

UPRAVLJANJE ZNANJEM

Završni ispit – praktični dio

Kolegij: Upravljanje znanjem
Nositelj kolegija: *izv. prof. dr. sc. Ana Meštrović*
Asistent: *dr. sc. Slobodan Beliga*

Izradio: Vedran Orešković
Datum izrade: 12.2.2023.

Popis cjelina

<i>Opis data set-a</i>	1
<i>Postavljena pitanja.....</i>	2
<i>Rješenje.....</i>	3
Zadatak (1).....	3
Dohvaćanje potrebnih vrijednosti.....	3
Definiranje grafa G	5
Crtanje grafa.....	7
Pod grafovi	9
Zadatak (2).....	28
Izlazni stupanj, ulazni stupanj i distribucija stupnjeva.....	28
Jako povezane komponente, dijametar, radijus	30
Zadatak (3).....	31
Centralnost međupoloženosti	31

Opis data set-a

U ovom radu analizirati će problem kompleksnih mreža u domeni transporta. Data set se sastoji od dvije radne stranice **nodelist** i **edgelist**.

Nodelist (sadrži 125 luka) sadrži stupce:

- **ID** – identifikacija grada
- **LABEL** – naziv grada
- **LAT** – zemljopisna širina (u stupnjevima)
- **LNG** – zemljopisna dužina (u stupnjevima)

Edgelist predstavlja broj odrađenih vožnji između dva odredišta te se sastoji od sljedećih stupaca:

- **source** – početni grad
- **target** – krajnji grad
- **weight** – broj vožnji

U ovom radu konstruirati će težinski usmjereni graf. Dostupne informacije poput *usmjerenosti bridova* korištenjem *polazišnog vrha* iz varijable **source** te *odredišnog vrha* iz varijable **target** i *težine bridova* iz varijable **weight** potaknuli su me na tu odluku.

Postavljena pitanja

1) (SREDIŠNJA RAZINA)

Trguju li gradovi sa susjednim gradovima,
dali su te trgovačke rute formirale zajednice?

2) (GLOBALNA RAZINA)

Dali je promet prema gradovima veći od prometa iz gradova?
Kakva je distribucija prometa po zajednicama?

3) (LOKALNA RAZINA)

Gradove povezuju rute koje prenose teret te je logično da će neki gradovi imati mali broj ruta. Postoji li opasnost od ne mogućnosti transporta ukoliko se prekine ruta? Koji gradovi su u opasnosti od ovog problema tj. koji gradovi bi bili odcjepljeni od ostalih gradova?

Rješenje

Zadatak (1)

Dohvaćanje potrebnih vrijednosti

Uvoz potrebnih biblioteka.

```
In [13]: 1 import pandas as pd
2 import pandas as pd
3 import numpy as np
4 from random import sample
5 import pandas as pd
6 import networkx.algorithms.community as nxcom
7 import networkx as nx
8 import matplotlib.pyplot as plt
9 import matplotlib.patches as mpatches
```

Excel izvor podataka postavio sam na svoj [GitHub](#) (radi lakšeg izvođenja te testiranja koda) .

```
In [14]: 1 url = "https://github.com/vedran-o/NetworkX-analyses/blob/main/dataTransportation.xlsx?raw=true"
```

Pristupom na link te pohranom datoteke [dataTransportation.xlsx](#) možemo vidjeti stvarni oblik podataka u programu **Excel**.

source	target	weight
2	P17	P26
3	P26	P17
4	P26	P28
5	P28	P26
6	P27	P47
7	P47	P27
8	P16	P58
9	P58	P16
10	P108	P124
11	P9	P16
12	P16	P9
13	P52	P72
14	P72	P52
15	P95	P113
16	P124	P108
17	P45	P109
18	P113	P95
19	P109	P45
20	P40	P34
21	P34	P40
22	P113	P108
23	P108	P113
24	P77	P117
25	P117	P77
26	P97	P44
27	P51	P97
28	P25	P77
29	P67	P113
30	P72	P107
31	P77	P25
32	P107	P72

Zanimljiv je oblik bilježnice (dvije radne stranice u donjem lijevom kutu). Njima možemo pristupiti pomoću slijedećih naredba te podijeliti data set u popis bridova i vrhova.

```
In [15]: 1 edges_sheet = pd.read_excel(url, sheet_name='edges')
2 nodes_sheet = pd.read_excel(url, sheet_name='Nodes')
```

Možemo i ispisati pohranjene vrijednosti **vrhova** i **bridova**.

```
In [16]: 1 print("Edges sheet:")
2 print(edges_sheet)
3
4 print("Nodes sheet:")
5 print(nodes_sheet)

Edges sheet:
      source target  weight
0       P17    P26     444
1       P26    P17     402
2       P26    P28     152
3       P28    P26     152
4       P27    P47     130
...
316     P36    P57      1
317     P43    P36      1
318     P57   P106      1
319   P101    P45      1
320   P112    P43      1

[321 rows x 3 columns]

Nodes sheet:
      ID LABEL      LAT      LNG
0     P1    P1  56.162930  10.203920
1     P2    P2  36.838141 -2.459736
2     P3    P3  52.373084  4.899902
3     P4    P4  43.598164 13.510075
4     P5    P5  51.216667  4.416667
...
120   P121   P121  57.105568 12.250775
121   P122   P122  45.440840 12.315510
122   P123   P123  57.389444 21.560556
123   P124   P124  55.429659 13.820415
124   P125   P125  51.318940  3.206850

[125 rows x 4 columns]
```

Definiranje grafa G

Nakon uspješno dohvaćenih vrhova i bridova možemo kreirati usmjereni težinski graf. Graf ćemo postaviti na usmjereni radi dostupnih podataka o bridovima tj. stupci **source** i **target**. Stupac **source** početak je brida dok u stupcu **target** nalazi se kraj brida (u točki), samim tim možemo odrediti smjer brida. Graf će biti težinski radi dostupnog stupca **weight** koji definira popularnost puta tj. broj korištenja puta. Sada možemo dodati vrhove te odgovarajuće bridove u graf **G**.

```
In [17]: 1 G = nx.DiGraph()
2
3 # Adding nodes to the graph
4 for index, row in nodes_sheet.iterrows():
5     G.add_node(row['ID'], pos=(row['LAT'], row['LNG']))
6
7 # Adding edges to the graph
8 for index, row in edges_sheet.iterrows():
9     G.add_edge(row['source'], row['target'], weight=row['weight'])
```

Korištenjem **pohlepnog algoritma** kreirati ćemo **zajednice** u grafu.

```
In [18]: 1 # Find communities in the graph
2 communities = nxcom.greedy_modularity_communities(G)
3 communities = sorted(communities, key=len, reverse=True)
4 print("Ova mreža sadrži {} unikatnih zajednica.".format(len(communities)))
```

Izvođenjem koda vidljivo je da **pohlepni algoritam** kreira **14 zajednica**. Želim naglasiti da će zajednice biti predstavljene vrijednostima od 0 do 13 uvećanjem od 1.

Slijedeća funkcija dodjeljuje zajednicu svakom **vrhu**.

```
6 # Function to set the community of each node
7 def set_node_community(graph, community_list):
8     for i, c in enumerate(community_list):
9         for node in c:
10             graph.nodes[node]['community'] = i + 1
11
```

Slijedeća funkcija dodjeljuje zajednicu svakom **bridu**.

```
12 # Function to set the community of each edge
13 def set_edge_community(graph):
14     for v, w in graph.edges:
15         if graph.nodes[v]['community'] == graph.nodes[w]['community']:
16             graph.edges[v, w]['community'] = graph.nodes[v]['community']
17         else:
18             graph.edges[v, w]['community'] = 0
19
```

Slijedeća funkcija dodjeljuje **RGB** vrijednost za **svaku zajednicu** (generira se vrijednost pomoću postavljene formule prvo **RED** vrijednost piksela potom **GREEN** vrijednost piksela te napislijetku **BLUE** vrijednost piksela) ova funkcija je odlična jer nam omogućuje lako skaliranje boja zajednica tj. dodavanjem novih zajednica automatski se generira unikatna boja zajednice.

```

20 # Function to generate a color for each node
21 def get_node_color(i, r_offset=1, g_offset=1, b_offset=1):
22     r0, g0, b0 = 0, 0, 0
23     n = 16
24     low, high = 0.1, 0.9
25     span = high - low
26     r = low + span * (((i + r_offset) * 1) % n) / (n - 1)
27     g = low + span * (((i + g_offset) * 2) % n) / (n - 1)
28     b = low + span * (((i + b_offset) * 3) % n) / (n - 1)
29     return (r, g, b)

```

Prikaz izlaza funkcije `get_node_color`:

```
Out[132]: (0.20666666666666667, 0.3133333333333335, 0.42000000000000004)
```

Prethodne tri funkcije dostupne su javno na projektu [analiza mreže karate kluba](#).

Sada možemo pozvati prethodno definirane funkcije.

```

In [19]: 1 # Set the community of each node and edge
2 set_node_community(G, communities)
3 set_edge_community(G)
4
5 # Generate node colors based on community
6 node_colors = [get_node_color(G.nodes[v]['community']) for v in G.nodes]
7

```

Dodatno sam definirao bridove koji se nalaze unutar zajednice **internal_edges** te bridove koji povezuju zajednice **external_edges**. Boja bridova koji se nalaze u zajednici postavljena je na **crnu** dok je boja bridova koji povezuju zajednice **siva**.

```

8 # Separate internal and external edges
9 external_edges = [(v, w) for v, w in G.edges if G.edges[v, w]['community'] == 0]
10 internal_edges = [(v, w) for v, w in G.edges if G.edges[v, w]['community'] > 0]
11 edge_colors = ['black' for e in internal_edges]
12

```

Svakom vrhu sada možemo dati poziciju **LAN** i **LNG**.

```

13 # Get the position of each node
14 node_pos = nx.get_node_attributes(G, 'pos')

```

Crtanje grafa

Sada možemo nacrtati mrežu.

```
In [20]: 1 # Draw the graph
2 plt.figure(figsize=(100, 100))
3
4 nx.draw_networkx(G, pos=node_pos, node_size=5000, node_color=node_colors, edgelist=internal_edges, edge_color=edge_colors)
5
6 nizic = []
7
8 for i in range(0, len(community)):
9     nizic.append(mpatches.Patch(color=node_colors[i], label=community[i]))
10
11 print(nizic)
12
13 plt.title("Korištenje 'Pohlepne' grupacija")
14 plt.legend(handles=nizic, loc="upper left")
15 plt.show()
```

Mreži ćemo dati veće dimenzije od 100x100. Za crtanje grafa potrebno je dostaviti slijedeće parametre u naredbu `nx.draw_networkx`:

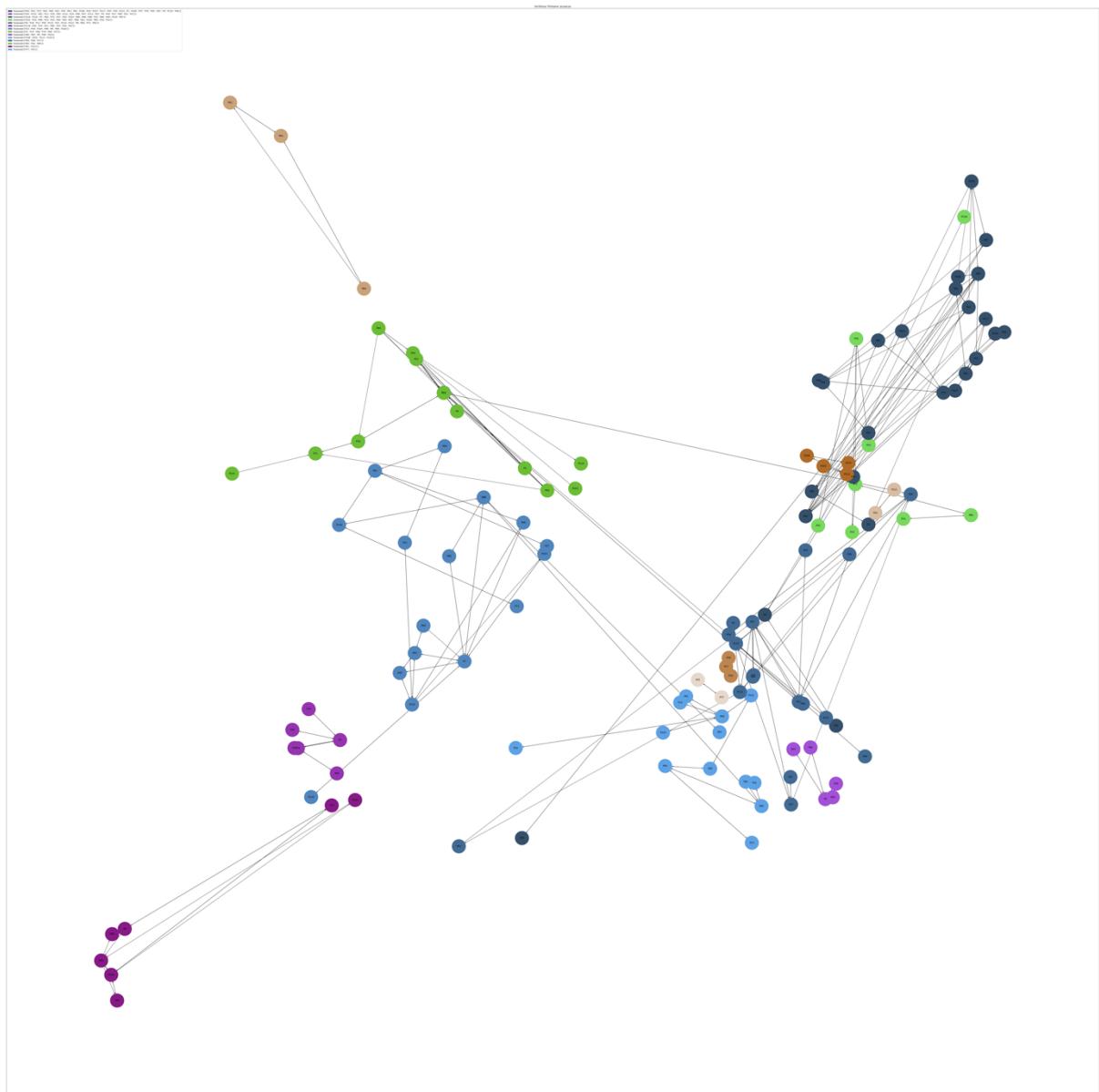
1. **G** – graf
2. **pos** – pozicija točaka (iz varijable `node_pos` koja čita graf G u kojem su pohranjene koordinate)
3. **node_size** – veličina točaka pri crtaju
4. **node_color** – boja točaka (grupirali smo ih preko funkcije te dinamički dodijelili boju)
5. **edgelist** – popis bridova za crtaju (ovdje sam postavio samo bridove koji se nalaze u zajednicama)
6. **edge_color** – boja bridova (dobivamo iz varijable `edge_colors`)

Rječnik `nizic` definiran je u svrhu identifikacije vrhova po zajednici te za crtaju legende u grafu.

```
[■ frozenset({P54, P43, P72, P62, P95, P52, P76, P91, P81, P106, P29, P107, P117, P35, P78, P123, P1, P109, P57, P70, P45, P67, P3, P119, P36})
■ frozenset({P30, P125, P62, P11, P39, P40, P12, P24, P48, P47, P111, P97, P5, P44, P27, P96, P51, P31})
■ frozenset({P16, P10, P7, P82, P73, P37, P20, P103, P88, P68, P98, P15, P66, P49, P120, P83})
■ frozenset({P101, P19, P89, P23, P32, P94, P93, P87, P86, P61, P105, P85, P10, P14})
■ frozenset({P4, P18, P12, P50, P115, P22, P114, P122, P8, P84, P71, P92})
■ frozenset({P18, P56, P23, P21, P80, P53, P34, P55})
■ frozenset({P13, P59, P104, P99, P60, P90, P100})
■ frozenset({P2, P74, P38, P79, P69, P75})
■ frozenset({P46, P65, P9, P58, P16})
■ frozenset({P108, P102, P113, P124})
■ frozenset({P26, P28, P17})
■ frozenset({P64, P42, P60})
■ frozenset({P41, P121})
■ frozenset({P77, P25})]
```

U legendi grafa direktno sam upario dinamično generirane boje te generirane zajednice. Micanje `frozensest-a` moguće je no ovakav prikaz sortiran je po silaznom principu (najveća zajednica prema najmanjoj zajednici) indeksi zajednica od najveće (zajednica[0]) do najmanje (zajednica[13]).

Prikaz grafa sa legendom i naslovom:



Radi velikih dimenzija koje su stvarno potrebne kako bi se graf pravilno vidi legenda je izuzetno mala. Dimenzijs nam omogućuju zumiranje do 100X samim tim legenda će se vidjeti perfektno ukoliko se plot preuzme lokalno.

Pod grafovi

Pohlepni algoritam kreira 14 unikatnih zajednica (indeksi od 0 do 13). Moj cilj bio je svaki od tih grafova izvući te nacrtati na zasebnom ekranu. Za ovaj zadatak vratio sam se na početak te definirao stupac u data set-u **vrhova** i **bridova** pod nazivom **group**. Moja zamisao je definirati dinamički grupe tj. zajednice pridruživanjem vrijednosti iz varijable **communities** i **ID-a** svakog vrha:

```
In [21]: 1 nodes_sheet['group'] = -1
2 edges_sheet['group'] = -1
3
4 for i, community in enumerate(communities):
5     for node in community:
6         nodes_sheet.loc[nodes_sheet['ID'] == node, 'group'] = i
7
```

Pohranjene vrijednosti mogu se pročitati iz rječnika **node_group_map** oblika **tuple** (pričaz prvih dvanaest točaka):

```
Out[31]: {'P1': 0,
          'P2': 7,
          'P3': 0,
          'P4': 4,
          'P5': 1,
          'P6': 6,
          'P7': 2,
          'P8': 4,
          'P9': 8,
          'P10': 3,
          'P11': 1,
          'P12': 4,
```

Naredba za generiranje **node_group_map** rječnika:

```
8 # Create a dictionary from the nodes_sheet that maps node names to their group
9 node_group_map = dict(zip(nodes_sheet['LABEL'], nodes_sheet['group']))
```

Potrebno je kreirati funkciju koja će pročitati vrijednost brida kojeg želimo usporediti. Za usporedbu koristiti će svoj algoritam koji uspoređuje vrhove u zajednici. Kreirati će dva stupca u data set-u bridova **source_group** i **target_group**. Logički će gledati da li brid dolazi 'izvana' (brid pokušava spojiti dvije zajednice) te će takve bridove maknuti, a bridovi koji se nalaze u promatranoj zajednici imati će podrijetlo tj. **source_group** i **target_group** iz iste zajednice.

```

11 # Define a function that returns the group of a node, given its name
12 def get_group(node_name):
13     return node_group_map[node_name]
14
15 # Add a new column to the edges_sheet that indicates the group of each source node
16 edges_sheet['source_group'] = edges_sheet['source'].apply(get_group)
17
18 # Add a new column to the edges_sheet that indicates the group of each target node
19 edges_sheet['target_group'] = edges_sheet['target'].apply(get_group)
20
21 # Filter the edges_sheet to only include rows where the source and target nodes are in the same group
22 same_group_edges = edges_sheet[edges_sheet['source_group'] == edges_sheet['target_group']]
23
24 # Copy the group of the source node to a new column in the same_group_edges dataframe
25 same_group_edges['group'] = same_group_edges['source_group']
26
27 # Drop the source_group and target_group columns, as they are no longer needed
28 same_group_edges.drop(columns=['source_group', 'target_group'], inplace=True)
29
30 # Save the result to a new CSV file
31 same_group_edges

```

Rezultat sam prvo pohranio u **CSV** format radi lakše analize, a potom provjere i analize aktivno ga koristim za kreiranje pod grafova.

Prikaz **bridova grupiranih u zajednice u CSV formatu:**

	source	target	weight	group
0	P17	P26	444	10
1	P26	P17	402	10
2	P26	P28	152	10
3	P28	P26	152	10
4	P27	P47	130	1
...
311	P120	P66	2	2
312	P123	P67	2	0
316	P36	P57	1	0
317	P43	P36	1	0
318	P57	P106	1	0

278 rows × 4 columns

Sada možemo pozvati i prikazati pod grafove:

```
In [27]: 1 for z in range(0, 14):
2     edge_0 = same_group_edges[same_group_edges['group'] == z]
3     node_0 = nodes_sheet[nodes_sheet['group'] == z]
4
5     F = nx.DiGraph()
6
7     # Adding nodes to the graph
8     for index, row in edge_0.iterrows():
9         F.add_node(row['ID'], pos=(row['LAT'], row['LNG']))
10
11    # Adding edges to the graph
12    for index, row in edge_0.iterrows():
13        F.add_edge(row['source'], row['target'], weight=row['weight'])
14
15    node_pos_F = nx.get_node_attributes(F, 'pos')
16
17    print(F.edges)
18
19    print(edge_0)
20    print(node_0)
21
22
23
24    plt.figure(figsize=(100, 100))
25    nizic_pod = []
26    nizic_pod.append(mpatches.Patch(color=node_colors[z], label=communities[z]))
27    print(nizic)
28
29    plt.title("Korištenje 'Pohlepne' grupacije - pod graf")
30    plt.legend(handles=nizic_pod, loc="upper left")
31    nx.draw_networkx(F, pos=node_pos_F, node_size=5000, edgelist=F.edges(), node_color=node_colors[z])
32    plt.show()
```

Grafove pozivamo putem for funkcije sa parametrom z. Varijable edge_0 i node_0 dinamički mijenjaju svoj stanje po iteraciji. Ranije sam spomenuo indexe od najvećeg pod grafa do najmanjeg i upravo taj pristup koriste naredbe. Brzi pregled koda:

- Linija 2 i 3: Definiraju bridove i vrhove pod grafa ovisno o parametru (na prvom pokretanju najveći pod graf)
- Linija 5: Kreira novi graf koji će koristiti za definiranje pod grafova
- Linija 8 i 9: Čitaju vrhove pod grafa te njihovu lokaciju na ekranu kako bi dosljednost s originalnog grafa bila održana
- Linija 12 i 13: Čitaju bridove pod grafa njihovu početnu i krajnju točku te težinu
- Linija 15: Poziva attribute vrhova
- Linija 18 do 21: Nemaju pretjeranu važnost (postavljene su radi izuzetno velike količine podataka kao logička provjera tj. dali se grupiraju dobri vrhovi i bridovi)
- Linija 24: Omogućuje veliki prikaz grafa
- Linija 25 do 27: Uparuje ime zajednice pod grafa i pripadajućih vrhova **sa istom bojom** koju je imao u prethodnom grafu
- Linija 29 do 32: Crta graf

Ovi koraci ponavljaju se za sve pod grafove!

Prikazati će svih 14 pod grafova sa dodatnom analizom i ispisom njihovih bridova i vrhova:

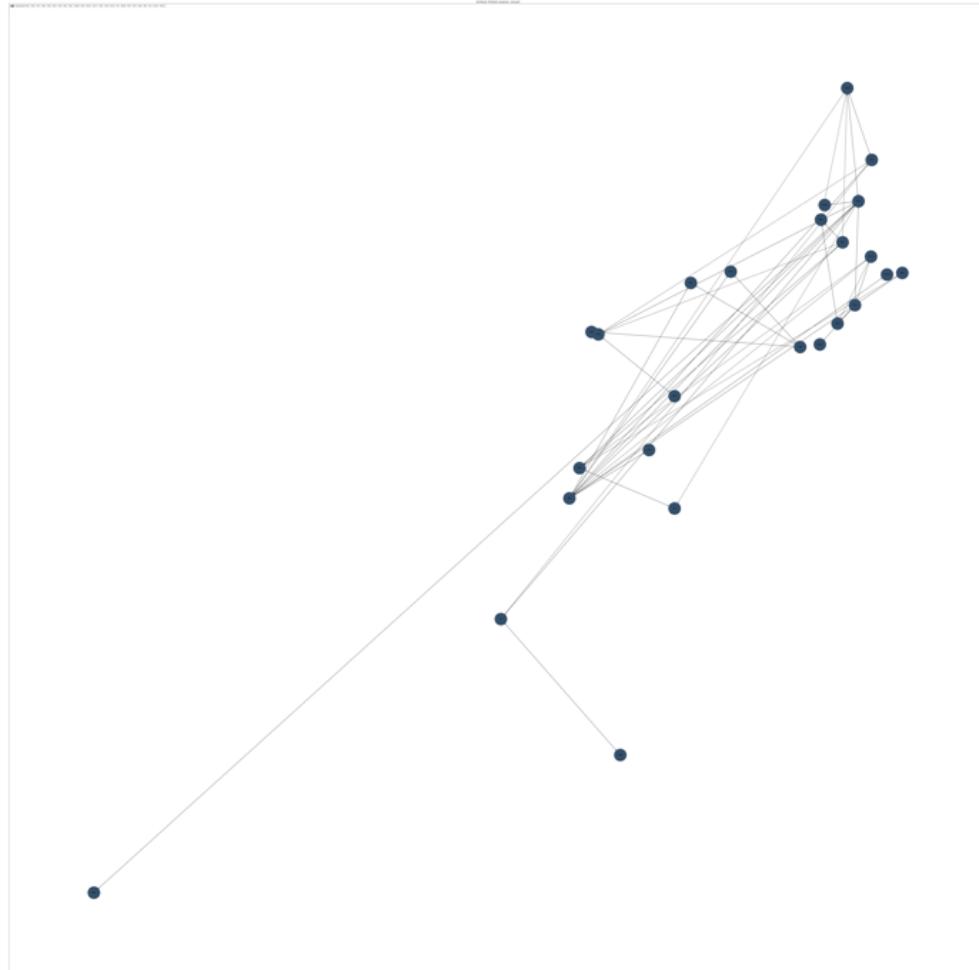
1. Pod graf [0] - najveći

```

source target weight group
11    P52    P72    80    0
12    P72    P52    80    0
15    P45    P109   78    0
17    P109   P45    76    0
22    P72    P117   52    0
...
309   P117   P95    2     0
312   P123   P67    2     0
316   P36    P57    1     0
317   P43    P36    1     0
318   P57    P106   1     0

[67 rows x 4 columns]
   ID LABEL      LAT      LNG group
0  P1  P1  56.162930  10.203920  0
2  P3  P3  52.373084  4.899902  0
28  P29  P29  43.489640 -8.219340  0
34  P35  P35  54.350000 18.666666  0
35  P36  P36  54.500000 18.549999  0
42  P43  P43  59.828340 22.965870  0
44  P45  P45  60.175556 24.934166  0
51  P52  P52  59.718940 19.065780  0
53  P54  P54  56.161564 15.586606  0
56  P57  P57  60.466667 26.916666  0
62  P63  P63  56.516667 21.016666  0
66  P67  P67  53.868927 10.687294  0
69  P70  P70  55.604980 13.003820  0
71  P72  P72  60.100000 19.950001  0
75  P76  P76  54.978250 -1.617780  0
77  P78  P78  58.903366 17.947926  0
88  P81  P81  59.356667 24.053055  0
90  P91  P91  61.133333 21.500000  0
94  P95  P95  54.088694 12.148493  0
105  P106  P106  59.931050 30.360900  0
106  P107  P107  59.332577 18.064903  0
108  P109  P109  59.436958 24.753527  0
116  P117  P117  60.450000 22.283333  0
118  P119  P119  60.800000 21.416666  0
122  P123  P123  57.389444 21.560556  0

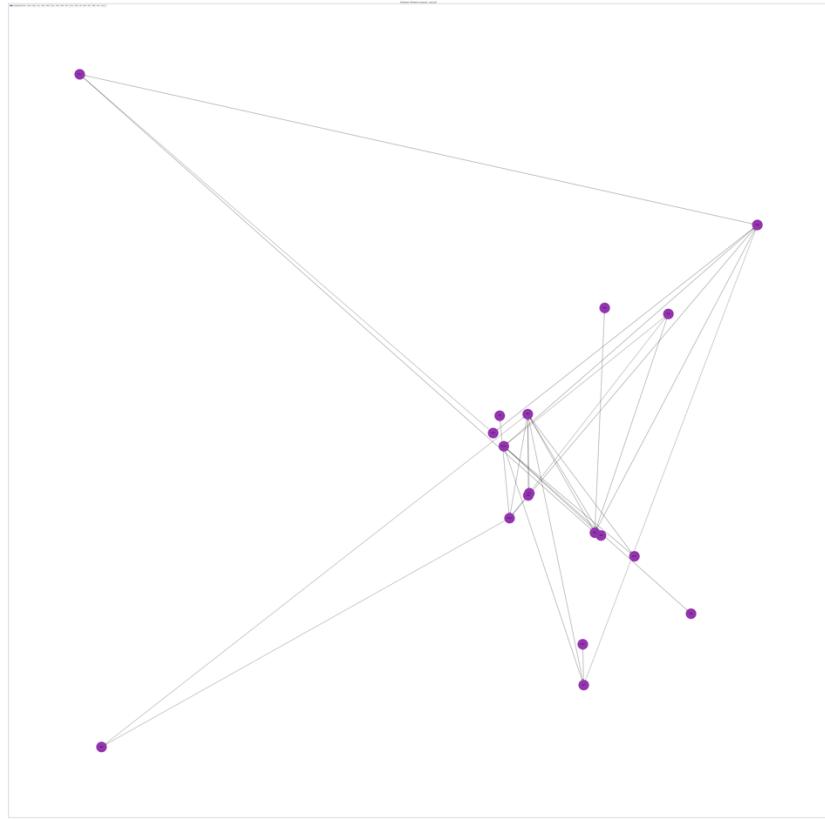
```



2. Pod graf [1]

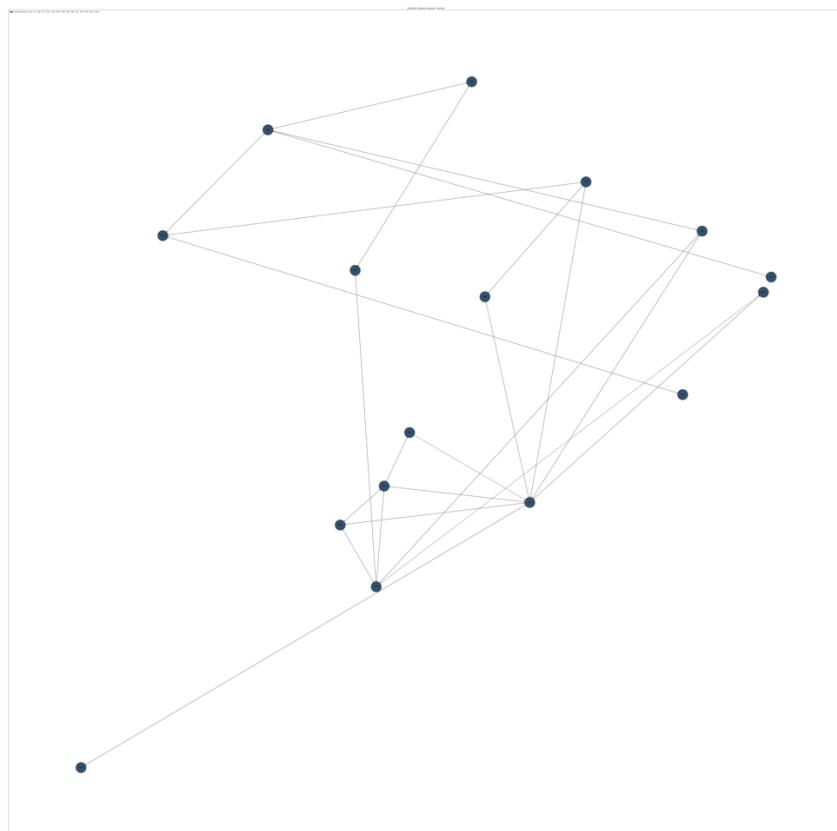
	source	target	weight	group	
4	P27	P47	130	1	
5	P47	P27	130	1	
24	P97	P44	48	1	
25	P51	P97	44	1	
32	P44	P97	40	1	
43	P31	P97	32	1	
46	P97	P31	32	1	
47	P97	P51	30	1	
81	P48	P97	14	1	
82	P48	P125	14	1	
97	P97	P48	14	1	
102	P125	P48	14	1	
104	P30	P51	12	1	
106	P40	P39	12	1	
107	P40	P125	12	1	
110	P51	P30	12	1	
111	P51	P40	12	1	
118	P111	P125	12	1	
119	P112	P97	12	1	
120	P125	P40	11	1	
124	P24	P51	10	1	
125	P39	P40	10	1	
126	P40	P51	10	1	
130	P51	P24	10	1	
132	P97	P112	10	1	
133	P125	P111	10	1	
155	P7	P125	6	1	
164	P96	P125	6	1	
165	P97	P111	6	1	
169	P111	P97	6	1	
172	P125	P27	6	1	
173	P125	P96	6	1	
182	P11	P40	4	1	
187	P27	P97	4	1	
190	P40	P112	4	1	
219	P112	P62	4	1	
229	P5	P112	2	1	
233	P11	P51	2	1	
241	P27	P40	2	1	
243	P30	P125	2	1	
245	P39	P11	2	1	
246	P40	P11	2	1	
260	P51	P11	2	1	
269	P62	P97	2	1	
292	P97	P27	2	1	
302	P112	P5	2	1	
303	P112	P30	2	1	
304	P112	P40	2	1	
	ID	LABEL	LAT	LNG	group
4	P5	P5	51.216667	4.416667	1
10	P11	P11	40.632720	17.941760	1
23	P24	P24	53.859330	8.687900	1
26	P27	P27	53.333056	-6.248889	1
29	P30	P30	55.466667	8.450000	1
30	P31	P31	51.961720	1.351250	1
38	P39	P39	51.050018	3.730330	1
39	P40	P40	57.708870	11.974560	1
43	P44	P44	51.934730	1.266290	1
46	P47	P47	53.309440	-4.633030	1
47	P48	P48	53.767620	-0.327410	1
50	P51	P51	53.614010	-0.215910	1
61	P62	P62	41.182770	-8.703050	1
95	P96	P96	56.036460	-3.423060	1
96	P97	P97	51.922500	4.479167	1
110	P111	P111	54.610118	-1.151240	1
111	P112	P112	51.463020	0.366490	1
124	P125	P125	51.318940	3.206850	1

[<matplotlib.patches.Patch object at 0x2f12b1]



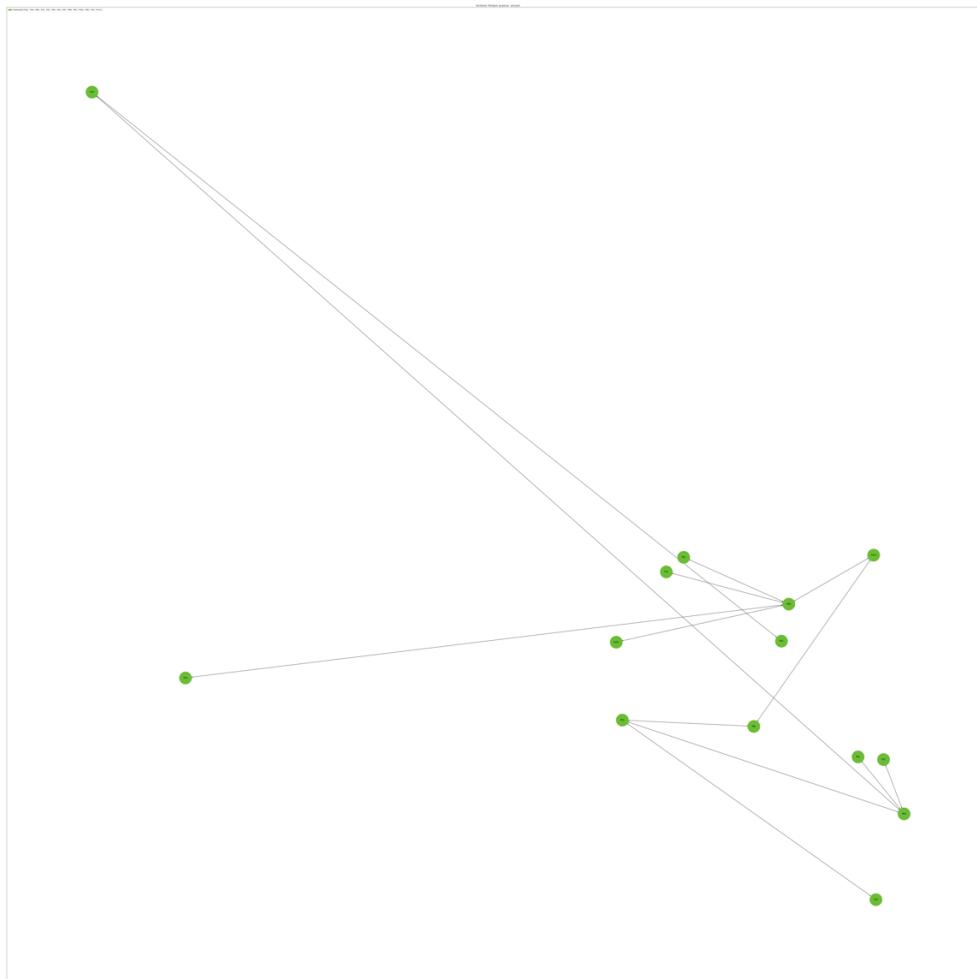
3. Pod graf [2]

	source	target	weight	group	
53	P49	P120	26	2	
54	P120	P49	26	2	
61	P83	P120	18	2	
63	P7	P83	16	2	
68	P68	P7	16	2	
74	P120	P83	16	2	
94	P83	P7	14	2	
121	P7	P49	10	2	
127	P49	P7	10	2	
134	P7	P20	8	2	
135	P7	P66	8	2	
137	P20	P7	8	2	
140	P37	P82	8	2	
142	P66	P7	8	2	
147	P7	P103	6	2	
152	P15	P98	6	2	
153	P15	P120	6	2	
157	P66	P120	6	2	
159	P73	P116	6	2	
161	P82	P66	6	2	
166	P98	P15	6	2	
168	P103	P7	6	2	
170	P116	P73	6	2	
171	P120	P15	6	2	
181	P7	P88	4	2	
184	P20	P88	4	2	
200	P66	P82	4	2	
204	P82	P37	4	2	
205	P82	P98	4	2	
206	P82	P116	4	2	
207	P83	P49	4	2	
208	P88	P7	4	2	
209	P88	P20	4	2	
216	P98	P82	4	2	
220	P116	P82	4	2	
222	P120	P103	4	2	
232	P7	P110	2	2	
237	P20	P116	2	2	
258	P49	P83	2	2	
277	P68	P83	2	2	
280	P83	P68	2	2	
301	P110	P7	2	2	
307	P116	P20	2	2	
311	P120	P66	2	2	
	ID	LABEL	LAT	LNG	group
6	P7	P7	41.388787	2.159895	2
14	P15	P15	39.207384	9.134617	2
19	P20	P20	42.092420	11.795410	2
36	P37	P37	44.406316	8.933859	2
48	P49	P49	39.020000	1.482140	2
65	P66	P66	43.542636	10.316005	2
67	P68	P68	39.887320	4.259610	2
72	P73	P73	43.300000	5.400000	2
81	P82	P82	38.115820	13.359761	2
82	P83	P83	39.569600	2.659160	2
87	P88	P88	40.828370	8.341700	2
97	P98	P98	40.663210	14.803745	2
102	P103	P103	44.309048	8.477154	2
109	P110	P110	35.780000	-5.810000	2
115	P116	P116	36.802778	10.179722	2
119	P120	P120	39.469900	-0.376200	2



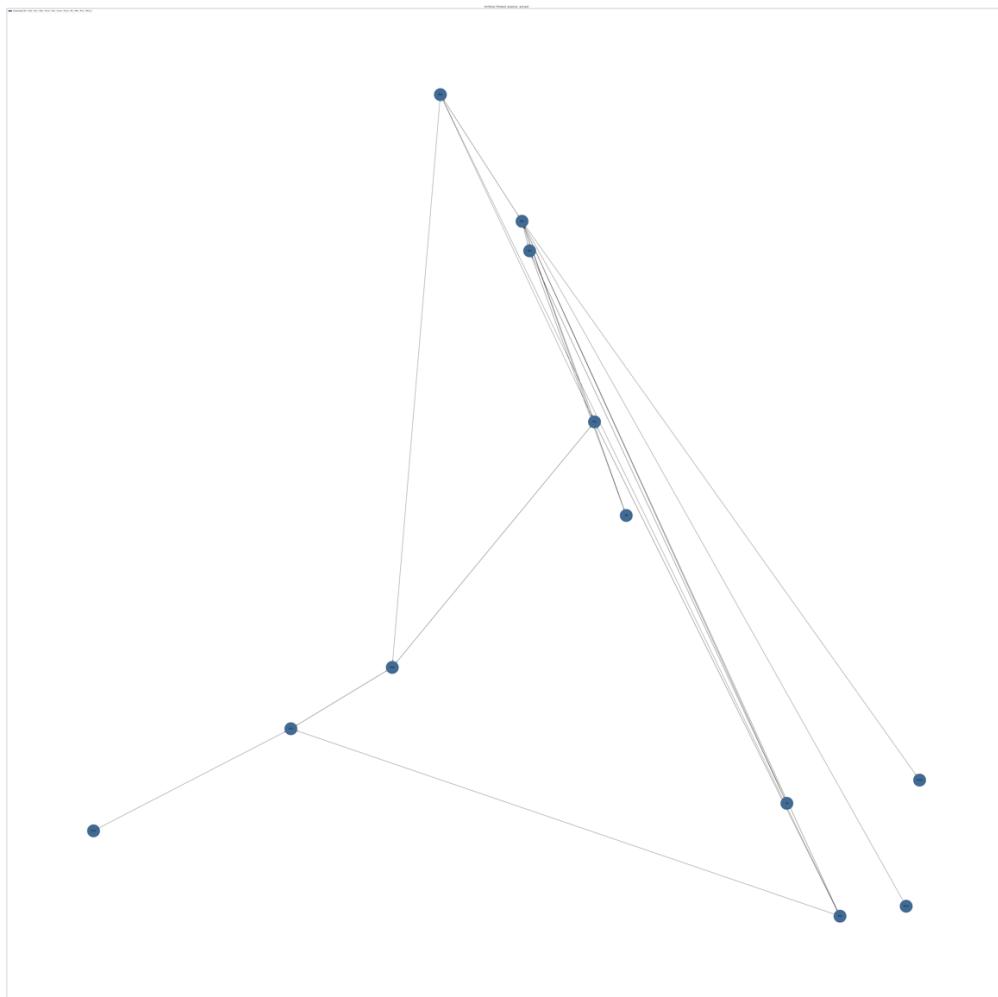
4. Pod graf [3]

source	target	weight	group		
31	P14	40	3		
33	P89	40	3		
48	P32	28	3		
49	P85	28	3		
50	P94	28	3		
51	P94	28	3		
58	P86	20	3		
59	P93	20	3		
65	P19	16	3		
67	P61	16	3		
72	P89	16	3		
73	P94	16	3		
76	P19	14	3		
95	P87	14	3		
96	P89	14	3		
99	P105	14	3		
149	P10	6	3		
162	P89	6	3		
167	P101	6	3		
210	P89	4	3		
212	P93	4	3		
213	P94	4	3		
239	P23	2	3		
284	P86	2	3		
290	P93	2	3		
296	P101	2	3		
	ID	LABEL	LAT	LNG	group
9	P10	P10	43.262706	-2.925282	3
13	P14	P14	49.276650	-0.258650	3
18	P19	P19	42.092420	11.795410	3
22	P23	P23	51.898611	-8.495833	3
31	P32	P32	51.993921	-4.975980	3
60	P61	P61	49.493804	0.107675	3
84	P85	P85	51.674040	-4.908630	3
85	P86	P86	50.371525	-4.143847	3
86	P87	P87	50.716667	-2.000000	3
88	P89	P89	50.809081	-1.071425	3
92	P93	P93	48.726190	-3.985320	3
93	P94	P94	52.251300	-6.341490	3
100	P101	P101	51.870170	0.159610	3
104	P105	P105	48.649330	-2.025670	3



5. Pod graf [4]

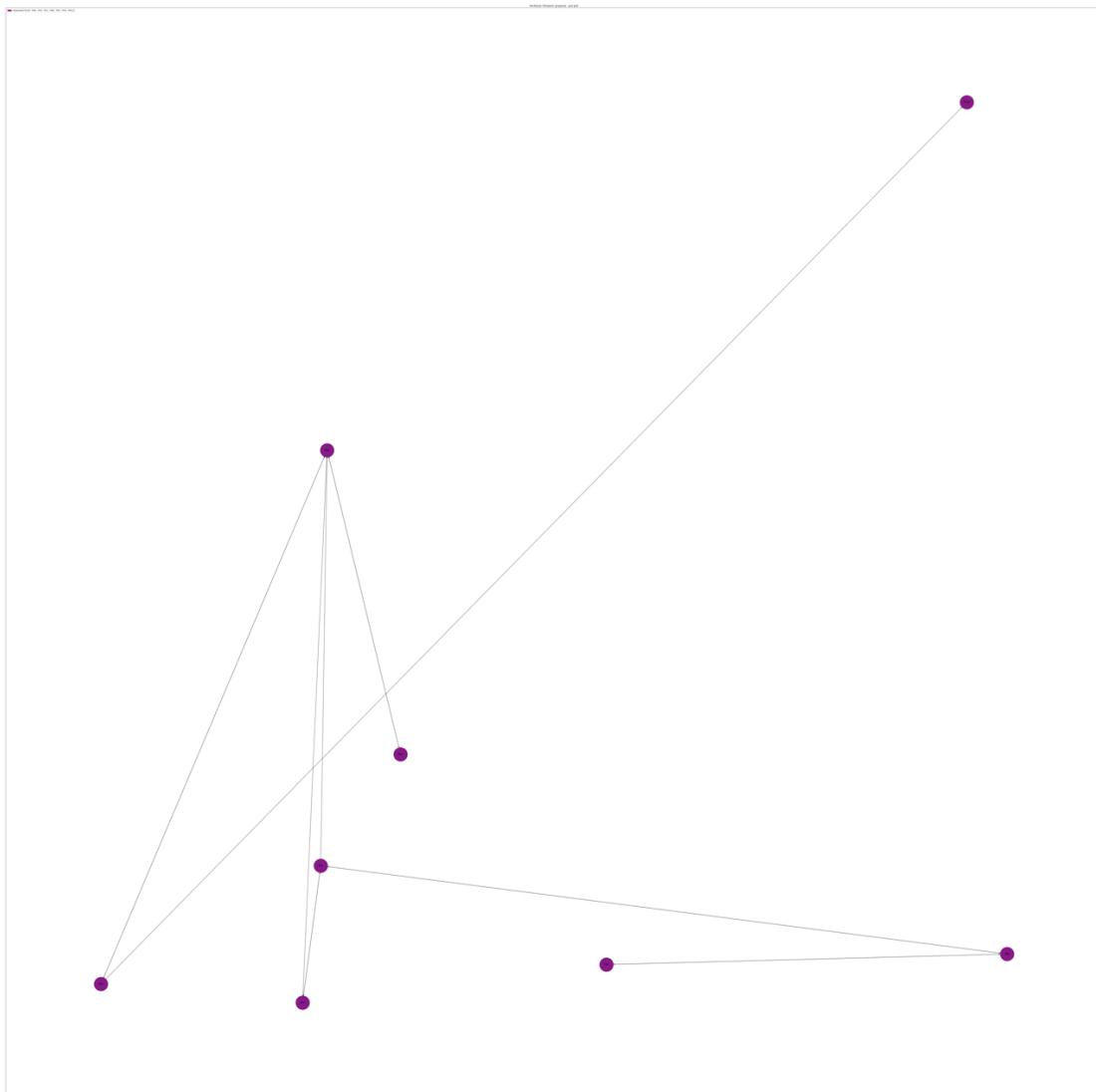
	source	target	weight	group	
37	P50	P84	36	4	
42	P84	P50	34	4	
62	P4	P50	16	4	
64	P12	P50	16	4	
66	P50	P4	16	4	
83	P50	P12	14	4	
116	P84	P12	12	4	
122	P8	P50	10	4	
128	P50	P8	10	4	
129	P50	P22	10	4	
136	P18	P71	8	4	
138	P22	P50	8	4	
141	P50	P92	8	4	
144	P71	P18	8	4	
148	P8	P22	6	4	
150	P12	P18	6	4	
151	P12	P92	6	4	
154	P22	P8	6	4	
163	P92	P12	6	4	
178	P4	P22	4	4	
183	P18	P12	4	4	
186	P22	P4	4	4	
211	P92	P50	4	4	
223	P122	P50	4	4	
259	P50	P114	2	4	
281	P84	P18	2	4	
283	P84	P92	2	4	
289	P92	P71	2	4	
306	P115	P71	2	4	
	ID	LABEL	LAT	LNG	group
3	P4	P4	43.598164	13.510075	4
7	P8	P8	41.117734	16.851185	4
11	P12	P12	40.627731	17.936815	4
17	P18	P18	37.582134	15.087190	4
21	P22	P22	39.624268	19.921670	4
49	P50	P50	39.586148	20.265530	4
70	P71	P71	35.937494	14.375410	4
83	P84	P84	38.246638	21.734570	4
91	P92	P92	44.417500	12.201111	4
113	P114	P114	45.648611	13.780000	4
114	P115	P115	32.887200	13.191330	4
121	P122	P122	45.440840	12.315510	4



6. Pod graf [5]

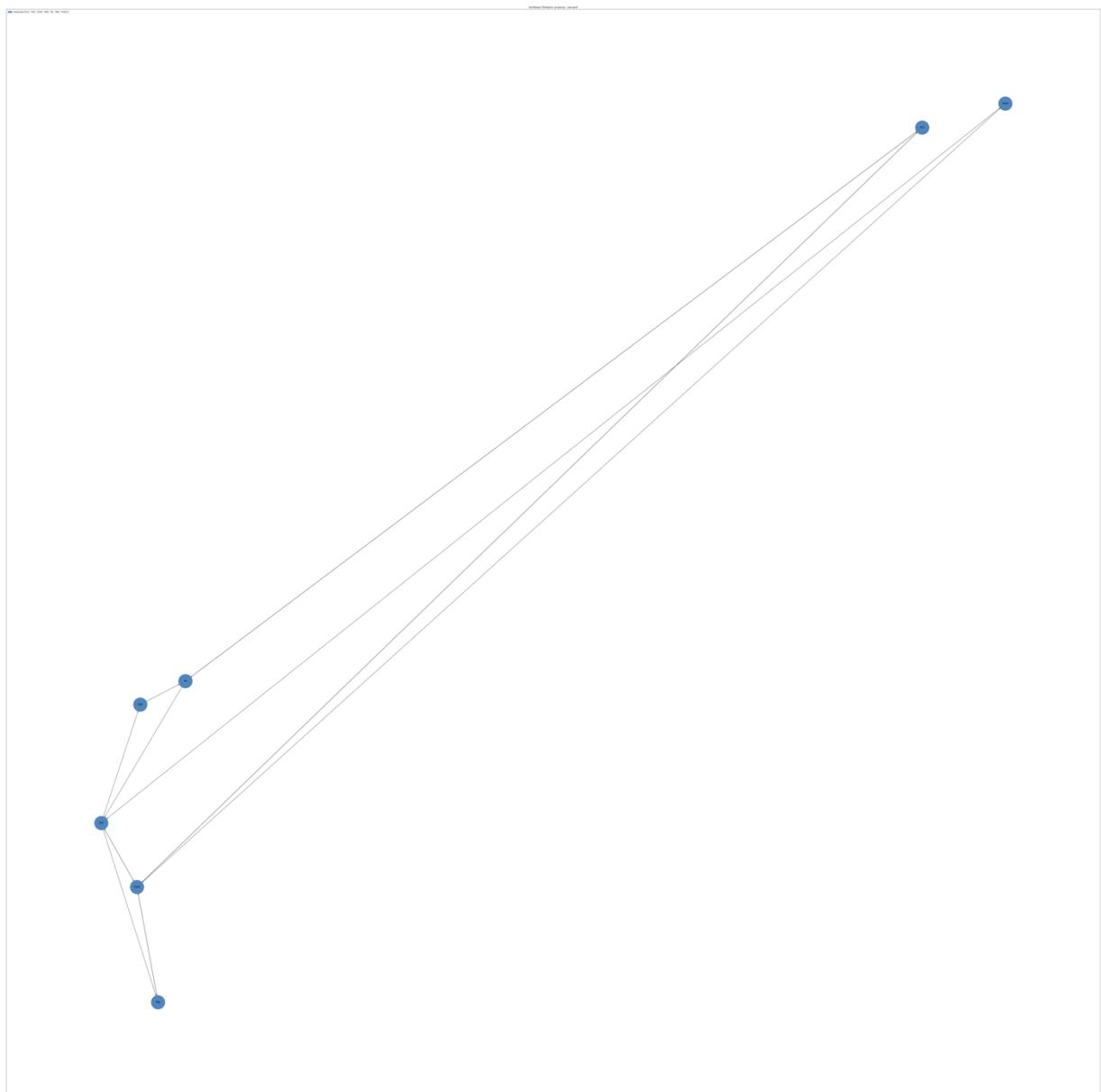
	source	target	weight	group
77	P21	P80	14	5
84	P53	P56	14	5
86	P56	P53	14	5
93	P80	P21	14	5
105	P34	P80	12	5
113	P55	P56	12	5
114	P80	P34	12	5
131	P56	P55	10	5
185	P21	P56	4	5
188	P33	P21	4	5
238	P21	P33	2	5
262	P55	P118	2	5
263	P56	P33	2	5

	ID	LABEL	LAT	LNG	group
20	P21	P21	55.677681	12.570934	5
32	P33	P33	55.565676	9.752565	5
33	P34	P34	57.440735	10.536688	5
52	P53	P53	56.179682	14.861884	5
54	P55	P55	54.321329	10.134888	5
55	P56	P56	55.717222	21.117500	5
79	P80	P80	59.913860	10.752240	5
117	P118	P118	59.664410	28.279210	5



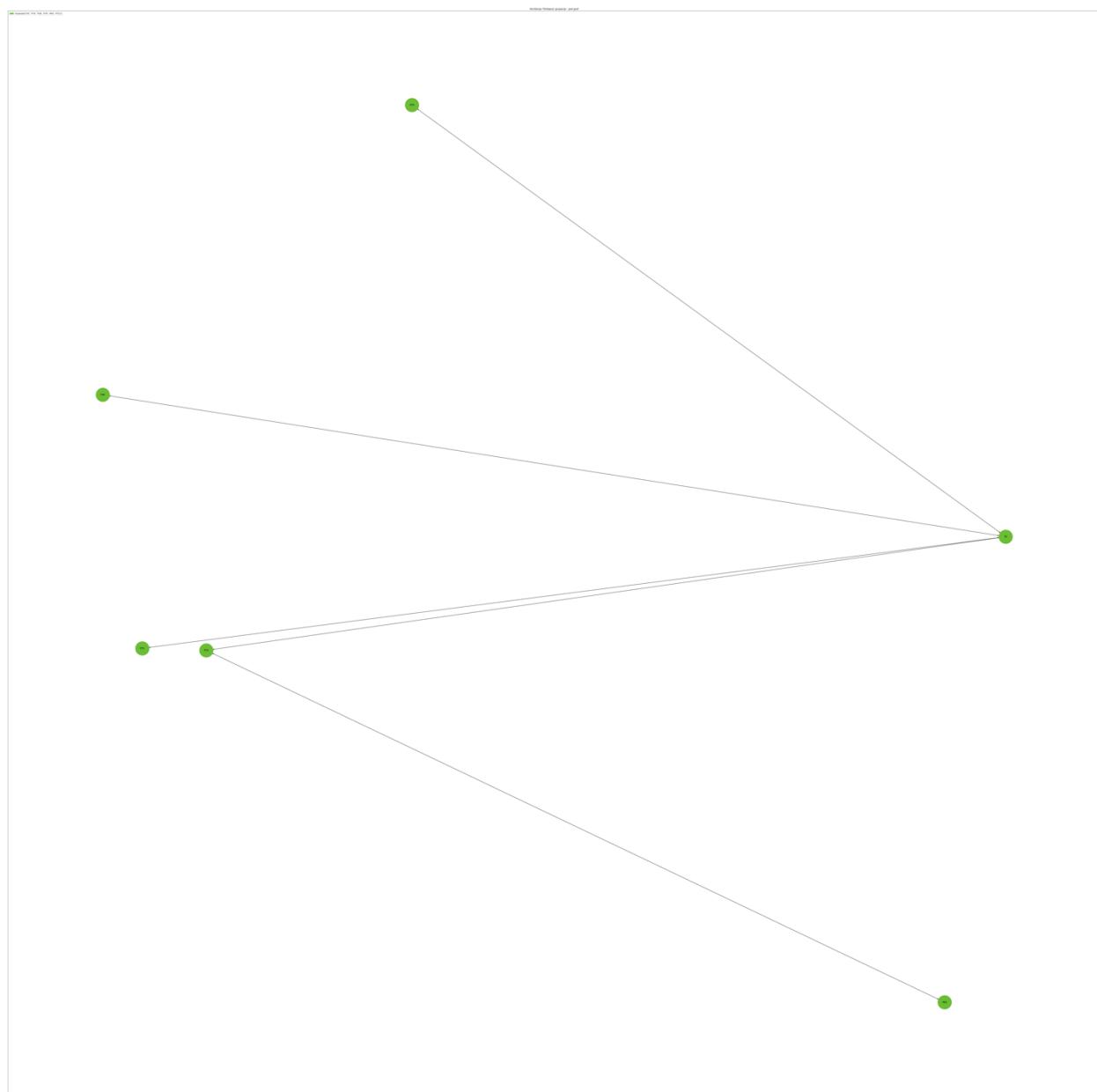
7. Pod graf [6]

	source	target	weight	group	
197	P59	P100	4	6	
217	P99	P100	4	6	
218	P100	P59	4	6	
230	P6	P13	2	6	
231	P6	P59	2	6	
234	P13	P6	2	6	
235	P13	P100	2	6	
265	P59	P99	2	6	
266	P59	P99	2	6	
285	P99	P6	2	6	
293	P100	P13	2	6	
294	P100	P99	2	6	
295	P100	P104	2	6	
297	P104	P59	2	6	
	ID	LABEL	LAT	LNG	group
5	P6	P6	28.965160	-13.555030	6
12	P13	P13	36.533611	-6.299444	6
58	P59	P59	28.100000	-15.416667	6
89	P99	P99	28.500820	-13.862830	6
98	P99	P99	28.683980	-17.764570	6
99	P100	P100	28.468239	-16.254616	6
103	P104	P104	37.389090	-5.984450	6



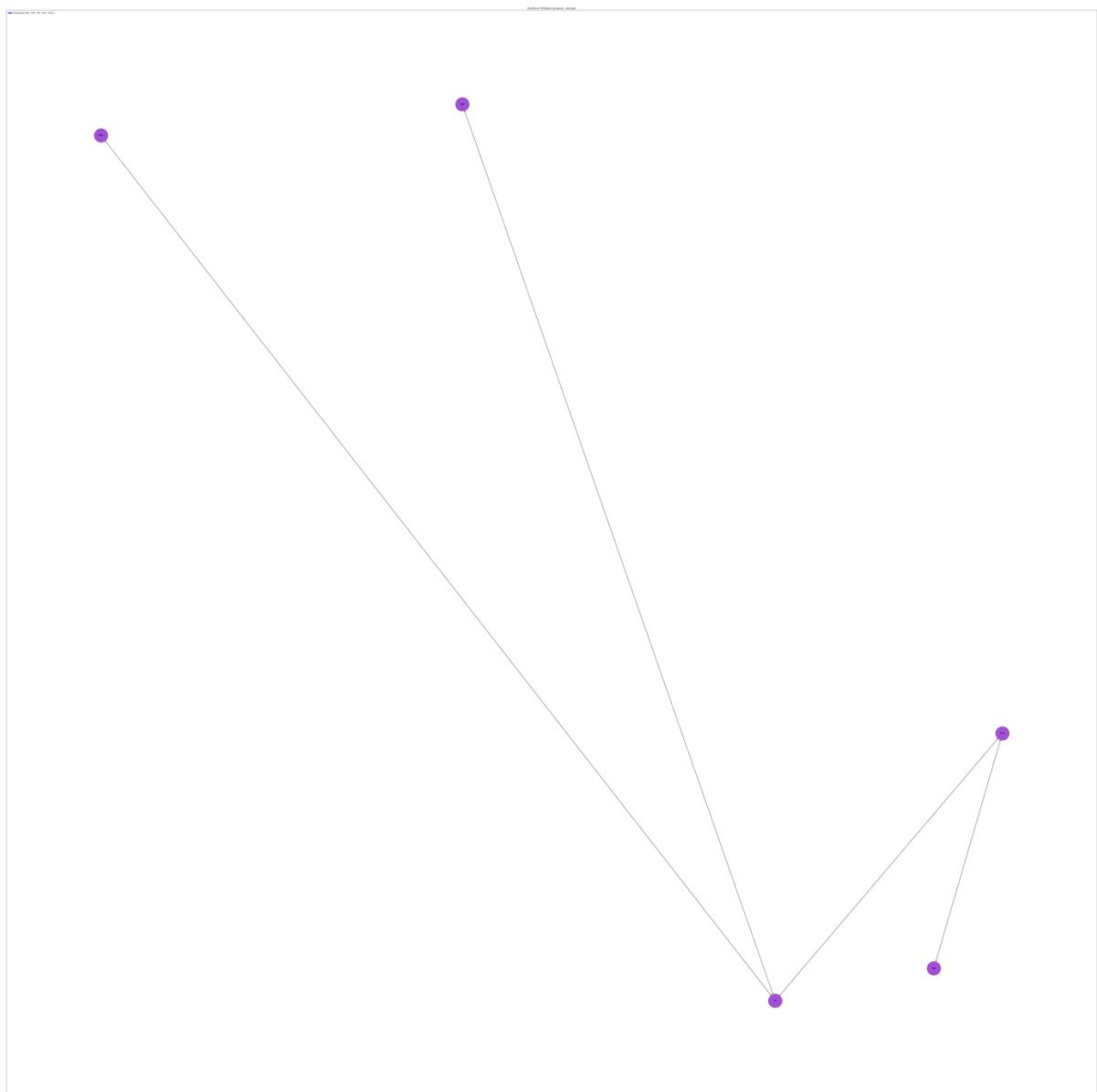
8. Pod graf [7]

	source	target	weight	group
	ID LABEL LAT LNG group			
60	P2	P75	18	7
69	P69	P74	16	7
70	P74	P69	16	7
71	P75	P2	16	7
89	P74	P2	14	7
146	P2	P38	6	7
156	P38	P2	6	7
176	P2	P74	4	7
177	P2	P79	4	7
203	P79	P2	4	7



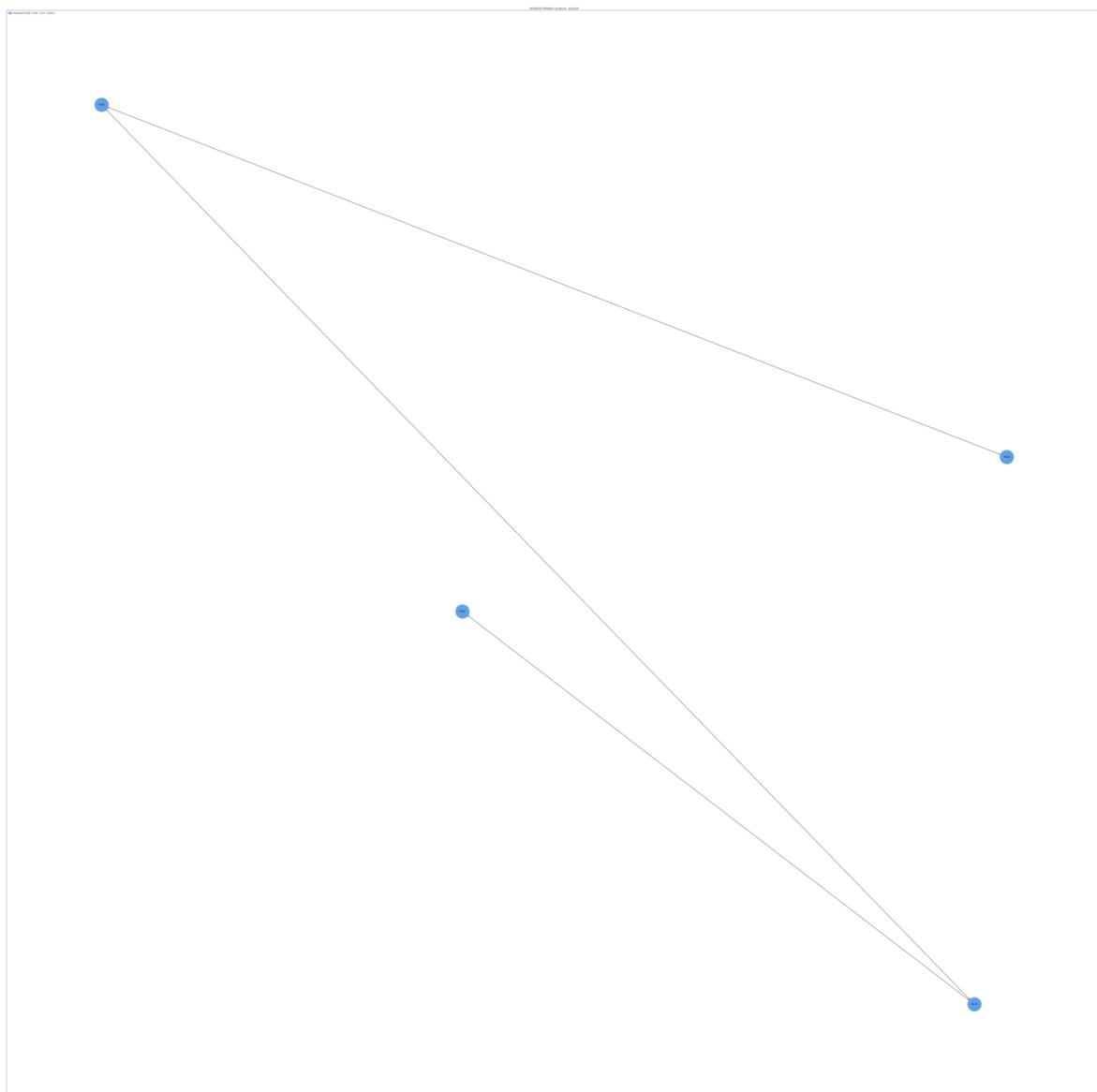
9. Pod graf [8]

```
--> source target weight group
6    P16    P58    92    8
7    P58    P16    92    8
9    P9     P16    80    8
10   P16    P9     80    8
40   P9     P65    34    8
41   P65    P9     34    8
56   P9     P46    24    8
57   P46    P9     24    8
ID LABEL LAT LNG group
8    P9     P9    54.583333 -5.933333 8
15   P16    P16   54.976470 -5.026880 8
45   P46    P46   54.041990 -2.894470 8
57   P58    P58   54.857800 -5.823620 8
64   P65    P65   53.416667 -3.000000 8
```



10. Pod graf [9]

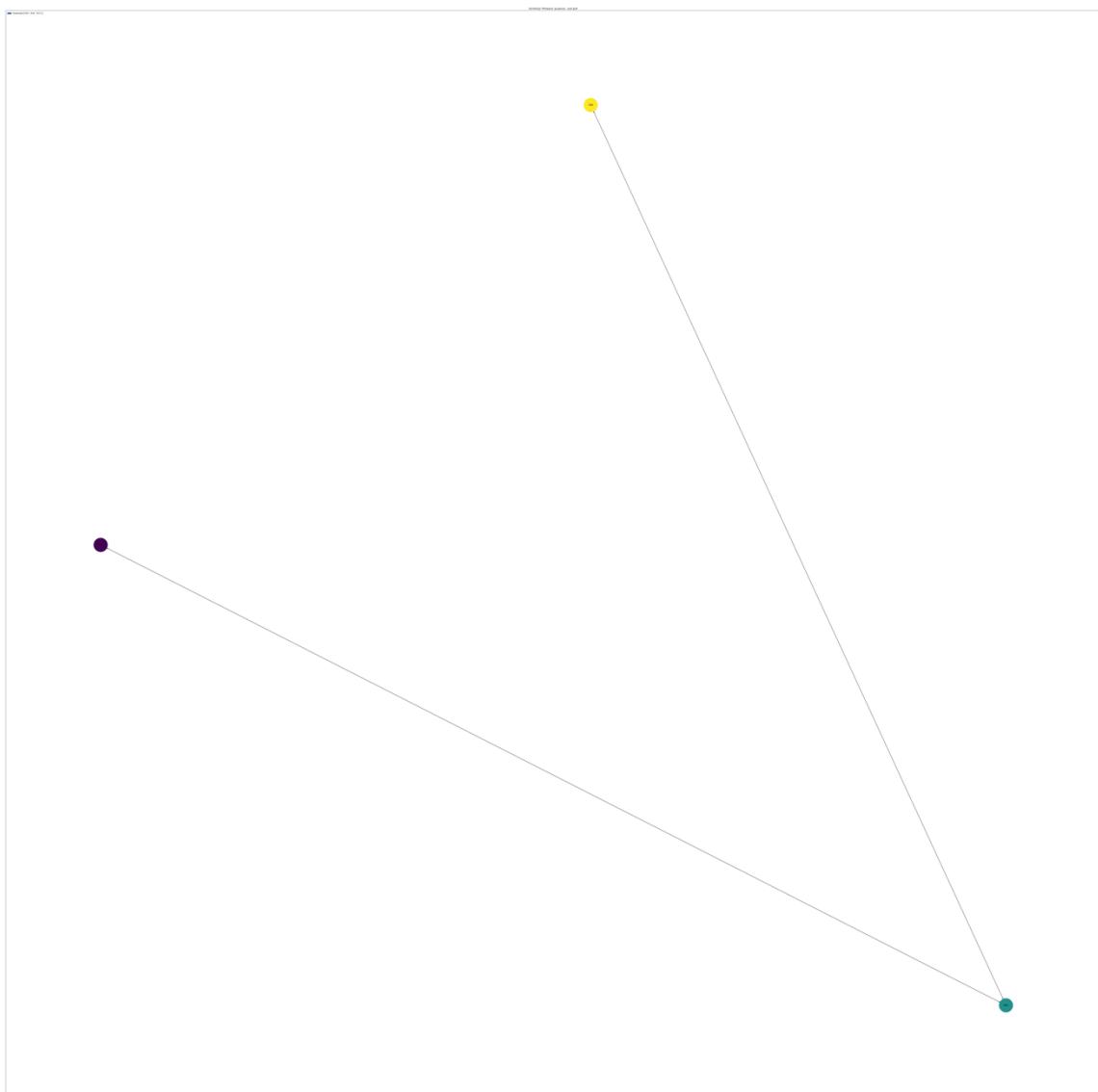
	source	target	weight	group	
8	P108	P124	82	9	
14	P124	P108	80	9	
20	P113	P108	54	9	
21	P108	P113	53	9	
98	P102	P113	14	9	
100	P113	P102	14	9	
	ID	LABEL	LAT	LNG	group
101	P102	P102	54.515910	13.633190	9
107	P108	P108	53.910030	14.247570	9
112	P113	P113	55.375136	13.156911	9
123	P124	P124	55.429659	13.826415	9



11. Pod graf [10] – anomalija u apliciranju boja. Veze i bridovi dobri

	source	target	weight	group
0	P17	P26	444	10
1	P26	P17	402	10
2	P26	P28	152	10
3	P28	P26	152	10

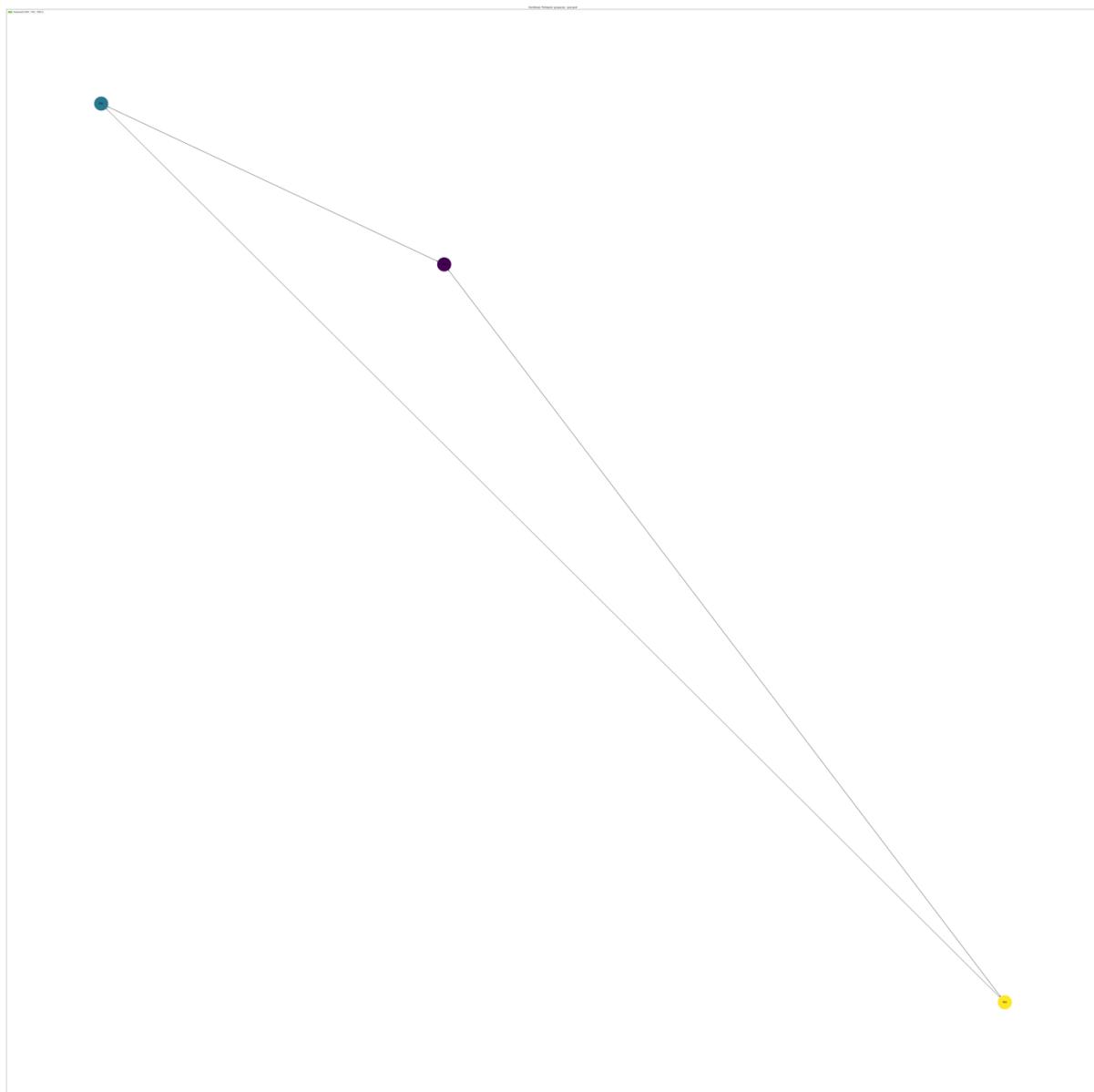
	ID	LABEL	LAT	LNG	group
16	P17	P17	50.958102	1.852055	10
25	P26	P26	51.127870	1.313400	10
27	P28	P28	51.050000	2.366667	10



12. Pod graf [11] – anomalija u apliciranju boja. Veze i bridovi dobri

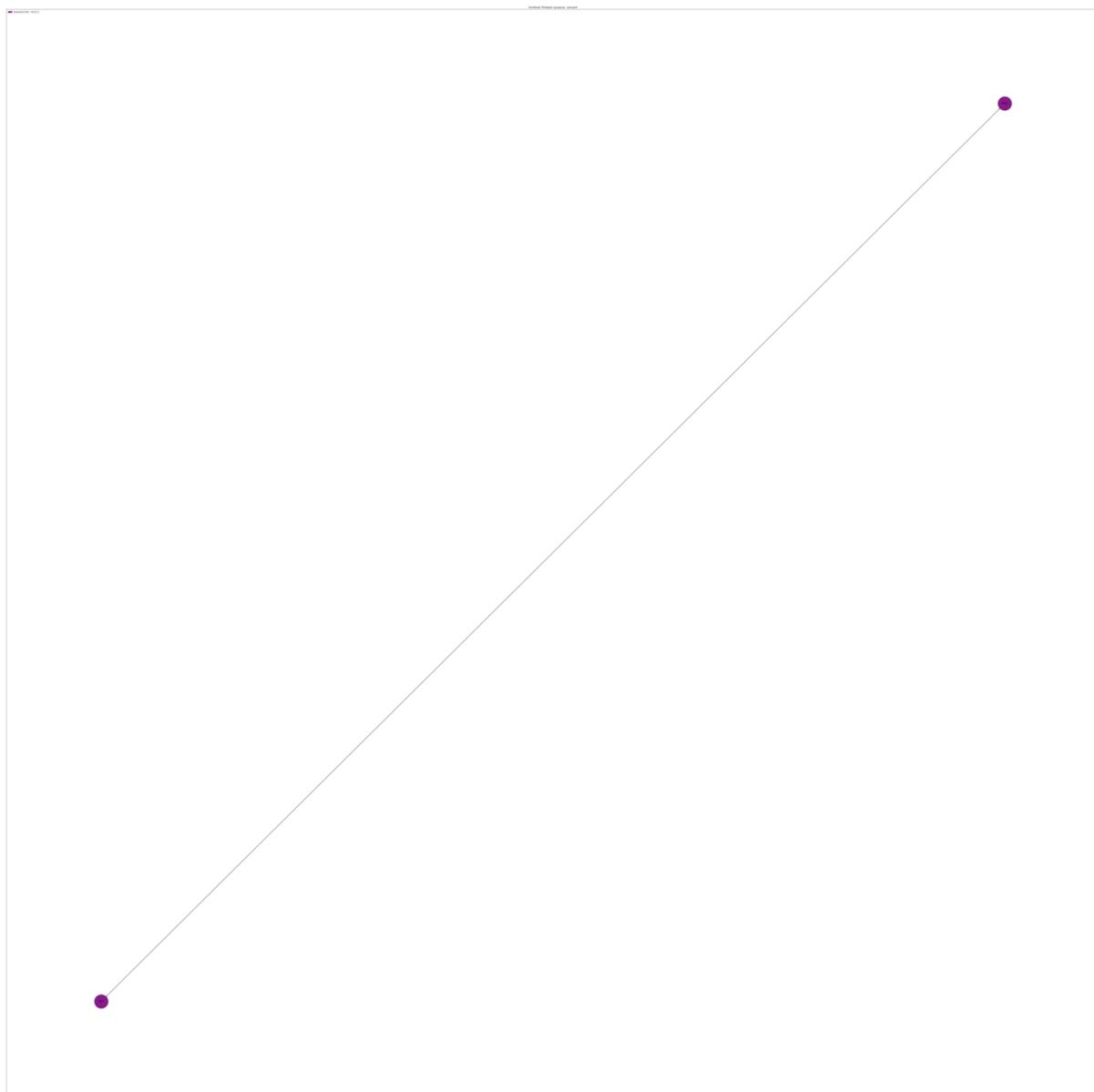
	source	target	weight	group	
191	P42	P64	4	11	
	ID	LABEL	LAT	LNG	group
199	P64	P60	4	11	
267	P60	P42	2	11	
268	P60	P64	2	11	
271	P64	P42	2	11	

	ID	LABEL	LAT	LNG	group
41	P42	P42	32.815556	34.989166	11
59	P60	P60	37.714170	24.057660	11
63	P64	P64	34.675000	33.033333	11



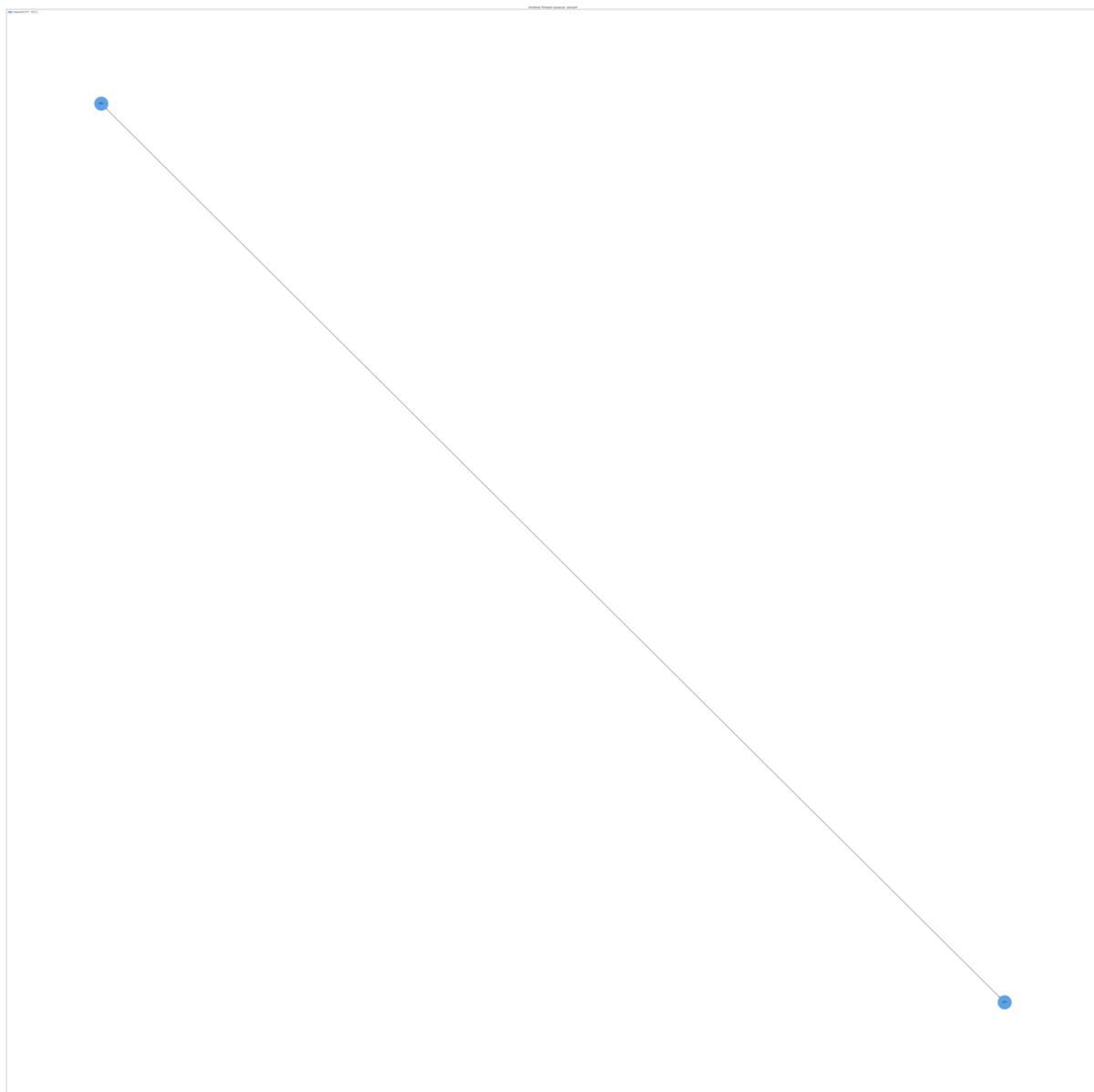
13. Pod graf [12]

	source	target	weight	group	
52	P41	P121	26	12	
55	P121	P41	26	12	
	ID	LABEL	LAT	LNG	group
40	P41	P41	56.415783	10.878245	12
120	P121	P121	57.105568	12.250775	12



14. Pod graf [13] – najmanji pod graf

	source	target	weight	group	
26	P25	P77	42	13	
29	P77	P25	42	13	
	ID	LABEL	LAT	LNG	group
24	P25	P25	49.92299	1.07748	13
76	P77	P77	50.79307	0.04557	13



Dodatno sam napravio analizu grafa G:

- **Modularnost** - Vrijednost modularnosti kreće se od **-1** do **1**, s vrijednostima blizu **1** koje označavaju **dobro grupiranje ili strukturu zajednice na grafikonu**, a vrijednosti blizu **-1** ukazuju na **loše klasteriranje**. Vrijednost modularnosti od približno **0,76** je dobar rezultat koji ukazuje na da na grafikonu postoji **snažna struktura zajednice** te odgovara na postavljeno pitanje ovog zadatka.

```
In [32]: 1 modularity = nx.algorithms.community.modularity(G, nx.algorithms.community.greedy_modularity_communities(G))
2
3 print("Modularnost grafa (G): ", modularity)
```

Modularnost grafa (G): 0.7616384473891873

- **Ispis bridova i vrhova po zajednici** – varijabla vraća broj bridova i vrhova po zajednicama uz cilj dodatne provjere pod grafova.

```
5 for i, community in enumerate(communitys):
6     print(f"Community {i+1}:")
7     subgraph = G.subgraph(community)
8     print(f"Number of nodes: {subgraph.number_of_nodes()}")
9     print(f"Number of edges: {subgraph.number_of_edges()}")
```

Community 1:	Community 8:
Number of nodes: 25	Number of nodes: 6
Number of edges: 67	Number of edges: 10
Community 2:	Community 9:
Number of nodes: 18	Number of nodes: 5
Number of edges: 48	Number of edges: 8
Community 3:	Community 10:
Number of nodes: 16	Number of nodes: 4
Number of edges: 44	Number of edges: 6
Community 4:	Community 11:
Number of nodes: 14	Number of nodes: 3
Number of edges: 26	Number of edges: 4
Community 5:	Community 12:
Number of nodes: 12	Number of nodes: 3
Number of edges: 29	Number of edges: 5
Community 6:	Community 13:
Number of nodes: 8	Number of nodes: 2
Number of edges: 13	Number of edges: 2
Community 7:	Community 14:
Number of nodes: 7	Number of nodes: 2
Number of edges: 14	Number of edges: 2

- **Omjer bridova u zajednici i izvan zajednice** - Vrijednost omjera bridova unutar zajednica prema bridovima između zajednica pokazatelj je koliko su dobro definirane zajednice na grafikonu. Visoka vrijednost za ovaj omjer znači da postoji mnogo više rubova unutar zajednica u usporedbi između zajednica, što sugerira da su zajednice dobro definirane i različite jedna od druge.

U mom slučaju, vrijednost omjera je približno 6,47 te ta vrijednost pokazuje da postoji mnogo više bridova unutar zajednica nego između zajednica. Ovaj proces dodatna je potvrda da su zajednice jedinstvene i dobro podijeljene.

```
17 total_edges = G.number_of_edges()  
18  
19  
20 ratio = edges_within / (total_edges - edges_within)  
21  
22 print(f"Omjer bridova u zajednici i bridova izvan zajednice je: {ratio}")
```

Omjer bridova u zajednici i bridova izvan zajednice je: 6.465116279069767

Zadatak (2)

Izlazni stupanj, ulazni stupanj i distribucija stupnjeva

Stvorio sam **tri stupčasta grafikona** pomoću biblioteke **Plotly Express**.

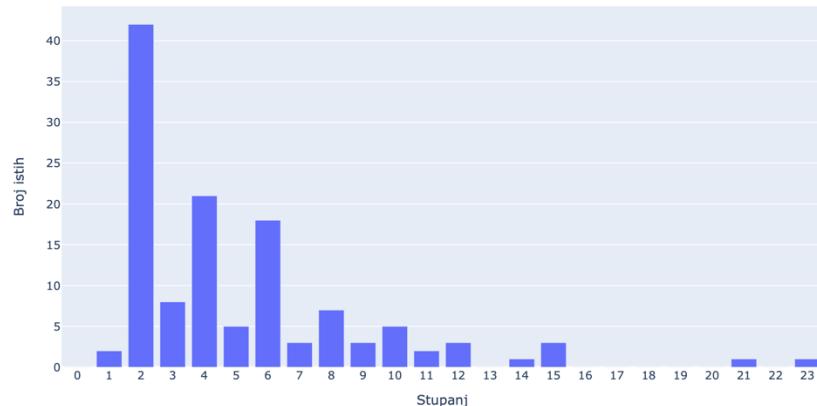
Plotly Express biblioteka omogućuje lako i uredno crtanje grafova sa unaprijed definiranim korisnim funkcijama poput **povećanja i smanjenja segmenata, suženja promatranog intervala te legende** prilikom prelaska kursorom preko grafa.

Prvi grafikon prikazuje **distribuciju unutarnjih stupnjeva čvorova** na grafu G, drugi grafikon prikazuje **distribuciju vanjskih stupnjeva čvorova** na grafu, a treći grafikon prikazuje **distribuciju ukupnih stupnjeva** (zbroj in-stupnjeva i out-degree) **čvorova** u grafu.

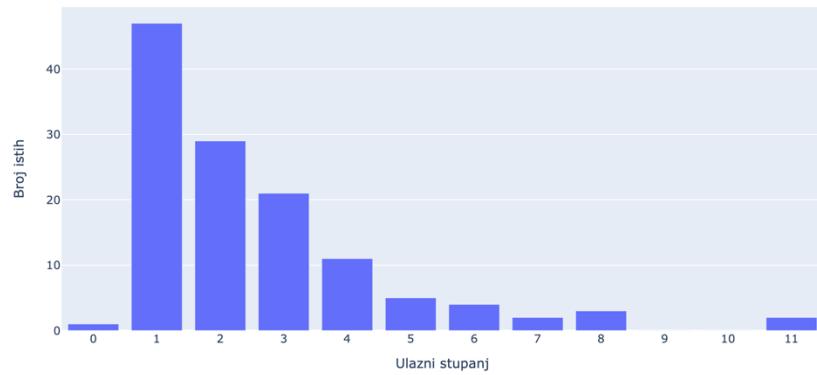
```
In [39]: 1 import networkx as nx
2 import plotly.express as px
3 import numpy as np
4
5 in_degree_sequence = np.array([d for n, d in G.in_degree()])
6 out_degree_sequence = np.array([d for n, d in G.out_degree()])
7
8 in_degree_hist, in_degree_bin_edges = np.histogram(in_degree_sequence, bins=range(in_degree_sequence.max() + 2))
9 out_degree_hist, out_degree_bin_edges = np.histogram(out_degree_sequence, bins=range(out_degree_sequence.max() + 10))
10
11 fig1 = px.bar(x=list(range(len(in_degree_hist))), y=in_degree_hist, title="Ulazni stupanj")
12 fig1.update_layout(xaxis_title="Ulazni stupanj", yaxis_title="Broj istih")
13 fig1.update_xaxes(tickmode="array", tickvals=list(range(len(in_degree_hist))), ticktext=list(range(len(in_degree_hist))))
14
15 fig2 = px.bar(x=list(range(len(out_degree_hist))), y=out_degree_hist, title="Izlazni stupanj")
16 fig2.update_layout(xaxis_title="Izlazni stupanj", yaxis_title="Broj istih")
17
18 fig2.update_xaxes(tickmode="array", tickvals=list(range(len(out_degree_hist))), ticktext=list(range(len(out_degree_hist))))
19
20 degree_sequence = [d for n, d in G.degree()]
21 degree_count = nx.degree_histogram(G)
22
23 fig = px.bar(x=range(len(degree_count)), y=degree_count)
24 fig.update_layout(title_text="Distribucija stupnjeva", xaxis_title="Stupanj", yaxis_title="Broj istih")
25 fig.update_xaxes(tickmode="array", tickvals=list(range(len(degree_count))), ticktext=list(range(len(degree_count))))
26
27 fig.show()
28 fig1.show()
29 fig2.show()
```

Prikazane su vizualizacije: **izlazni stupanj, ulazni stupanj i distribucija stupnjeva**.

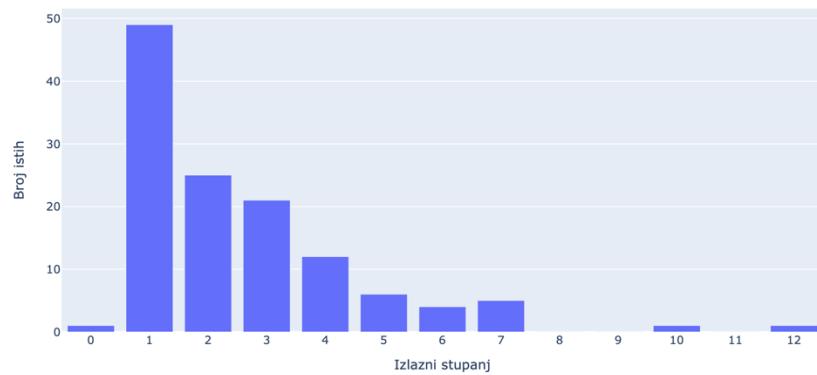
Distribucija stupnjeva



Ulazni stupanj



Izlazni stupanj



Možemo opaziti da su ulazni i izlazni stupnjevi vrhova jednaki. Najčešće ulazne i izlazne veze grafova su 2 i 3.

Jako povezane komponente, dijametar, radius

Izračunao sam **jako povezane komponente** usmjerenog grafa G pomoću funkcije `nx.strongly_connected_components`, koja vraća **popis skupova**, gdje svaki skup predstavlja **jako povezanu komponentu u grafu**.

Dodatno sam izračunao **dijametar i radius**.

```
In [41]: 1 strongly_connected_components = nx.strongly_connected_components(G)
2
3 for component in strongly_connected_components:
4     subgraph = G.subgraph(component)
5     diameter = nx.diameter(subgraph)
6     radius = nx.radius(subgraph)
7     print("ID vrhova u komponenti:", component)
8     print("Diametar komponente:", diameter)
9     print("Radius komponente:", radius)
10    print("\n")
```

```
ID vrhova u komponenti: {'P54', 'P30', 'P43', 'P63', 'P65', 'P56', 'P58', 'P87', 'P125', 'P76', 'P11', 'P113', 'P39',
', 'P91', 'P101', 'P86', 'P85', 'P108', 'P29', 'P19', 'P102', 'P112', 'P32', 'P24', 'P48', 'P1', 'P78', 'P123', 'P9
3', 'P16', 'P57', 'P67', 'P3', 'P53', 'P5', 'P105', 'P119', 'P55', 'P44', 'P72', 'P95', 'P23', 'P89', 'P9', 'P94',
'P33', 'P124', 'P52', 'P62', 'P46', 'P81', 'P34', 'P51', 'P10', 'P106', 'P14', 'P40', 'P118', 'P107', 'P117', 'P35
', 'P47', 'P111', 'P80', 'P45', 'P21', 'P109', 'P70', 'P97', 'P61', 'P27', 'P96', 'P36', 'P31'}
Diametar komponente: 11
Radius komponente: 6
```

```
ID vrhova u komponenti: {'P74', 'P79', 'P38', 'P2', 'P69', 'P75'}
Diametar komponente: 3
Radius komponente: 2
```

```
ID vrhova u komponenti: {'P4', 'P116', 'P115', 'P22', 'P110', 'P7', 'P73', 'P92', 'P103', 'P12', 'P50', 'P49', 'P66
', 'P84', 'P83', 'P18', 'P82', 'P8', 'P37', 'P71', 'P20', 'P88', 'P68', 'P98', 'P15', 'P120'}
Diametar komponente: 7
Radius komponente: 4
```

```
ID vrhova u komponenti: {'P13', 'P59', 'P104', 'P100', 'P99', 'P6', 'P90'}
Diametar komponente: 4
Radius komponente: 2
```

```
ID vrhova u komponenti: {'P26', 'P28', 'P17'}
Diametar komponente: 2
Radius komponente: 1
```

Zadatak (3)

Centralnost međupoloženosti

Izračunao sam **centralnost međuposloženosti grafa** kako bi utvrdio potencijalne slabosti tj. gradove koji bi mogli postati izolirani. Prvo sam izračunao međuposloženosti pomoću naredbe **nx.betweenness_centrality** koja sadrži vrhove i bridove grafa G te uzima u obzir karakteristike grafa poput **usmjerenosti** i **težinskih vrijednosti bridova**. Podaci su potom i normalizirani.

```
In [54]: 1 bet_cent = nx.betweenness_centrality(G, weight='weight', normalized=True)
2 print("Centralnost međupoloženosti iznosi:")
3 print(bet_cent)
```

Nakon pokretanja vidljiva je lista u obliku **tuple** tj. **{[oznaka_node-a]:[centralnost međuposloženosti za pripadajući node]}**.

```
Centralnost međupoloženosti iznosi:
{'P1': 0.0, 'P2': 0.001180173092053012, 'P3': 0.009801993181222135, 'P4': 0.0, 'P5': 0.0124562898854795, 'P6': 0.0
005245213742460005, 'P7': 0.010883818515604509, 'P8': 0.0, 'P9': 0.02779963283503803, 'P10': 0.05953317597692106, 'P11': 0.01740755310778914, 'P12': 0.008720167846839759, 'P13': 0.0003933910306845004, 'P14': 0.0, 'P15': 0.0025570416994492523, 'P16': 0.00944138473642801, 'P17': 0.0, 'P18': 0.17571466037241017, 'P19': 0.04183057959611854, 'P20': 0.004196170993968004, 'P21': 0.023210070810385522, 'P22': 0.0032782585890375033, 'P23': 0.0, 'P24': 0.0, 'P25': 0.0, 'P26': 0.00013113034356150013, 'P27': 0.11401783372672436, 'P28': 0.0, 'P29': 0.0, 'P30': 0.007277734067663257, 'P31': 0.0, 'P32': 0.0, 'P33': 0.013768686073957514, 'P34': 0.0, 'P35': 0.0, 'P36': 0.016565040650406503, 'P37': 0.0, 'P38': 0.0, 'P39': 0.0, 'P40': 0.051862050878573304, 'P41': 0.0, 'P42': 0.0, 'P43': 0.016746437625666577, 'P44': 0.0, 'P45': 0.1415114957601189, 'P46': 0.0, 'P47': 0.0, 'P48': 0.011358073258151934, 'P49': 0.0021527231401346275, 'P50': 0.013375295043273012, 'P51': 0.032486449864498654, 'P52': 0.0, 'P53': 0.0, 'P54': 0.0, 'P55': 0.05704169944925256, 'P56': 0.04510883818515604, 'P57': 0.04911268467523386, 'P58': 0.0, 'P59': 0.0007212168895882507, 'P60': 0.0, 'P61': 0.0, 'P62': 0.05730396013637556, 'P63': 0.018620508785733018, 'P64': 6.5556517178075007e-05, 'P65': 0.03619197482297404, 'P66': 0.016150887315324765, 'P67': 0.10310669638954453, 'P68': 0.0, 'P69': 0.0, 'P70': 0.0, 'P71': 0.0067532126934172565, 'P72': 0.00944138473642801, 'P73': 0.0, 'P74': 0.0005245213742460005, 'P75': 0.0, 'P76': 0.0, 'P77': 0.0, 'P78': 0.00944138473642801, 'P79': 0.0, 'P80': 0.00944138473642801, 'P81': 0.004720692368214005, 'P82': 0.011594107876562636, 'P83': 0.0033001136462977535, 'P84': 0.001966955153422502, 'P85': 0.0, 'P86': 0.01088381851560451, 'P87': 0.0, 'P88': 0.0017921146953405018, 'P89': 0.03829006031995804, 'P90': 0.0001966955153422502, 'P91': 0.049204475915726915, 'P92': 0.012752425911355888, 'P93': 0.0114083988950511, 'P94': 0.027275111460792027, 'P95': 0.011309992132179387, 'P96': 0.0, 'P97': 0.07376081825334382, 'P98': 0.008545327388757759, 'P99': 0.0, 'P100': 0.0007212168895882507, 'P101': 0.02924206661421453, 'P102': 0.0, 'P103': 0.00022947810123262523, 'P104': 0.00013113034356150013, 'P105': 0.0, 'P106': 0.136100183582481, 'P107': 0.0, 'P108': 0.00944138473642801, 'P109': 0.0, 'P110': 0.0, 'P111': 0.0, 'P112': 0.11695078241104989, 'P113': 0.02779963283503803, 'P114': 0.0, 'P115': 0.000983477576711251, 'P116': 0.00786782061369006, 'P117': 0.004720692368214005, 'P118': 0.03029110936270653, 'P119': 0.0, 'P120': 0.005813445231226505, 'P121': 0.0, 'P122': 0.0, 'P123': 0.0, 'P124': 0.0, 'P125': 0.028389719381064778}
```

Vrhovi s malim vrijednostima imaju izrazito veliku šansu (pri micanju bridova grafa) ostati nepovezani. Zbog ljepe vizualizacije izbacio sam sve vrhove s vrijednosti **0** (to su najosjetljiviji gradovi te sam ih prikazao u sljedećem ispisu).

```
5 bet_cent_df = pd.DataFrame(list(bet_cent.items()), columns=['Vrh', 'Centralnost međupoloženosti'])
6
7 bet_cent_null = bet_cent_df[bet_cent_df['Centralnost međupoloženosti'] <= 0]
8
9 print('\nMicanje vrhova koji imaju vrijednost jednaku 0:\n{}'.format(bet_cent_null))
10
```

Micanje vrhova koji imaju vrijednost jednaku 0:

Vrh Centralnost međupoštenosti

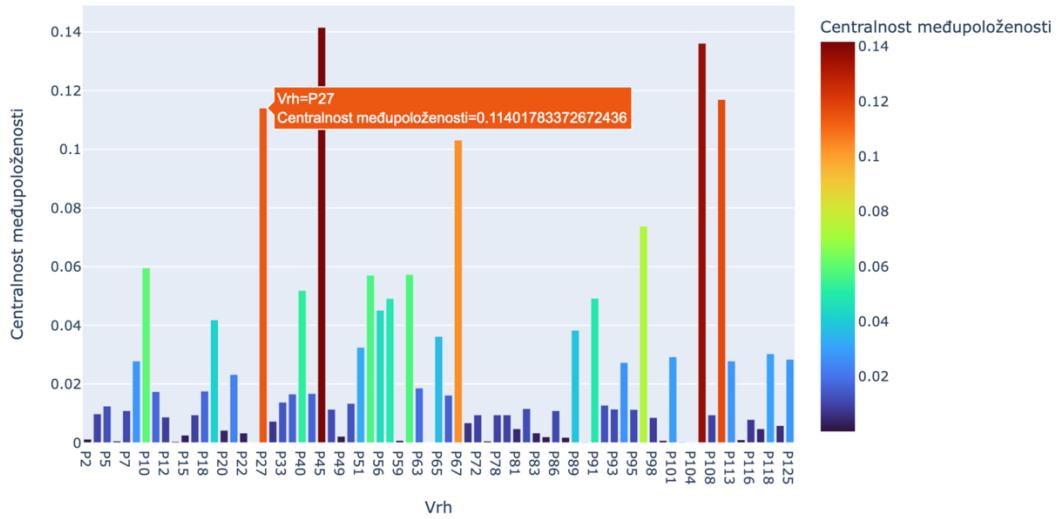
0	P1	0.0
3	P4	0.0
7	P8	0.0
13	P14	0.0
16	P17	0.0
22	P23	0.0
23	P24	0.0
24	P25	0.0
27	P28	0.0
28	P29	0.0
30	P31	0.0
31	P32	0.0
33	P34	0.0
34	P35	0.0
36	P37	0.0
37	P38	0.0
38	P39	0.0
40	P41	0.0
41	P42	0.0
43	P44	0.0
45	P46	0.0
46	P47	0.0
51	P52	0.0
52	P53	0.0
53	P54	0.0
57	P58	0.0
59	P60	0.0
60	P61	0.0
67	P68	0.0
68	P69	0.0
69	P70	0.0
72	P73	0.0
74	P75	0.0
75	P76	0.0
76	P77	0.0
78	P79	0.0
84	P85	0.0
86	P87	0.0
95	P96	0.0
98	P99	0.0
101	P102	0.0
104	P105	0.0
106	P107	0.0
108	P109	0.0
109	P110	0.0
110	P111	0.0
113	P114	0.0
118	P119	0.0
120	P121	0.0
121	P122	0.0
122	P123	0.0
123	P124	0.0

Ostale vrhove sam prikazao pomoću `plotly.express`:

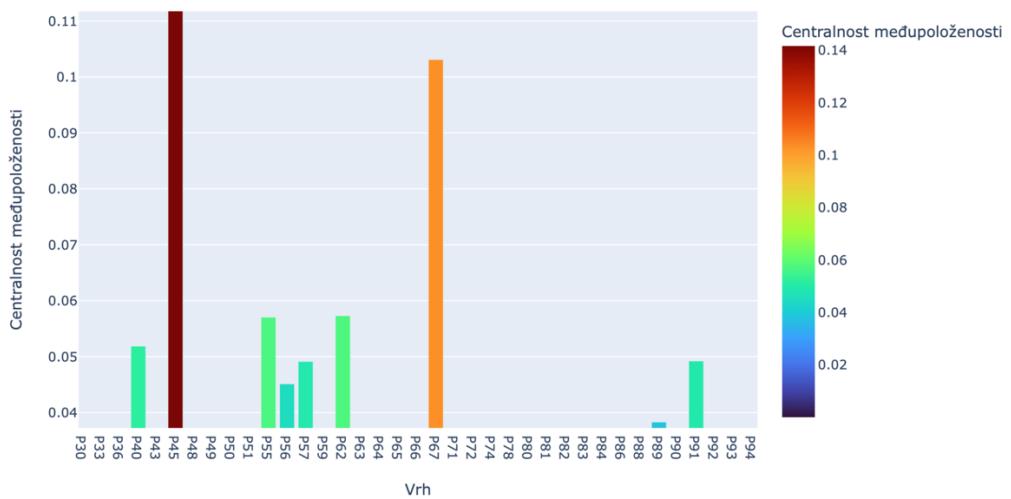
- Dodatno skalirana os X (oznake su bolje vidljive)
- Gradient boja označuje centralnost međuposloženosti čvora



- Dodano dinamičko alociranje vrijednosti iz pandas data set-a (prelazak kursora preko grafa)



- Mogućnost izoliranja regija



Ova vizualizacija prikazuje vrhove (plavih boja) koji pri micanju vrlo malo čvorova postaju odvojene komponente, a kao takvi nemaju mogućnost komunicirati s ostatkom grafa.
Vrhovi: **P45, P106, P112, P27** i **P67** najotporniji su na promjene tj. micanje bridova u grafu.