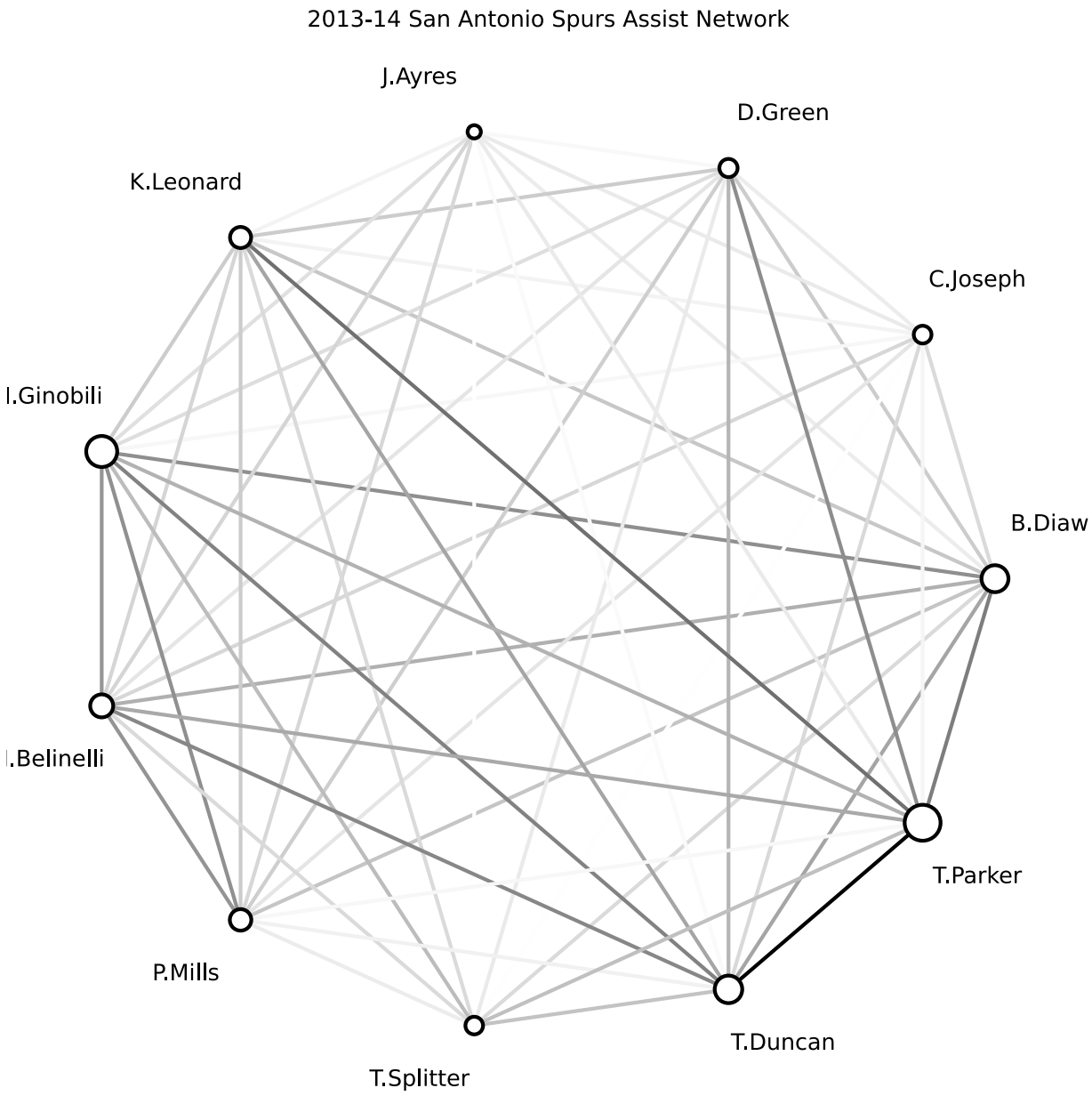

2015-16 Cleveland Cavaliers Assist Network
K.Love
K.Irving
J.Smith
L.James
I.Shumpe
ellavedova
T.Thompson
M.Williams
R.Jefferson


2020-21 Los Angeles Lakers Assist Network
K.Caldwell-Pope
D.Schroder
K.Kuzma
A.Davis
L.James
A.Caruso
M.Gasol
W.Matthews
M.Morris
T.Horton-Tucker
M.Harrell

2021-22 Golden State Warriors Assist Network




2016-17 Golden State Warriors Assist Network
I.Clark
D.Green
K.Durant
D.West
A.Iguoda
Thompson
P.McCaw
Z.Pachulia
S.Livingston
S.Curry


2013-14 San Antonio Spurs Assist Network
J.Ayres
D.Green
K.Leonard
C.Joseph
I.Ginobili
B.Diaw
.Belinelli
T.Parker
P.Mills
T.Duncan
T.Splitter

Conclusion

The difference between the two assist networks is pretty obvious at this point.

The 2017 Warriors' network clearly includes four players who are connected with black lines representing a lot of passing going on between them. Of course, these four players are Stephen Curry, Kevin Durant, Klay Thompson, and Draymond Green.

Meanwhile, the 2018 Rockets only have one connection that really stands out from the rest — James Harden and Clint Capela. Outside of that, the degree of connectivity for Houston is extremely low compared to the 2017 Warriors.

I think there's a lot more analysis that can be done with this data, and these visualizations are just a fraction of it. I'll certainly revisit this topic in the future.