دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

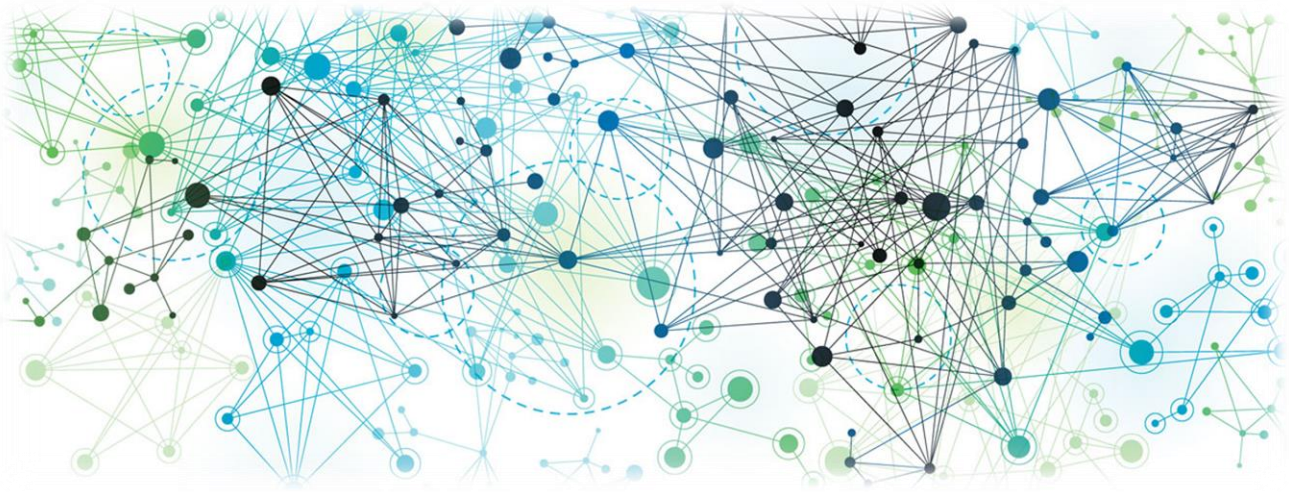
درس تحلیل شبکه های پیچیده

استاد درس جناب آقای دکتر چهرقانی

(تمرین سری دوم)

محسن عبادپور | ۴۰۰۱۳۱۰۸۰ | m.ebadpour@aut.ac.ir

نیمسال اول سال تحصیلی ۱۴۰۲-۱۴۰۱

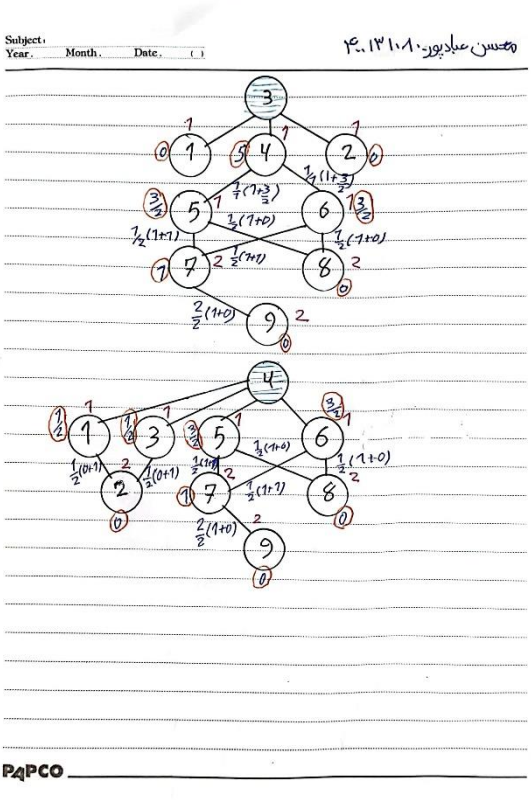


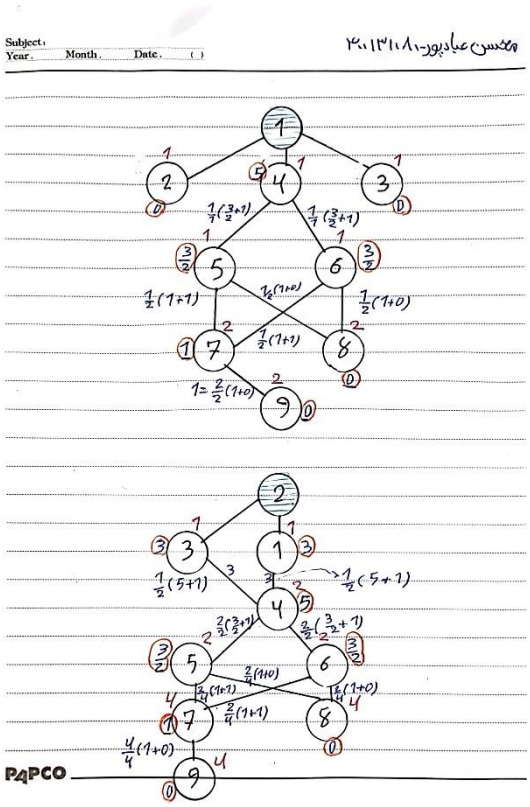
فهرست پاسخ ها

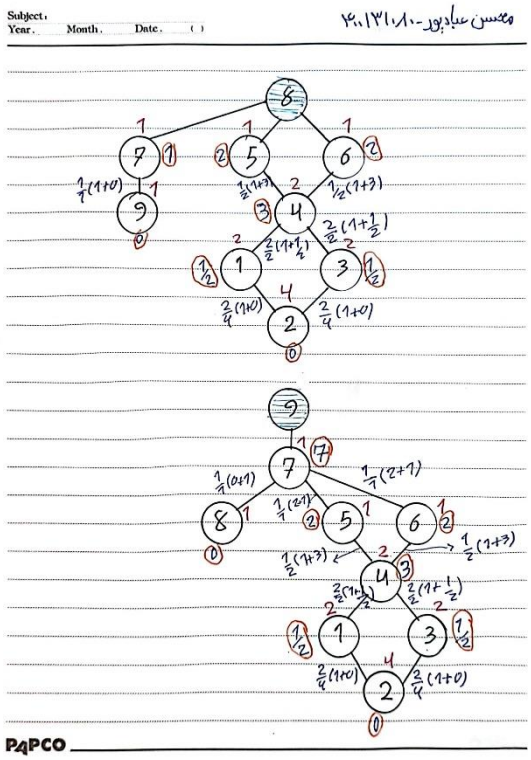
مسئله اول

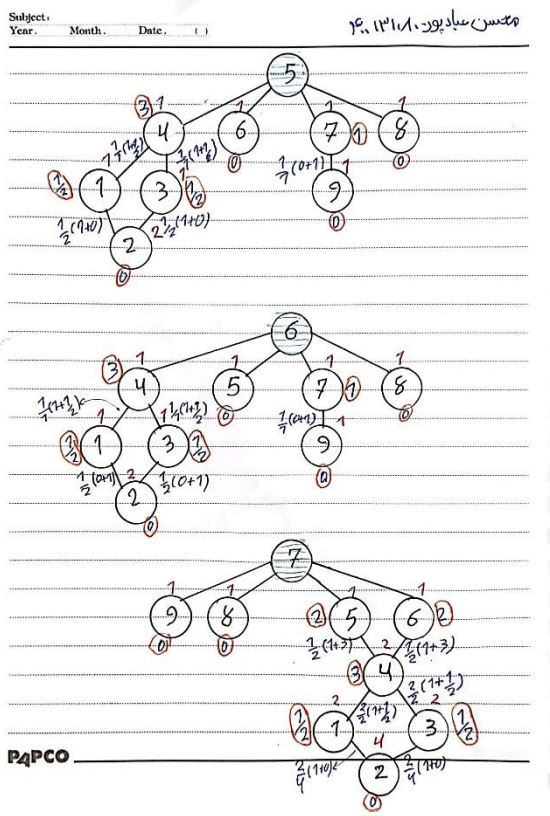
مسئله اول: الف) محاسبه edge betweenness centrality

برای پاسخ گویی به این سوال ابتدا هر یک از نود ها گراف را راس قرار داده و سپس یک DAG از آن می سازیم بطوری که بین نود های هم سطح یالی وجود نداشته باشد. سپس دو گام forward و backward را اعمال می‌کنیم. در محاسبات زیر، اعداد قرمز رنگ روی نودها نشان دهنده‌ی فاز forward بوده و تعداد طول کوتاه ترین مسیر را نمایش می دهد. اعدادی که دور آن در نودها خط کشیده شده است، dependency score مربوط به هر نود را نمایش می دهد که حاصل فاز backward می باشد که از مجموع dependency score فرزندان خود بدست آمده است. مقدار dependency score که از هر یال عبور می کند تا به نود والد برسد، dependency score آن یال در نظر گرفته می شود که محاسبات مربوط برای هر گراف روی یال مربوط نوشته شده است.





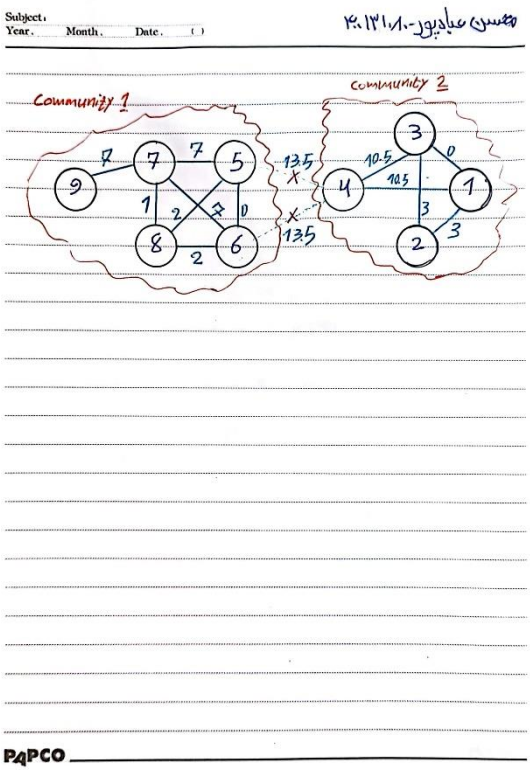




اکنون برای محاسبه‌ی edge betweenness هر یک از یال‌ها، مجموع مقادیر هر یال را در گراف های فوق حساب می‌کنیم که جدول زیر حاصل می شود(فایل ایکسل مربوط به جدول ضمیمه شده است):

Edge / Source Node	1-3	1-2	1-4	2-3	3-4	4-5	4-6	5-6	5-7	5-8	6-7	6-8	7-8	7-9
#1	0	0	0	0	0	2.5	2.5	0	1	0.5	1	0.5	0	1
#2	0	0	3	0	3	2.5	2.5	0	1	0.5	1	0.5	0	1
#3	0	0	0	0	0	2.5	2.5	0	1	0.5	1	0.5	0	1
#4	0	0.5	0	0.5	0	0	0	0	1	0.5	1	0.5	0	1
#5	0	0.5	1.5	0.5	1.5	0	0	0	0	0	0	0	0	1
#6	0	0.5	1.5	0.5	1.5	0	0	0	0	0	0	0	0	1
#7	0	0.5	1.5	0.5	1.5	2	2	0	0	0	0	0	0	0
#8	0	0.5	1.5	0.5	1.5	2	2	0	0	0	0	0	0	1
#9	0	0.5	1.5	0.5	1.5	2	2	0	3	0	3	0	1	0
Sum	0	3	10.5	3	10.5	13.5	13.5	0	7	2	7	2	1	7

بر اساس جدول فوق و مقادیر edge betweenness centrality هر یک از یال های گراف، حد آستانه‌ی ۱۲ را در نظر گرفته و یال های با بیشتر از این مقدار را هرس می کنیم تا community های موجود در گراف مشخص شود که شکل زیر از گراف بدست می آید که شامل دو جامعه می باشد یال های حذف شده یال ها ۴-۵ و ۴-۶ می باشد.



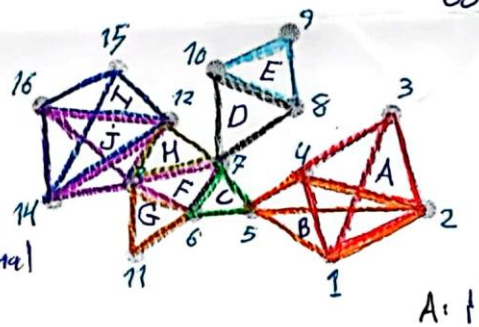
مسئله اول: ب) استخراج جوامع با الگوریتم Clique Percolation

برای الگوریتم Clique percolation ابتدا بایستی maximal cliques ها را استخراج نمود. طبق تعریف maximal cliques و توضیحات حین تدریس در کلاس درسی، نبایستی هر کلیک استخراجی زیر مجموعه‌ی کلیک دیگری باشد. استخراج کلیک توسط دو مجموعه Q و R انجام می پذیرد که ترتیب اعمال(انتخاب نود و اضافه کردن به صف) بصورت ترتیب lexical می باشد. برای این هدف، مجموعه Q اعضای کلیک استخراجی را نمایش داده و مجموعه R همسایه‌های مشترک اعضای Q را نمایش می دهد. هر بار یک عضو از R (بصورت lexical) انتخاب و به Q افزوده می شود و سپس اعضای R بروزرسانی می شود و این روند تا زمانی که R تهی شود ادامه می‌یابد؛ وقتی که تهی شد مجموعه‌ی Q یک maximal clique را نمایش می دهد. در این حین کلیک استخراجی را ذخیره و بصورت backtrack جست و جو را برای یافتن ادامه میدهیم. در تصویر زیر برای نمونه محاسبات

مربوط به یافتن یک maximal clique آورده شده است(A: 1,2,3,4)؛ اما از جایی که گراف کوچک بوده و کاملا بصورت عینی و شهودی میتوان سایر maximal cliques ها را استخراج نمود، از نوشتن محاسبات مجموعه‌ای برای هر یک صرف نظر شده و به یک نمونه اتکا شده است. در نهایت ۱۰ مورد maximal clique حاصل شده است که بترتیب از A تا J نام گذاری شده و اعضای هر کلیک با نود های شماره گذاری شده در تصویر زیر آورده شده است:

ب) با استفاده از الگوریتم Clique Percolation، Communityهای موجود در گراف زیر را مشخص کنید.

محسن عابدپور - ۱۴۰۱۳۱۰۸



steps to find a maximal cliques:

1 $\begin{cases} Q=\{1\} \\ R=\{2,3,4,5\} \end{cases}$

2 $\begin{cases} Q=\{1,2\} \\ R=\{2,3,4,5\} \cap \Gamma\{2\}=\{3,4,5\} \end{cases}$

3 $\begin{cases} Q=\{1,2,3\} \\ R=\{3,4,5\} \cap \Gamma\{3\}=\{4\} \end{cases}$

4 $\begin{cases} Q=\{1,2,3,4\} \\ R=\emptyset \end{cases}$ * → should backtrack

first maximal cliques found

A: {1,2,3,4}
B: {1,2,4,5}
C: {5,6,7}
D: {7,8,10}
E: {8,9,10}
F: {6,7,13}
G: {6,11,13}
H: {7,12,13}
I: {12,14,15,16}
J: {12,14,13,16}

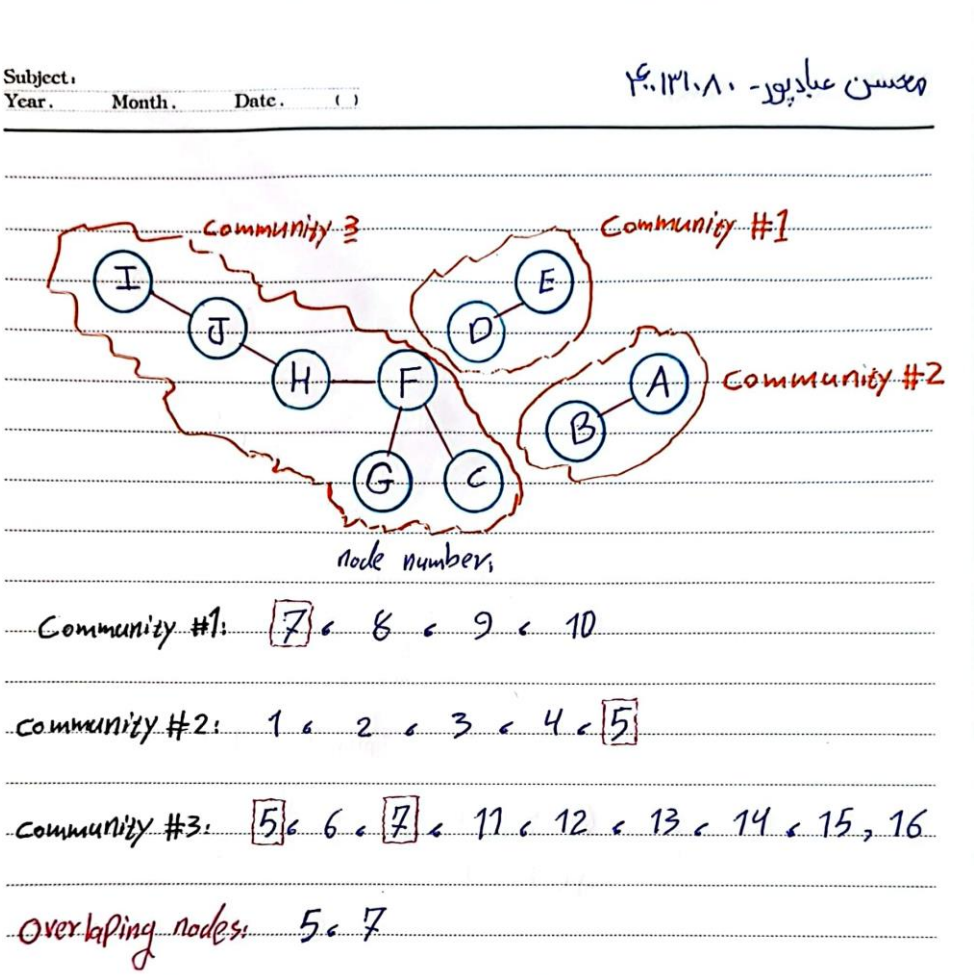
حال بر اساس کلیک های استخراجی بایستی ماتریس مجاورت کلیک‌ها را ایجاد نماییم که در آن هر کلیک نشانگر یک نود جدید (super-node) می باشد؛ در این ماتریس تعداد نود های مشترک بین دو کلیک در جدول در می‌شود. تصویر این ماتریس در شکل زیر آورده شده است(فایل ایکسل نیز ضمیمه شده است). همانطور که قابل انتظار است این ماتریس مجاورت نیز متقارن می‌باشد.

	K=3 | Remove below 2						Mohsen Ebadpour | 400131080			
Max. Cliques	A	B	C	D	E	F	G	H	I	J
A	4	3	0	0	0	0	0	0	0	0
B	3	4	1	0	0	0	0	0	0	0
C	0	1	3	1	0	2	1	1	0	0
D	0	0	1	3	2	1	0	1	0	0
E	0	0	0	2	3	0	0	0	0	0
F	0	0	2	1	0	3	2	2	0	1
G	0	0	1	0	0	2	3	1	0	1
H	0	0	1	1	0	2	1	3	1	2
I	0	0	0	0	0	0	0	1	4	3
J	0	0	0	0	0	1	1	2	3	4

حال بایستی یک مقدار برای K انتخاب و یالِ بین کلیک‌هایی که کمتر از K-1 نود مشترک دارند را حذف کنیم. بنده K را برابر با 3 در نظر گرفته و کلیک هایی که 2 یا بیشتر از آن نود مشترک دارند را متصل و کلیک هایی که 1 یا 0 نود مشترک دارند را قطع در نظر میگیرم که در جدول فوق مقادیر 1 همگی به صفر و سایر مقادیر به یک تبدیل می‌شود تا ماتریس مجاورت بین کلیک‌ها حاصل شود. ماتریس جدید بصورت زیر حاصل می‌شود:

	K=3 | Remove below 2						Mohsen Ebadpour | 400131080			
Max. Cliques	A	B	C	D	E	F	G	H	I	J
A	1	1	0	0	0	0	0	0	0	0
B	1	1	0	0	0	0	0	0	0	0
C	0	0	1	0	0	1	0	0	0	0
D	0	0	0	1	1	0	0	0	0	0
E	0	0	0	1	1	0	0	0	0	0
F	0	0	1	0	0	1	1	1	0	0
G	0	0	0	0	0	1	1	0	0	0
H	0	0	0	0	0	1	0	1	0	1
I	0	0	0	0	0	0	0	0	1	1
J	0	0	0	0	0	0	0	1	1	1

حال گراف متناظر با ماتریس مجاور فوق را رسم میکنیم که در آن هر نود نشانگر یک maximal clique استخراجی می باشد؛ هر جزء گراف نشانگر یک community میباشد.



در نتیجه در گراف موجود، سه جامعه شناسایی می شود که اعضای هر جامعه در محاسبات فوق مشخص می شود. دو نود همپوشان بین جوامع وجود دارد که مشخص شده است. شماره گذاری نودهای گراف در تصویر اول انجام شده است.

مسئله دوم

مسئله دوم: الف) محاسبه مقادیر Centrality ها

برای محاسبه‌ی هر کدام از centrality های مذکور در متن سوال، یک تابع بصورت مستقل پیاده سازی شده است که گراف مورد نظر را ورودی گرفته و یک دیکشنری بر میگرداند که در آن key متناظر با ID نود و Value برابر با مقدار centrality متناظر می‌باشد. دیکشنری ارسالی بر اساس مقادیر centrality از زیاد به کم مرتب شده می‌باشد تا در زمان رسم نمودار ها و ذخیره نتایج با چالش مواجه نباشیم و بتوانیم فایل های خواسته شده را بدرستی خروجی دهیم. عناوین توابع پیاده سازی شده برای centrality ها بترتیب متناظر با ClosenessCentralityScratch، EfficiencyCenteralityScratch، DegreeCentralityScratch و KatzCentralityScratch می باشد. در زمان پیاده سازی اطمینان از صحت پیاده سازی و مقادیر حاصل، نتایج با توابع آماده‌ی networkx مقایسه شده و در زمان گرفتن خروجی comment شده است. یک نمونه از توابع پیاده سازی شده در زیر آورده شده است:

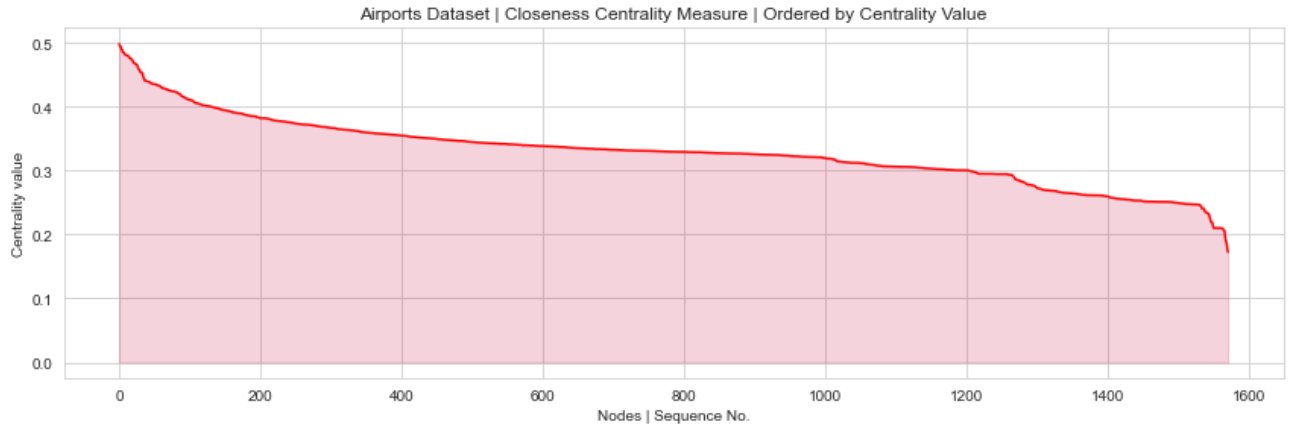
```python
def ClosenessCentralityScratch(graph):
    connected_components_sorted = sorted(nx.connected_components(graph), key=len, reverse=True)
    giant_graph = graph.subgraph(connected_components_sorted[0])
    giant_nodes = giant_graph.nodes()

    closeness_centrality = {}
    for node in tqdm(giant_nodes):
        shortest_paths = nx.shortest_path_length(giant_graph,source=node)
        sum_shortest_paths = 0
        for key in shortest_paths:
            sum_shortest_paths += shortest_paths[key]
        closeness_centrality[node] = (len(giant_nodes))/sum_shortest_paths
    closeness_centrality = dict(sorted(closeness_centrality.items(), key=lambda item: -item[1]))
    print(min(closeness_centrality.values()),max(closeness_centrality.values()))

    """
    # this code is for checking the correctness of above code
    closeness_centrality_2 = nx.closeness_centrality(giant_graph)
    closeness_centrality_2 = dict(sorted(closeness_centrality_2.items(), key=lambda item: -item[1]))
    print(min(closeness_centrality_2.values()),max(closeness_centrality_2.values()))
    """
    return closeness_centrality
```

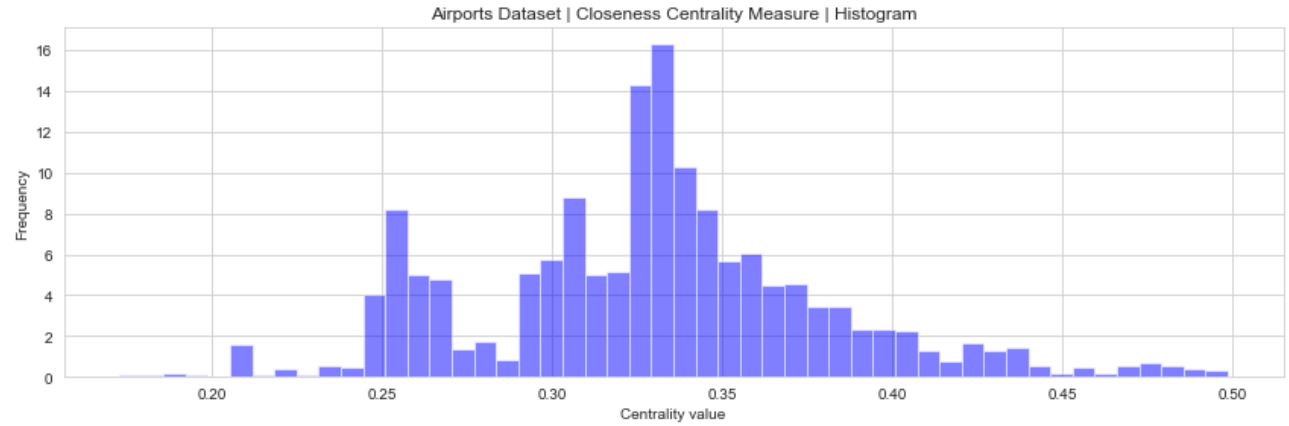
برای هر گراف و هر معیار یک فایل با پسوند txt. در پوشه P02 (A) قرار داده شده است که نام هر یک شامل مجموعه داده‌ی متناظر و معیار گزارش شده می باشد که با فرمت مقابل ذخیره شده است: "P2_"+dataset_name+measure_name

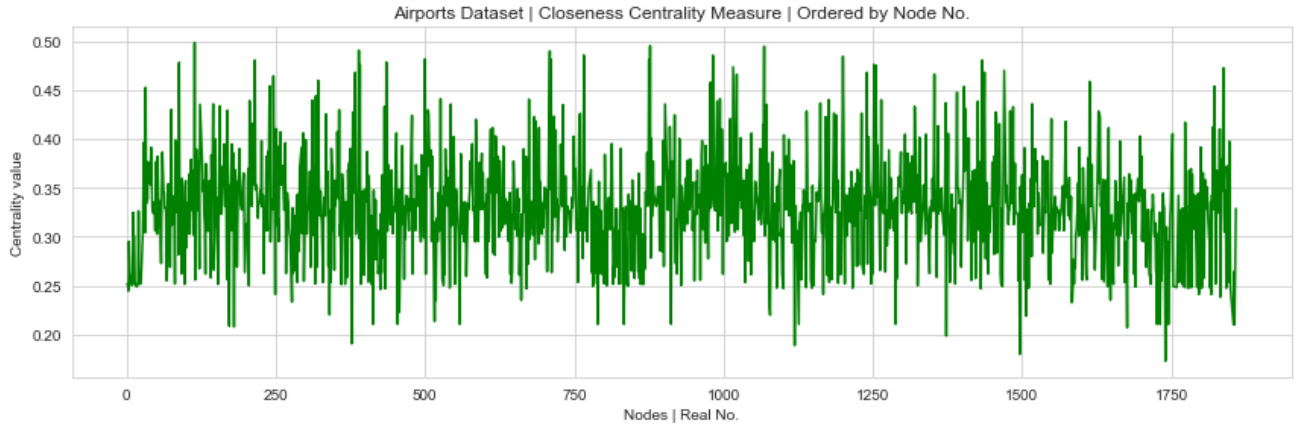
مسئله دوم: ب) رسم نمودار های Centrality | رسم هیستوگرام برای تحلیل بیشتر (فعالیت اضافی)

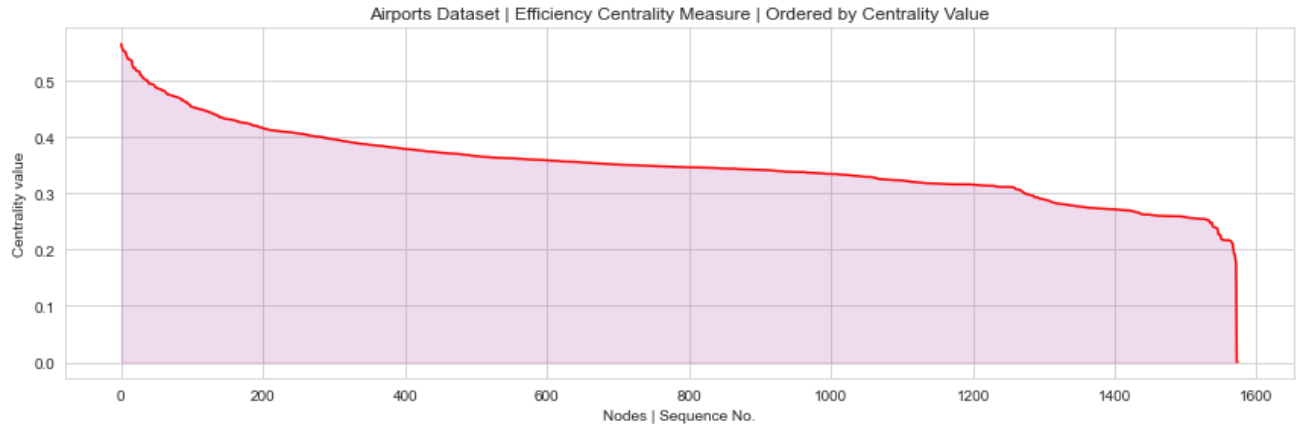
برای هر یک از معیار های چهارگانه‌ی خواسته شده در هر گراف سه نمودار رسم شده است. در نمودار اول، مقادیر مرکزیت بصورت نزولی مرتب شده و نشان داده شده است. محور عمودی آن مقدار مرکزیت و محور افقی آن صرفا یک شمارش ترتیبی به تعداد نود ها می باشد(ربطی به شماره نود متناظر با آن ندارد). در نمودار دوم هیستوگرام مقادیر حاصل از مرکزیت در ۵۰ بازه رسم شده است تا بتوان در صورت امکان برای تحلیل در قسمت بعد استفاده نمود. نمودار سوم نیز مقدار مرکزیت هر نود را نشان می دهد که در آن محور افقی دقیقا شماره نود واقعی را نشان داده و محور عمودی نیز مقدار مرکزیت را نشان می دهد.

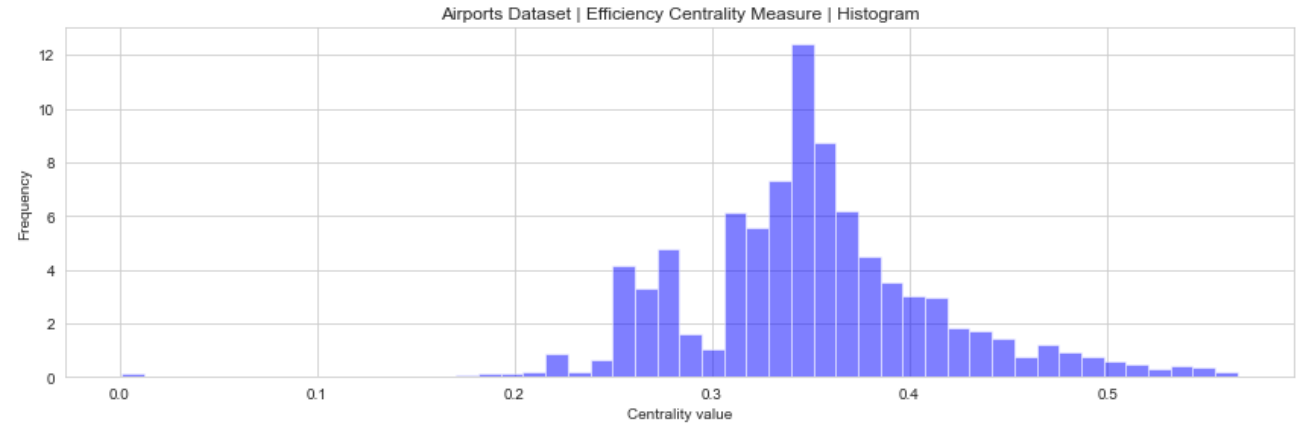
در صفحات بعدی بترتیب معیار های Closeness، Efficiency، Degree و Katz گزارش شده است که ابتدا برای گراف airports و سپس برای گراف Bible آورده شده است. همچنین فایل نمودار ها نیز در پوشه P02 (B) قرار داده شده است. لازم به ذکر است که در پیاده سازی انجام شده، مقدار مرکزیت degree بر اساس تعداد نود های گراف نرمال شده است.

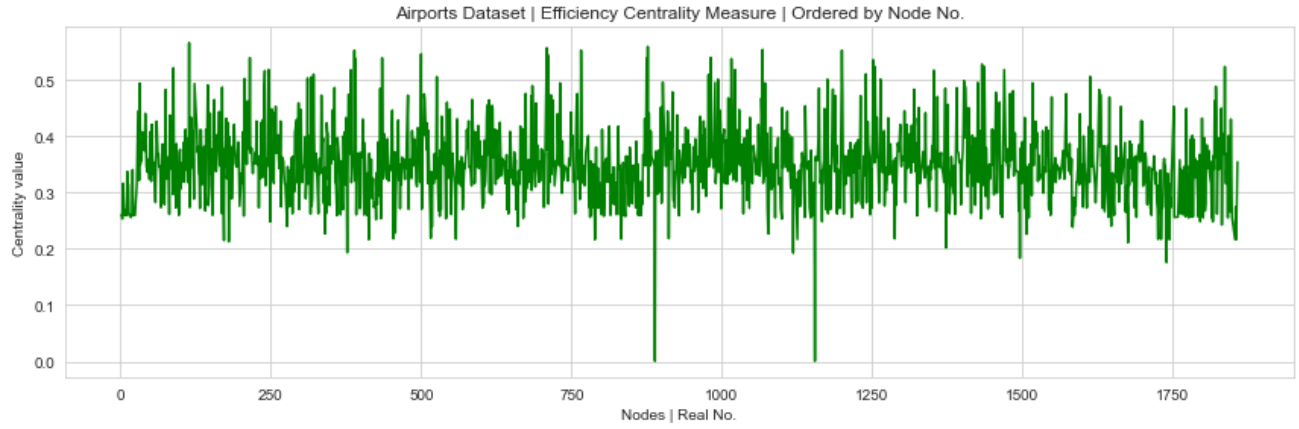

Airports Dataset | Closeness Centrality Measure | Ordered by Centrality Value
Centrality value
Nodes | Sequence No.

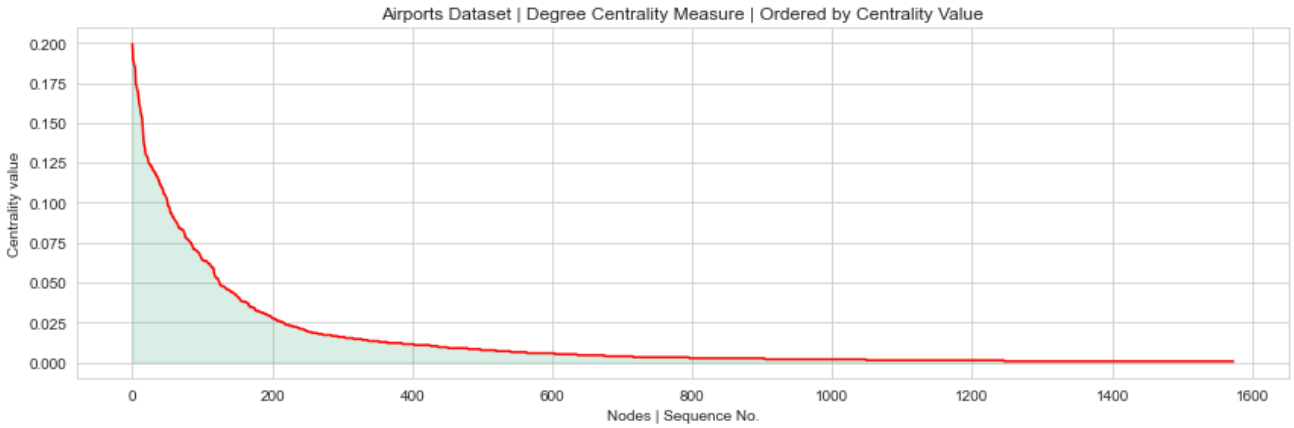

Airports Dataset | Closeness Centrality Measure | Histogram
Frequency
Centrality value

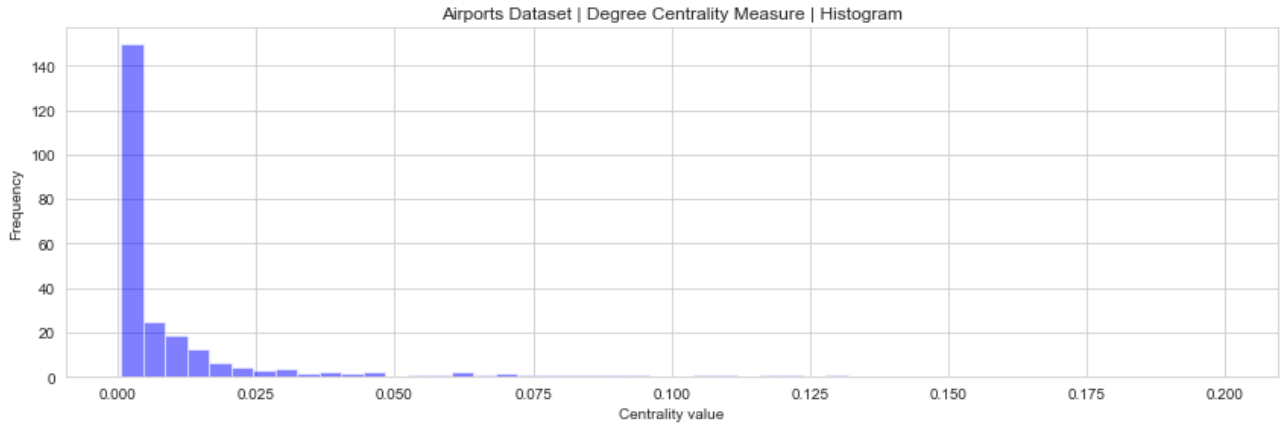

Airports Dataset | Closeness Centrality Measure | Ordered by Node No.
Centrality value
Nodes | Real No.

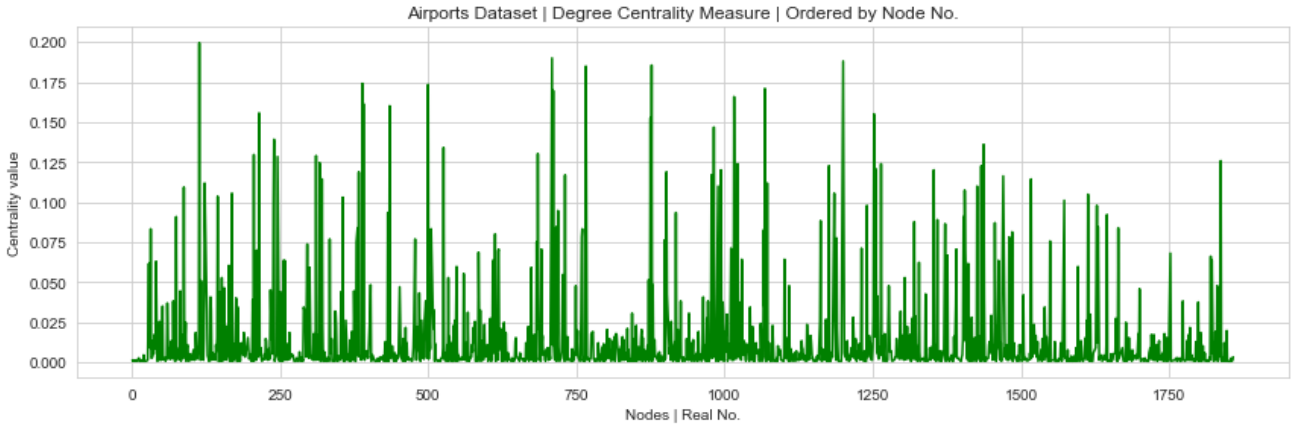

Airports Dataset | Efficiency Centrality Measure | Ordered by Centrality Value
Centrality value
0.0
0.1
0.2
0.3
0.4
0.5
0
200
400
600
800
1000
1200
1400
1600
Nodes | Sequence No.

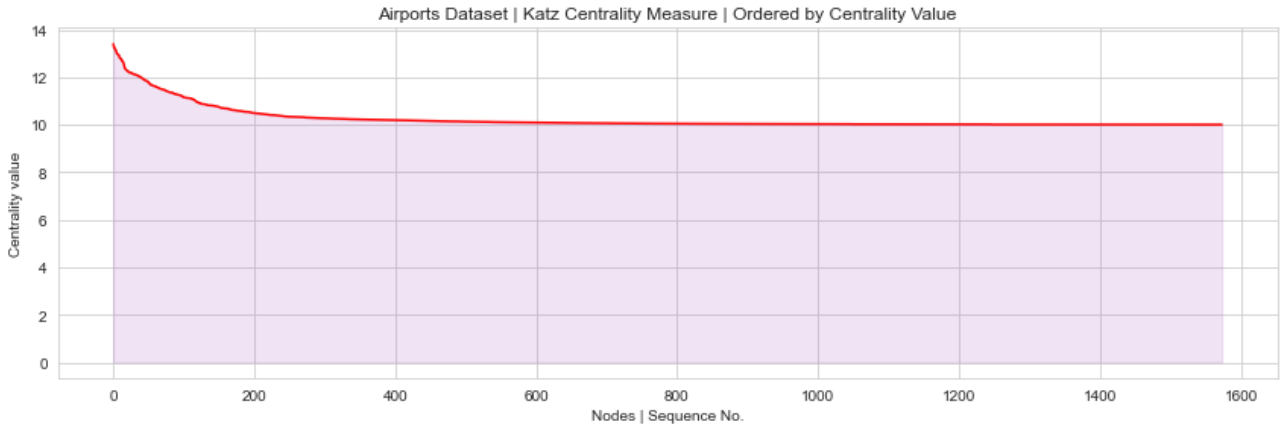

Airports Dataset | Efficiency Centrality Measure | Histogram
Frequency
0
2
4
6
8
10
12
0.0
0.1
0.2
0.3
0.4
0.5
Centrality value

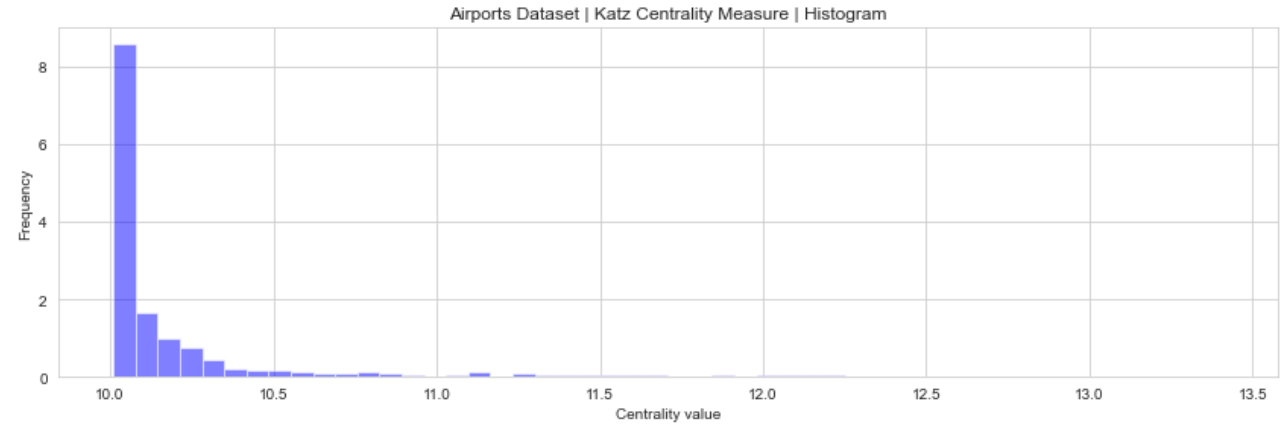

Airports Dataset | Efficiency Centrality Measure | Ordered by Node No.
Centrality value
0.0
0.1
0.2
0.3
0.4
0.5
0
250
500
750
1000
1250
1500
1750
Nodes | Real No.

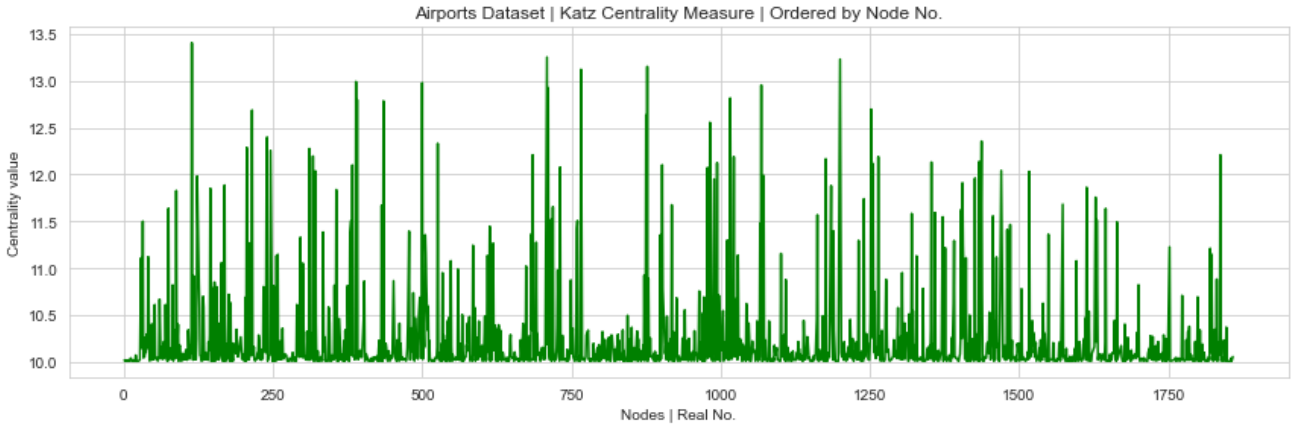

Airports Dataset | Degree Centrality Measure | Ordered by Centrality Value
Centrality value
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0
200
400
600
800
1000
1200
1400
1600
Nodes | Sequence No.

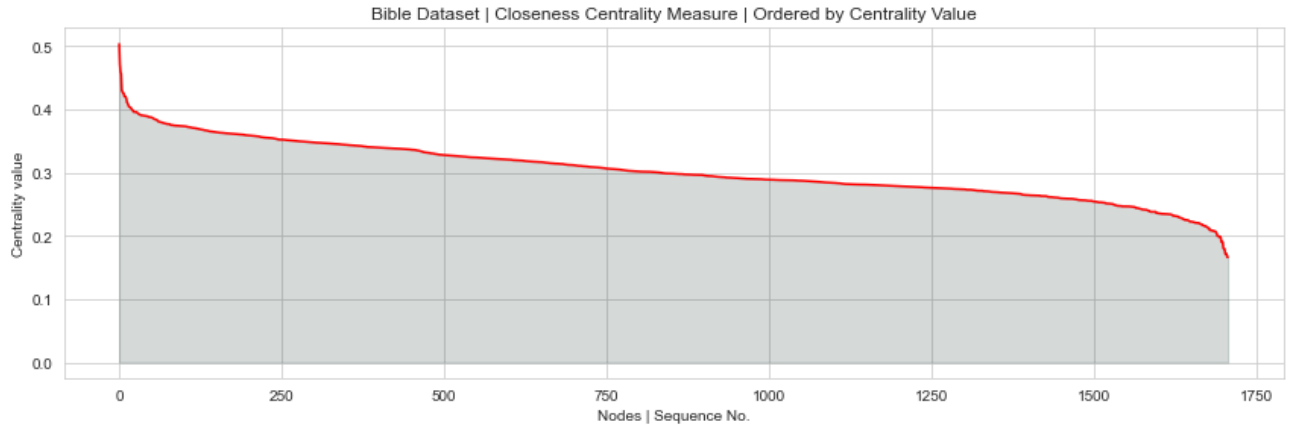

Airports Dataset | Degree Centrality Measure | Histogram
Frequency
140
120
100
80
60
40
20
0
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
Centrality value

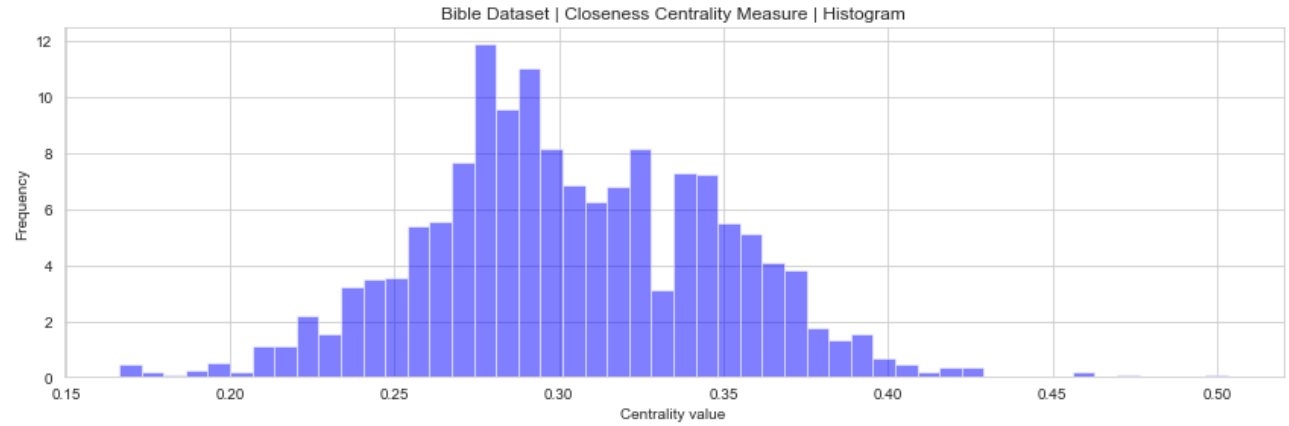

Airports Dataset | Degree Centrality Measure | Ordered by Node No.
Centrality value
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0
250
500
750
1000
1250
1500
1750
Nodes | Real No.

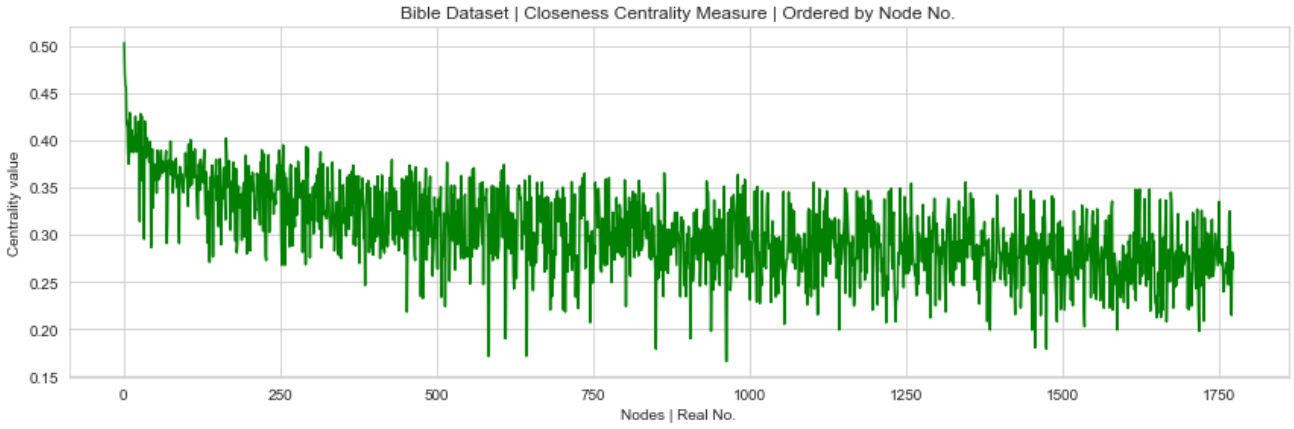

Airports Dataset | Katz Centrality Measure | Ordered by Centrality Value
Centrality value
Nodes | Sequence No.

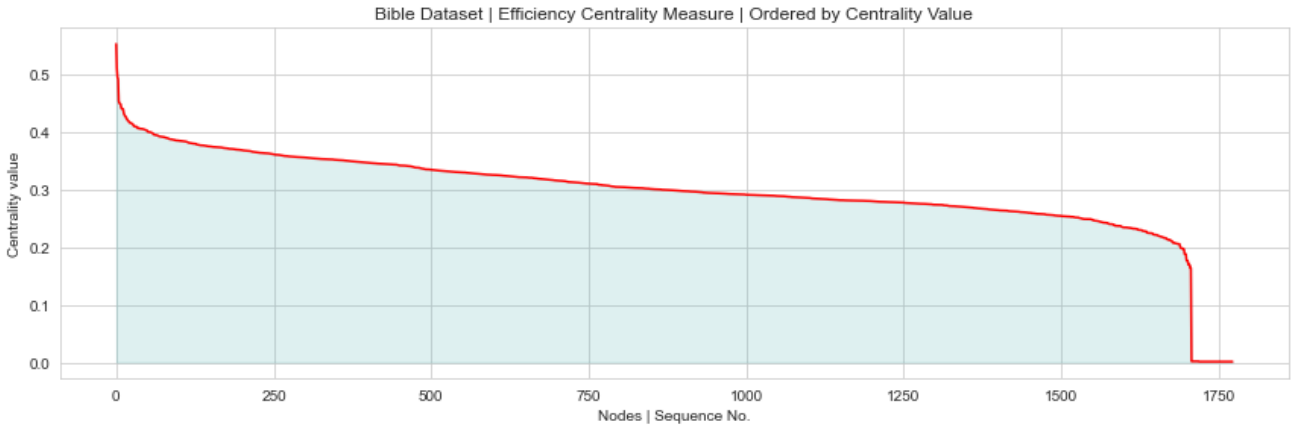

Airports Dataset | Katz Centrality Measure | Histogram
Frequency
Centrality value

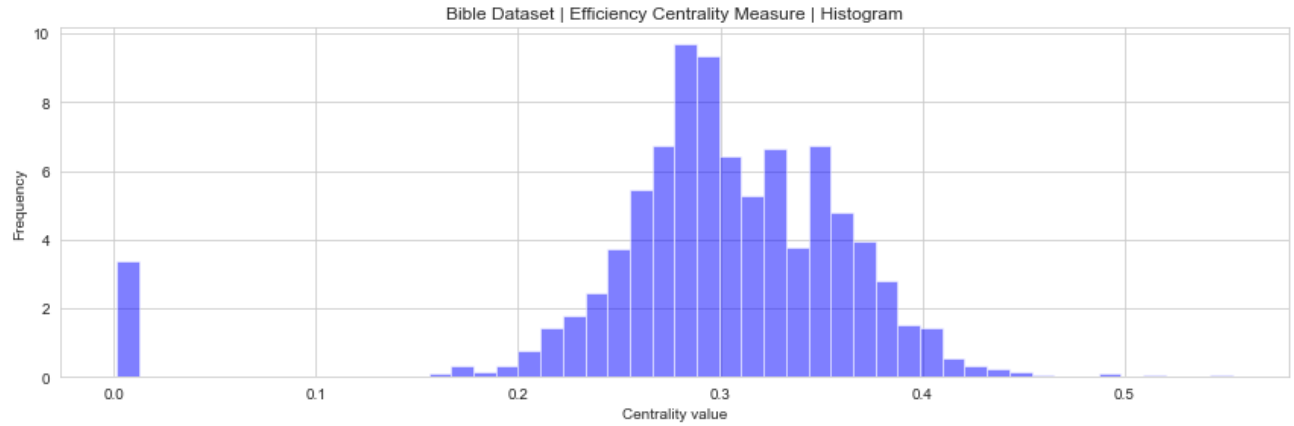

Airports Dataset | Katz Centrality Measure | Ordered by Node No.
Centrality value
Nodes | Real No.

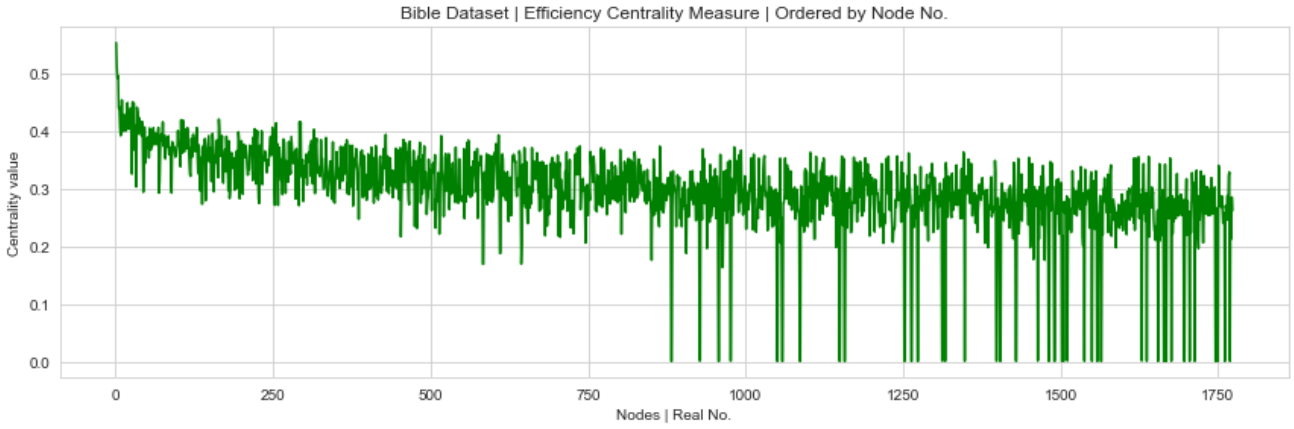

Bible Dataset | Closeness Centrality Measure | Ordered by Centrality Value
Centrality value
Nodes | Sequence No.

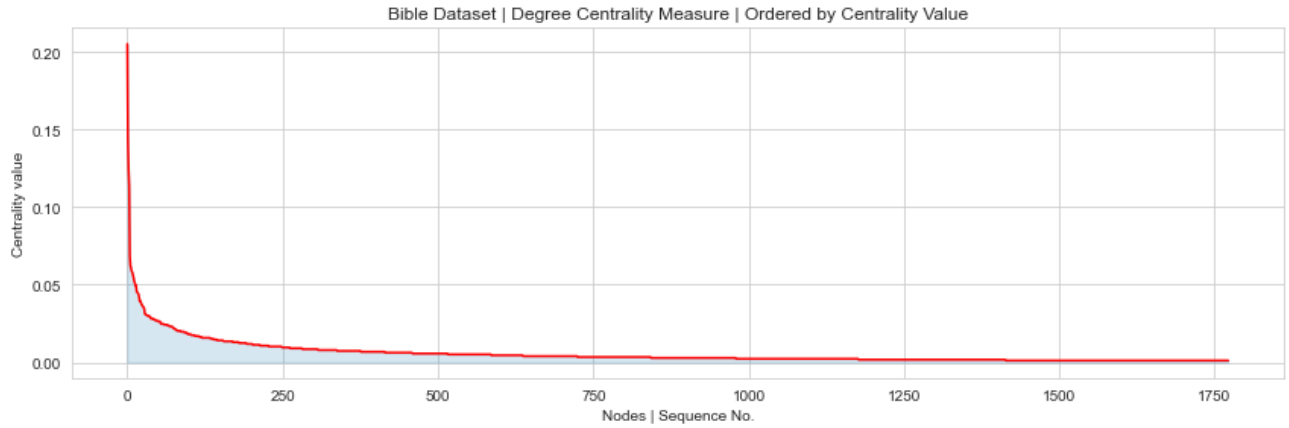

Bible Dataset | Closeness Centrality Measure | Histogram
Frequency
Centrality value

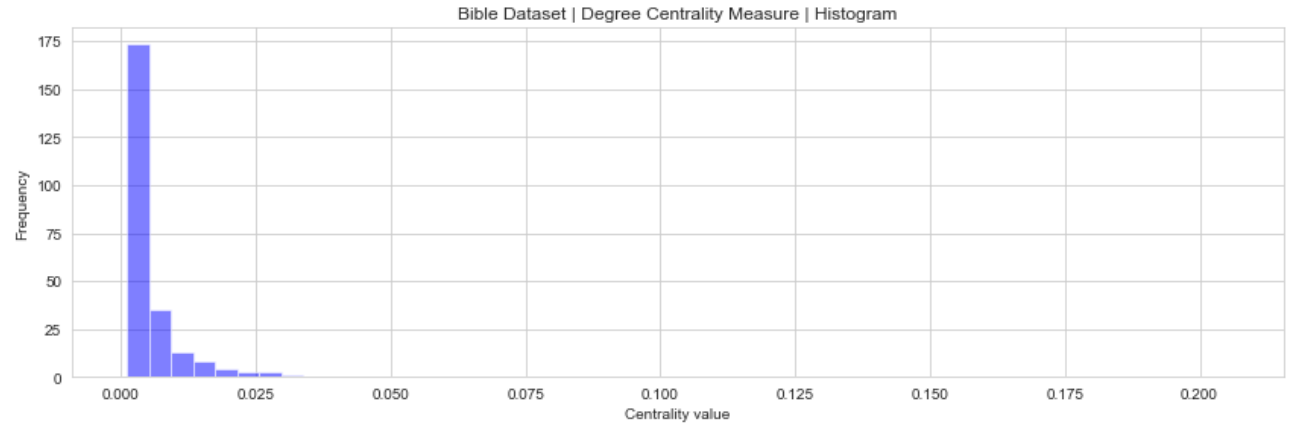

Bible Dataset | Closeness Centrality Measure | Ordered by Node No.
Centrality value
Nodes | Real No.

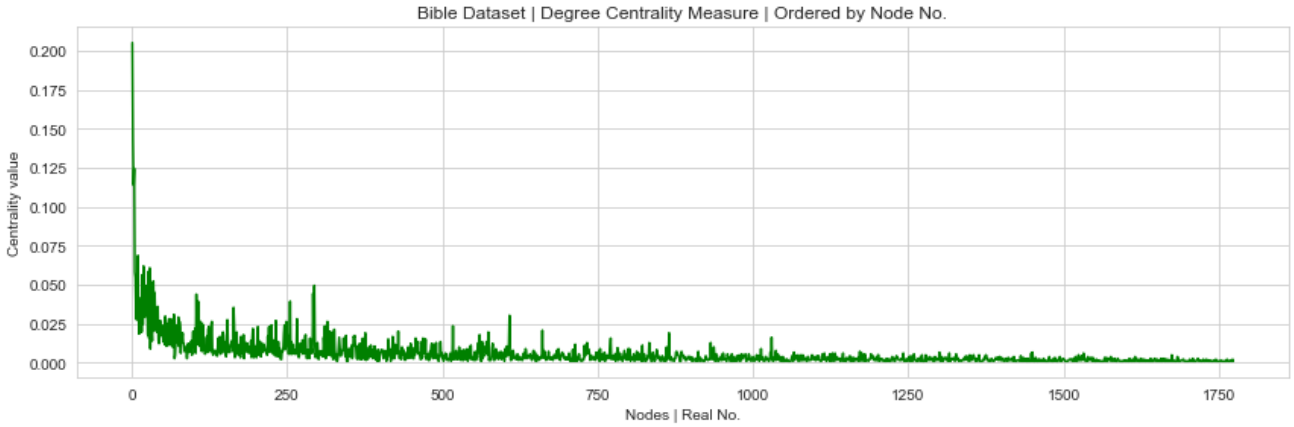

Bible Dataset | Efficiency Centrality Measure | Ordered by Centrality Value
Centrality value
0.0
0.1
0.2
0.3
0.4
0.5
0
250
500
750
1000
1250
1500
1750
Nodes | Sequence No.

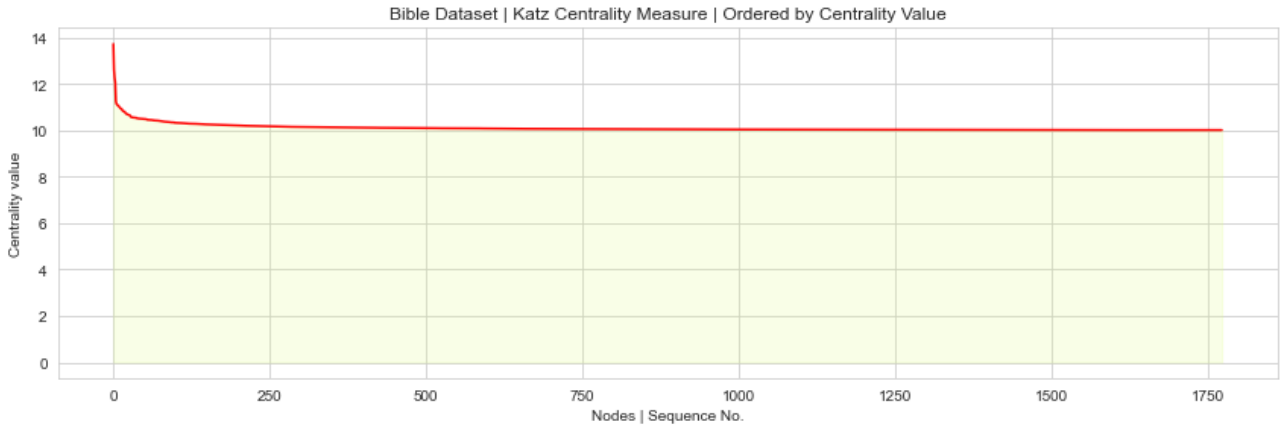

Bible Dataset | Efficiency Centrality Measure | Histogram
Frequency
0
2
4
6
8
10
0.0
0.1
0.2
0.3
0.4
0.5
Centrality value

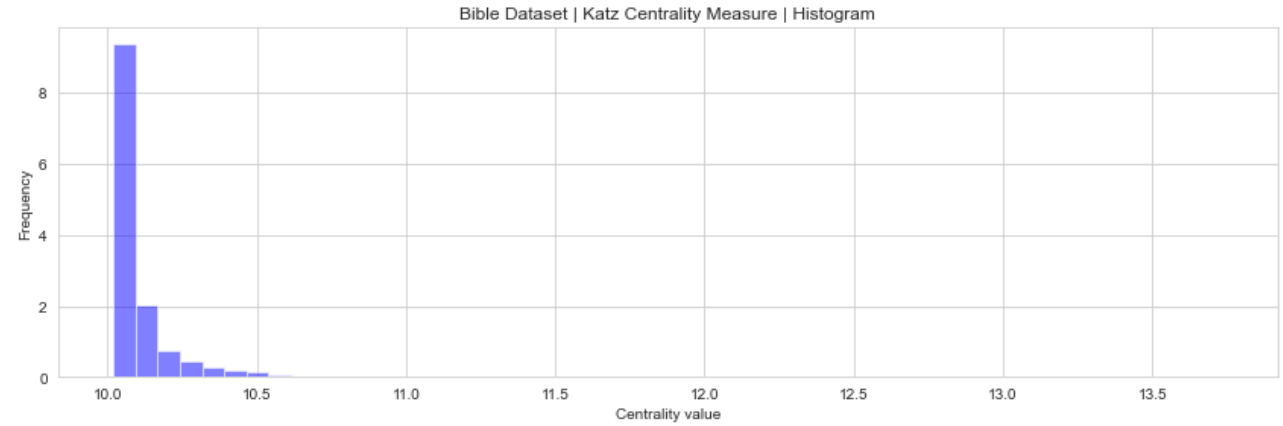

Bible Dataset | Efficiency Centrality Measure | Ordered by Node No.
Centrality value
0.0
0.1
0.2
0.3
0.4
0.5
0
250
500
750
1000
1250
1500
1750
Nodes | Real No.

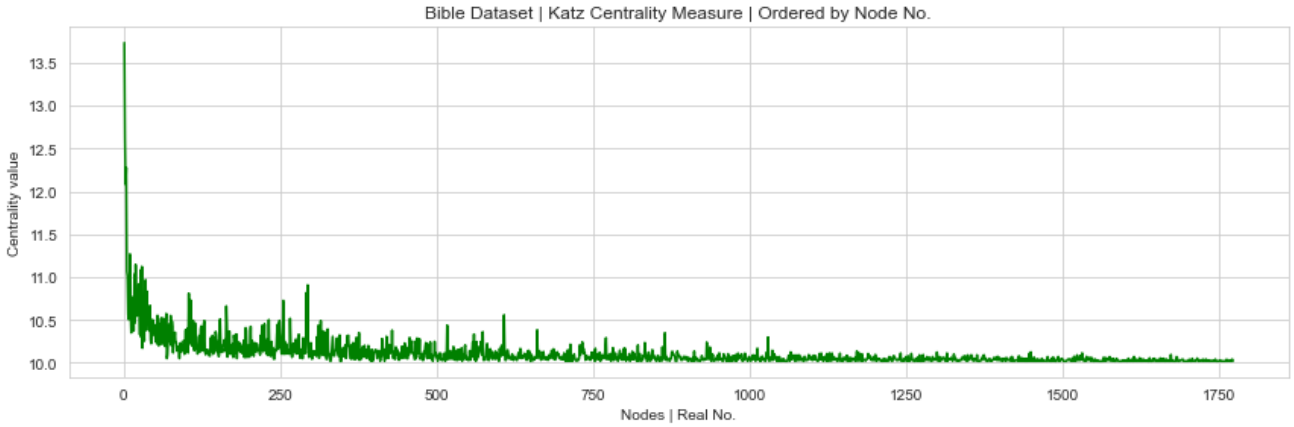

Bible Dataset | Degree Centrality Measure | Ordered by Centrality Value
Centrality value
0.20
0.15
0.10
0.05
0.00
0
250
500
750
1000
1250
1500
1750
Nodes | Sequence No.


Bible Dataset | Degree Centrality Measure | Histogram
Frequency
175
150
125
100
75
50
25
0
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
Centrality value


Bible Dataset | Degree Centrality Measure | Ordered by Node No.
Centrality value
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0
250
500
750
1000
1250
1500
1750
Nodes | Real No.


Bible Dataset | Katz Centrality Measure | Ordered by Centrality Value
Centrality value
Nodes | Sequence No.


Bible Dataset | Katz Centrality Measure | Histogram
Frequency
Centrality value


Bible Dataset | Katz Centrality Measure | Ordered by Node No.
Centrality value
Nodes | Real No.

مسئله دوم: ج) تحلیل نمودار های Centrality

ابتدا برای هر یک از معیارهای چهارگانه تحلیل استخراجی و نکات جالب مورد نظر اشاره شده و در نهایت برای هر گراف موارد مشاهده ذکر خواهد شد(این مورد در قسمت (ه) آورده می شود).

- **Closeness:** در هر دو گراف داده شده، تعداد اندکی نود وجود دارد که معیار آنان بالاتر از بقیه می‌باشد و هم چنین تعداد نود اندکی وجود دارد که معیار آنان کمتر از بقیه می باشد و این در حالی است که اکثریت نود ها معیار نزدیک بهمی دارند. این مورد نشان می دهد که با این معیار میتوان تعداد اندکی نود اما با تمایز بالا از متوسط نودها را شناسایی نمود. بر اساس توضیحات ارائه شده میتوان انتظار داشت که توزیع این معیار از خانواده توزیع های نرمال باشد که نمودار هیستوگرام نیز این مورد را تقریبا ثابت می‌کند. در محاسبات این معیار صرفا نود ها giant component در نظر گرفته شده است.
- **Efficiency:** برای این معیار نیز همانند معیار Closenes و در هر دو گراف، تعداد اندکی نود وجود دارد که معیار آنان بالاتر از بقیه می‌باشد و هم چنین تعداد نود اندکی وجود دارد که معیار آنان کمتر از بقیه می باشد و این همانند closeness در حالی است که اکثریت نود ها مقدار نزدیک و شبیه بهمی دارند. این نکته نشان می دهد که با این معیار میتوان تعداد اندکی نود اما با تمایز بالا از میانگین نود ها را شناسایی نمود. بر اساس موارد مذکور میتوان قابل ملاحظه است که توزیع این معیار از خانواده توزیع های نرمال بوده و هیستوگرام نیز این مورد را نشان میدهد. همچنین بر اساس شباهت روابط بین این معیار و closeness و تفاوت در عبارت 1/(n-1) که میتوان آن به عنوان یک term نرمالساز در نظر گرفت،

میتوان شباهت رفتاری نمودار های این دو معیار را توجیه نمود. در محاسبات این معیار همه نود ها در نظر گرفته شده است.

- **Degree:** این معیار که درجه هر نود را به عنوان مرکزیت آن در نظر میگیرد، نشان میدهد که توزیع درجه گراف های داده شده که از جنس گراف های دنیای واقعی هستند از توزیع power-law پیروی می کنند و نشان می دهند که هر چند اندک اما نود هایی با درجه بسیار بالا در گراف وجود دارد.
- **Katz:** دو مورد را می توان از نمودار های مربوط به این معیار استخراج کرده و تحلیل نمود، اولین مورد، پیروی این معیار از توزیع power-law می باشد که میتوان دلیل آن را استفاده از داده های ماتریس مجاورت (به عبارتی داده های درجه) در محاسبه این معیار توجیه نمود. (مورد مذکور در حد حدس و گمان اولیه و نکته جالب و برداشت شخصی بوده و برای بررسی دقیق تر نیازمند تحقیق و اثبات می باشد)

دومین مورد در خصوص این معیار، تعداد اندکی نود وجود دارد که معیار آنان بالاتر از بقیه می‌باشد و این در حالی است که اکثریت نود ها معیار نزدیک بهمی دارند. این مورد نشان می دهد که با این معیار میتوان تعداد اندکی نود اما با تمایز بالا را از میانگین نودها را شناسایی نمود. (در معیار های closeness و efficiency هم تعدادی نود بالاتر از میانگین و هم تعدادی پایین تر از میانگین وجود دارد که باعث می شود از دو طرف بتوان نودهای متفاوت را شناسایی کرده و بین آنان تمییز قائل شد که این در حالی است که در معیار katz و بواسطه‌ی توزیع power-law صرفا می توان تعدادی نود اندک را از بالا شناسایی نمود.

مسئله دوم: د) رسم جدول ها

ابتدا برای هر یک از گراف‌ها و برای هر معیار ۱۰ گره با اهمیت و مرکزیت بالا انتخاب شده و در کنار یکدیگر به یک دیکشنری زبان پایتون تبدیل کرده و سپس دیکشنری را به یک DataFrame از کتابخانه pandas تبدیل می‌کنیم تا با چاپ DataFrame بتوان جدول خواسته شده را خروج داد؛ جداول زیر حاصل می شود:

	Closeness Centrality	Efficiency Centrality	Degree Centrality	Katz Centrality
1	ATL | Atlanta, GA: Hartsfield-Jackson Atlanta ...	ATL | Atlanta, GA: Hartsfield-Jackson Atlanta ...	ATL | Atlanta, GA: Hartsfield-Jackson Atlanta ...	ATL | Atlanta, GA: Hartsfield-Jackson Atlanta ...
2	LAX | Los Angeles, CA: Los Angeles International	LAX | Los Angeles, CA: Los Angeles International	IAD | Washington, DC: Washington Dulles Intern...	IAD | Washington, DC: Washington Dulles Intern...
3	MSP | Minneapolis, MN: Minneapolis-St Paul Int...	IAD | Washington, DC: Washington Dulles Intern...	ORD | Chicago, IL: Chicago O'Hare International	ORD | Chicago, IL: Chicago O'Hare International
4	DEN | Denver, CO: Denver International	MSP | Minneapolis, MN: Minneapolis-St Paul Int...	LAX | Los Angeles, CA: Los Angeles International	LAX | Los Angeles, CA: Los Angeles International
5	IAD | Washington, DC: Washington Dulles Intern...	JFK | New York, NY: John F. Kennedy International	JFK | New York, NY: John F. Kennedy International	JFK | New York, NY: John F. Kennedy International
6	JFK | New York, NY: John F. Kennedy International	ORD | Chicago, IL: Chicago O'Hare International	DEN | Denver, CO: Denver International	DEN | Denver, CO: Denver International
7	MCO | Orlando, FL: Orlando International	DEN | Denver, CO: Denver International	EWR | Newark, NJ: Newark Liberty International	EWR | Newark, NJ: Newark Liberty International
8	ORD | Chicago, IL: Chicago O'Hare International	EWR | Newark, NJ: Newark Liberty International	MSP | Minneapolis, MN: Minneapolis-St Paul Int...	MSP | Minneapolis, MN: Minneapolis-St Paul Int...
9	EWR | Newark, NJ: Newark Liberty International	IAH | Houston, TX: George Bush Intercontinenta...	IAH | Houston, TX: George Bush Intercontinenta...	IAH | Houston, TX: George Bush Intercontinenta...
10	IAH | Houston, TX: George Bush Intercontinenta...	MCO | Orlando, FL: Orlando International	MIA | Miami, FL: Miami International	MIA | Miami, FL: Miami International

	Closeness Centrality	Efficiency Centrality	Degree Centrality	Katz Centrality
1	israel	israel	israel	israel
2	judah	judah	judah	judah
3	jerusalem	david	david	david
4	david	jerusalem	jerusalem	jerusalem
5	egypt	egypt	egypt	egypt
6	ephraim	ephraim	benjamin	benjamin
7	manasseh	benjamin	manasseh	manasseh
8	benjamin	manasseh	ephraim	ephraim
9	joseph	moses	saul	saul
10	moses	joseph	philistines	philistines

مسئله دوم: ۵) یافتن گره های مهم

برای نتیجه‌گیری می‌توان این گونه جمع بندی کرد که در معیار های Katz، Closeness و Efficiency هر سه می توانند در گراف های دنیای واقعی نود هایی را استخراج و شناسایی کنند که از میانگین نود ها اختلاف دارند چرا که اکثریت نود ها مقدار مرکزیت نزدیک بهمی دارند. نکته ای که قابل مشاهده است این است که بر اساس ذات گراف و مجموعه داده، هر یک از معیار های چهار گانه میتوانند خوب و تفسیر پذیر عمل کنند و بیان دقیق اینکه چه معیاری و به چه علتی برای کدام گراف بهتر است نیازمند مطالعه کاملا جزئی و پایه‌ای گراف و منشاء وجود آن است. ب

رای مثال در گراف فرودگاه ها، فرودگاهی که تعداد پرواز زیادی به سایر فرودگاه داشته و با نود های بیشتری در ارتباط است، از اهمیت زیادی برخوردار است(درجه نود) چرا که نشان می دهد آن فرودگاه یک فرودگاه ترانزیتی محسوب شده و توقف هواپیماها با احتمال بالایی در آن فرودگاه انجام می شود(یا مقصد گردشگری بالایی دارد) و آن فرودگاه می تواند مهم تلقی شود چرا که در صورت حذف، تعداد زیادی پرواز از بین خواهد رفت که در دنیای واقعی حتی قابل تصور نیست که میتواند چقدر زیانبار باشد اما در مجموعه داده اسامی نیمتوان همچین اطلاعاتی از معیار degree برای آن گراف استخراج نموده یا تحلیل کرد. برای مجموعه داده‌ی فروگاهی، در کنار degree میتوان از closeness یا Katz نیز استفاده نمود چرا که بر اساس نمودار ها تعداد اندکی نود را از بقیه نود ها تفکیک می کند که از درجه اهمیت با آنها متفاوت است. بر اساس مشاهده نموداری برای مجموعه داده اسامی میتوان معیار Katz و Efficiency را معیار های مهمی برای ایجاد تمایز و شناسایی گره های مهم در نظر گرفت چرا که تعداد نود بسیار اندکی را ارائه میکند که مرکزیت آن با میانگین سایر نود ها فاصله بالایی داشته و جداپذیری خوبی را بهمراه داشته و اهمیت بالای آن را نشان دهد.

مسئله سوم

مسئله سوم: محاسبه Eigenvector و اعمال K-means

برای حل این مسئله تمام قطعه کد های مورد نیاز بصورت دستی پیاده سازی شده و مورد گزارش قرار گرفته است. لازم به ذکر است که در زمان پیاده سازی، خروجی عددی توابع پیاده سازی شده و خروجی توابع آماده networkx مقایسه شده و اعداد حاصل تا پنج رقم اعشار باهم برابر بوده است(محاسبه مقادیر و بردار های ویژه دارای الگوریتم ها و خطا های متفاوتی بسته به نوع پیاده سازی است که این تفاوت کاملا قابل قبول است) برای این سوال از مجموعه داده airports استفاده شده است.

مسئله سوم: الف) محاسبه ماتریس Laplacian

برای خواندن گراف از فایل داده شده، تابع read_edges پیاده سازی شده است که فایل edges. را خوانده و یک شی از نوع Graph از کتابخانه network را باز می گرداند. ماتریس لاپلاسین برابر است با تفریق ماتریس مجاورت از ماتریس درجه. برای محاسبه ماتریس مجاورت یک تابع با عنوان create_adjacency_matrix پیاده سازی شده است که ابتدا یک ماتریس تماما صفر ایجاد کرده و سپس بر اساس یال ها و اتصالات موجود اقدام به تکمیل ماتریس مجاورت می کند. برای محاسبه ماتریس درجه یک تابع دیگر با عنوان create_degree_matrix پیاده سازی شده است که ماتریس مجاورت را ورودی گرفته و بر اساس مجموع هر ردیف، درجه نود متناظر را بدست می آورد. در نهایت تابع create_laplacian_matrix با استفاده از دو تابع فوق اقدام به ایجاد ماتریس لاپلاسین می کند.

مسئله سوم: ب) محاسبه دومین مقدار ویژه کوچک و بردار ویژه متناظر

برای محاسبه مقادیر و بردار های ویژه از تابع linalg.eig از کتابخانه numpy استفاده شده است. سپس بردار های ویژه را بر اساس مقدار ویژه های متناظر مرتب کرده و دومین مقدار ویژه کوچک و بردار ویژه متناظر را انتخاب کرده و برای مراحل بعد استفاده می کنیم. مقدار ویژه حاصل برابر با ۳۰۰.۱۳ بوده و بردار ویژه متناظر نیز در پوشه P03 ذخیره شده است.

مسئله سوم: ج) و د) خوشه بندی گراف و ارزیابی

برای محاسبه مقادیر modularity و min-cut بر اساس فرمول های اسلاید عمل شده و برای هر یک، یک تابع از پایه پیاده سازی شده است که بترتیب calculate_modularity و calculate_min_cut نام دارد و در گزارش از این توابع استفاده شده است.

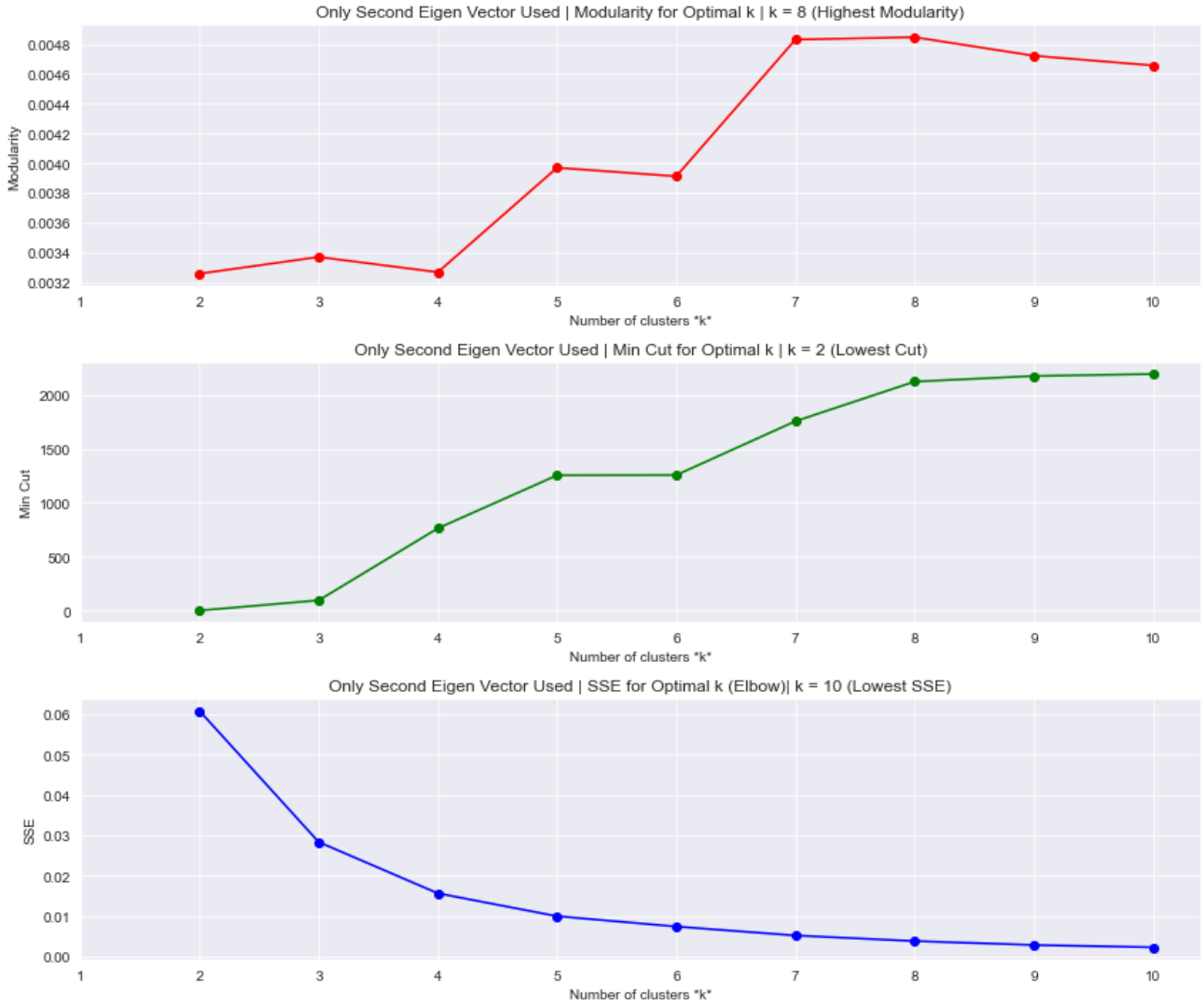
برای این قسمت از الگوریتم خوشه بندی K-Means برای خوشه بندی گراف استفاده شده است. دلیل این انتخاب تعداد کمتر هایپرپارامتر های آن نسبت به DBScan می باشد.(یک در مقابل دو) در این خوشه‌بندی برای هر نود یک عدد متناظر در بردار ویژه در نظر گرفته شده و عبارتی ما برای هر نود صرفا یک عدد داشته و به دنبال خوشه بندی نود ها با این تک اعداد هستیم.

برای انتخاب مناسب هایپرپارامتر K از سه معیار فاصله از مراکز خوشه(Elbow)، min-cut و modularity استفاده شده است. به ازای K های ۲ الی ۱۰ خوشه بندی برای روش K-Means انجام شده و معیار های فوق مورد گزارش قرار گرفته و برای هر کدام یک نمودار بر اساس افزایش K رسم شده است که این نمودار ها در صفحه بعدی قابل مشاهده می باشد که حاصل گزارش همه ی معیارها به ازای همه‌ی kها یکجا می باشد. (اساس و پایه‌ی این روش که مبتنی بر دومین مقدار ویژه کوچک است، برای خوشه بندی دودوئی و پیشروی بصورت سلسله مراتبی می باشد و چندان معنی دار نیست که از این روش برای خوشه بندی نود

های گراف در چندین خوشه استفاده نمود اما از جایی که یک هایپر پارامتر بایستی انتخاب و تعیین گردد، روند مطالعه K های مختلف انجام میپذیرد)

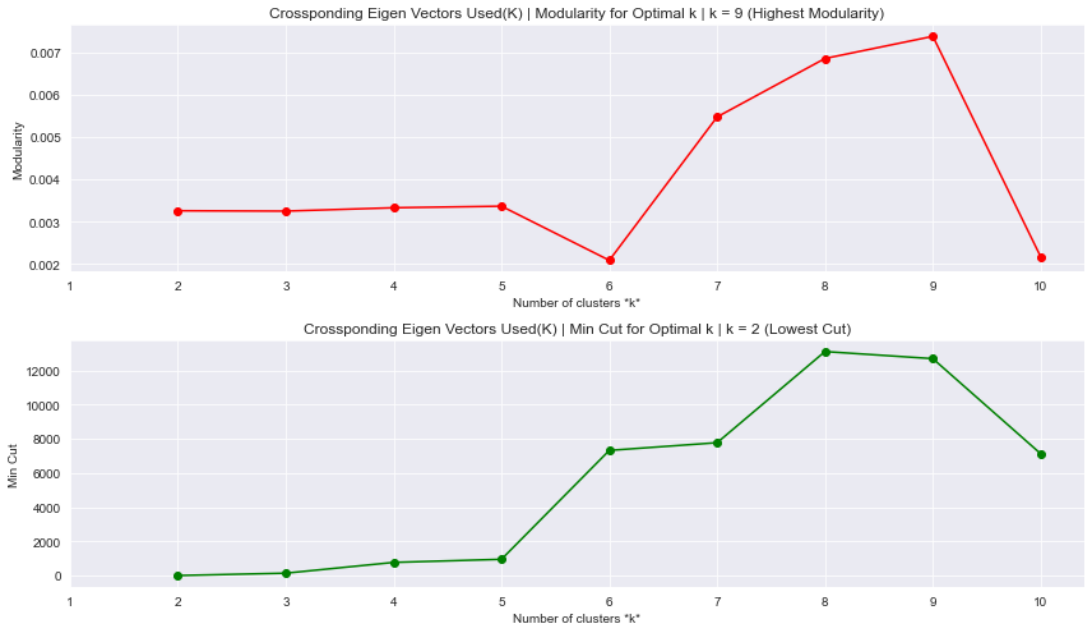
بر اساس معیار Elbow تعداد خوشه ها بهتر است برابر با چهار باشد.(فاصله از مراکز خوشه نسبت به حالت K=3 نصف شده است)

بر اساس معیار ماژولاریتی بهتر است تعداد خوشه ها برابر با ۸ باشد.(بیشترین ماژولاریتی ممکن در این صورت رخ داده است) بر اساس معیار min-cut بهتر است تعداد خوشه ها برابر با دو باشد که کمترین معیار را دارد.



مسئله سوم: فعالیت بیشتر | استفاده از چندین بردار ویژه

در حین تدریس و در کلاس درس بحث شد که استفاده از صرفا یک عدد برای یک نود که متناظر در بردار ویژه کوچک دوم می باشد چندان موثر نبوده و بزور میتواند خوشه بندی مفیدی را انجام دهد که در صورت مناسب بودن خوشه بندی نیز، صرفا به خوشه بندی دودویی خوب می تواند برسد. یک ایده ای که در کلاس درس مطرح شده و مورد تایید استاد قرار گرفت این بوده که به تعداد خوشه ها تعداد بردار ویژه از دومین به بعد را انتخاب کنیم تا بدین صورت برای هر نود به تعداد خوشه های منهای یک تا ویژگی داشته باشیم. برای مثال اگر میخواهیم خوشه بندی ۳ خوشه ای انجام دهیم از بردار ویژه کوچک دوم و سوم استفاده کنیم که در این صورت برای هر نود دو ویژگی خواهیم داشت. بنده این روش را پیاده سازی کرده ام که نتایج آن برای معیار های modularity و min-cut بصورت زیر حاصل می شود. همانطور که قابل مشاهده است بهترین k بر اساس معیار modularity برابر با ۹ خوشه است. تفاوت مهمی که اینجا رخ داده است این است که در خوشه بندی با این روش، ماژولاریتی ۵۰٪ افزایش پیدا کرده و کیفیت خوشه بندی بهتر شده است.



مسئله چهارم

مسئله چهارم: محاسبه سلسله مراتبی خوشه‌ها با Modularity optimization

با توجه به اینکه در متن سوال خواسته نشده است کدهای مورد نیاز دستی پیاده سازی شود/نشود و از طرفی معیار ماژولاریتی در سوال قبلی پیاده سازی شده و مورد ارزیابی قرار گرفته است، در این سوال صرفا برای محاسبه ماژولاریتی و محاسبه ماتریس آن از توابع آماده استفاده شده است تا ضمن جلوگیری از کپی/پست کردن کد از سوال قبل و آشنایی با توابع آماده، زمان پردازشی و محاسباتی نیز کاهش یابد.

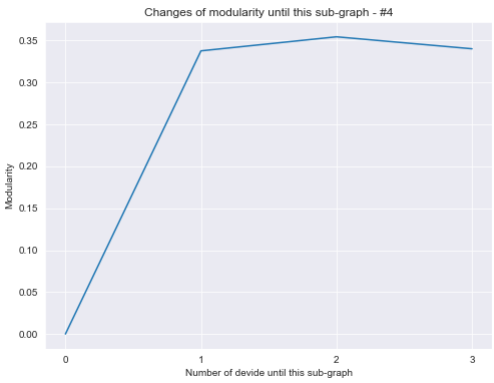
نیاز به ذکر است که استخراج جوامع و تمام مراحل سلسه مراتبی برای کشف خوشه ها بصورت دستی پیاده سازی شده است.

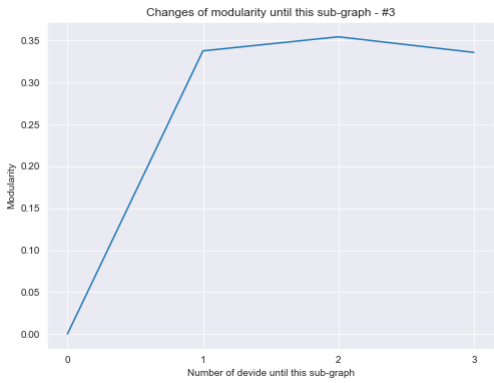
برای پیاده سازی محاسبه سلسله مراتبی خوشه‌ها با Modularity optimization، از تکنیک پیاده سازی جست و جوی اول عمق در درخت با استفاده از یک Stack بهره گرفته شده است. برای این منظور ابتدا در پشته، کل گراف و نود های آن را یک خوشه در نظر گرفته و در پشته قرار می دهیم. سپس در هر مرحله پشته را pop کرده و آخرین خوشه را بدست می آوریم. سپس مقدار ماژولایتی کل گراف را به ازای ترکیب موجود خوشه ها اندازه گیری می کنیم؛ سپس خوشه در دست را به دو خوشه و با روش تدریس شده در کلاس تقسیم می کنیم.(محاسبه ماتریس B و بردار ویژه متناظر با بزرگترین مقدار ویژه آن و خوشه بندی بر اساس علامت متناظر هر نود در بردار ویژه)

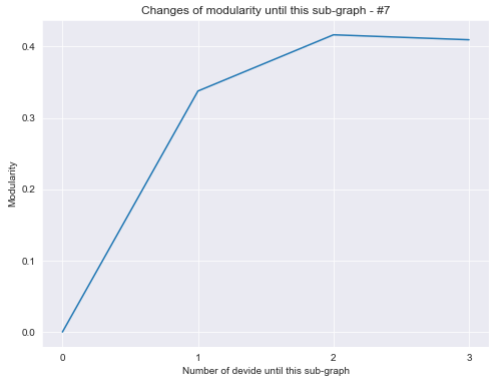
حال ماژولاریتی را برای مجددا برای ترکیب جدید خوشه ها اندازه گیری می کنیم؛ اگر ماژولاریتی افزایش داشته باشد، یعنی این تقسیم مناسب بوده و بهتر از انجام شود. پس، این تقسیم را انجام داده و خوشه های جدید را به پشته اضافه میکنیم تا در ادامه

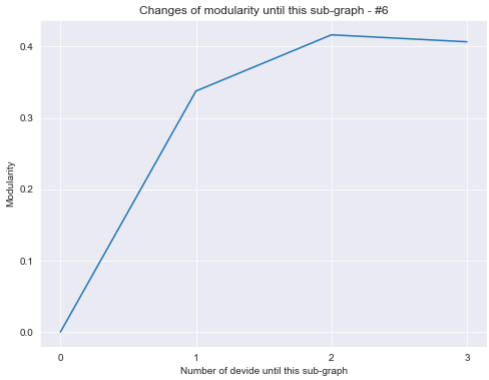
مورد ارزیابی قرار گیرد که همچنان امکان تقسیم به خوشه وجود دارد یا نه. اگر ماژولاریتی افزایش نداشته باشد، یعنی این خوشه‌ی در دست بهتر است بدو خوشه تقسیم نشود؛ لذا در این صورت موردی به پشته افزوده نمی شود و خوشه حاصل به عنوان خوشه نهایی در نظر گرفته می شود. این محاسبات بصورت درخت انجام می شود تا بهترین خوشه ها در جهتی که ماژولاریتی بیشینه شود حاصل شود. خروجی های موجود در فایل P04-outputs.txt در پوشه P04 مربوط به محاسبه سلسله مراتبی خوشه ها با روش بیان شده می باشد که در انتهای گزارش نیز آورده شده است. تعداد خوشه های نهایی برابر با ۴ بوده و تعداد اعضای هر یک بترتیب ۱۶۱۸، ۱۶۹۴، ۱۷۲۹ و ۲۰۷۴ می باشد. همچنین در پوشه مربوطه اعضای هر خوشه در قالب آرایه های numpy گزارش شده است(برای دسترسی سریع تر یک فایل متنی با عنوان Memberships of each node in graph نیز ذخیره شده است).

در طی مراحل پیمایش سلسله مراتبی درخت برای استخراج خوشه ها، میتوان در مواجه شدن با هر خوشه که امکان تقسیم بدو خوشه را نمیدهد، یک نمودار تغییرات ماژولاریتی از راس درخت را رسم نمود تا توسط آن بتوان سیر عدم ایجاد خوشه های جدید را نمایش داد. برای هر کدام چهار خوشه حاصل(که نتوانستند به خوشه های جدید تقسیم شوند) این نمودار در زیر رسم شده است:









**********> Tree of Processing <**********

--------> Community # 1

--------> Length of current sub-graph : 7115

--------> Length of positive cluster : 3347

--------> Length of negative cluster : 3768

--------> Current modularity: 0 New modularity if sub-graph devide: 0.3378

--------> Result: New modularity is better than current modularity. So, we will devide this sub-graph.

--------> Length of each Community after devide : [3768, 3347]

--------> New IDs of each Community after devide : [2, 3]

--------> --------> Community # 2

--------> --------> Length of current sub-graph : 3347

--------> --------> Length of positive cluster : 1729

--------> --------> Length of negative cluster : 1618

--------> --------> Current modularity: 0.3378 New modularity if sub-graph devide: 0.3546

--------> --------> Result: New modularity is better than current modularity. So, we will devide this sub-graph.

--------> --------> Length of each Community after devide : [3768, 1618, 1729]

--------> --------> New IDs of each Community after devide : [3, 4]

--------> --------> --------> Community # 3

--------> --------> --------> Length of current sub-graph : 1729

--------> --------> --------> Length of positive cluster : 828

--------> --------> --------> Length of negative cluster : 901

--------> --------> --------> Current modularity: 0.3546 New modularity if sub-graph devide: 0.3359

--------> --------> --------> Result: Current modularity is better than new modularity if sub-graph devide. So, we will not devide this sub-graph.

--------> --------> --------> Community # 4

--------> --------> --------> Length of current sub-graph : 1618

--------> --------> --------> Length of positive cluster : 853

--------> --------> --------> Length of negative cluster : 765

--------> --------> --------> Current modularity: 0.3546 New modularity if sub-graph devide: 0.3403

--------> --------> --------> Result: Current modularity is better than new modularity if sub-graph devide. So, we will not devide this sub-graph.

--------> --------> Community # 5

--------> --------> Length of current sub-graph : 3768

--------> --------> Length of positive cluster : 1694

--------> --------> Length of negative cluster : 2074

--------> --------> Current modularity: 0.3546 New modularity if sub-graph devide: 0.4166

--------> --------> Result: New modularity is better than current modularity. So, we will devide this sub-graph.

--------> --------> Length of each Community after devide : [2074, 1694]

--------> --------> New IDs of each Community after devide : [6, 7]

--------> --------> --------> Community # 6

--------> --------> --------> Length of current sub-graph : 1694

--------> --------> --------> Length of positive cluster : 951

--------> --------> --------> Length of negative cluster : 743

--------> --------> --------> Current modularity: 0.4166 New modularity if sub-graph devide: 0.4068

--------> --------> --------> Result: Current modularity is better than new modularity if sub-graph devide. So, we will not devide this sub-graph.

--------> --------> --------> Community # 7

--------> --------> --------> Length of current sub-graph : 2074

--------> --------> --------> Length of positive cluster : 1082

--------> --------> --------> Length of negative cluster : 992

--------> --------> --------> Current modularity: 0.4166 New modularity if sub-graph devide: 0.4096

--------> --------> --------> Result: Current modularity is better than new modularity if sub-graph devide. So, we will not devide this sub-graph.

Members of each Community:

Community # 1 : 1729

Community # 2 : 1618

Community # 3 : 1694

Community # 4 : 2074