

Sprawozdanie TASS

Projekt nr 1

Sebastian Smoliński

Nr albumu 269056

Semestr 20Z

Zadanie A

Analiza sieci w narzędziu Pajek

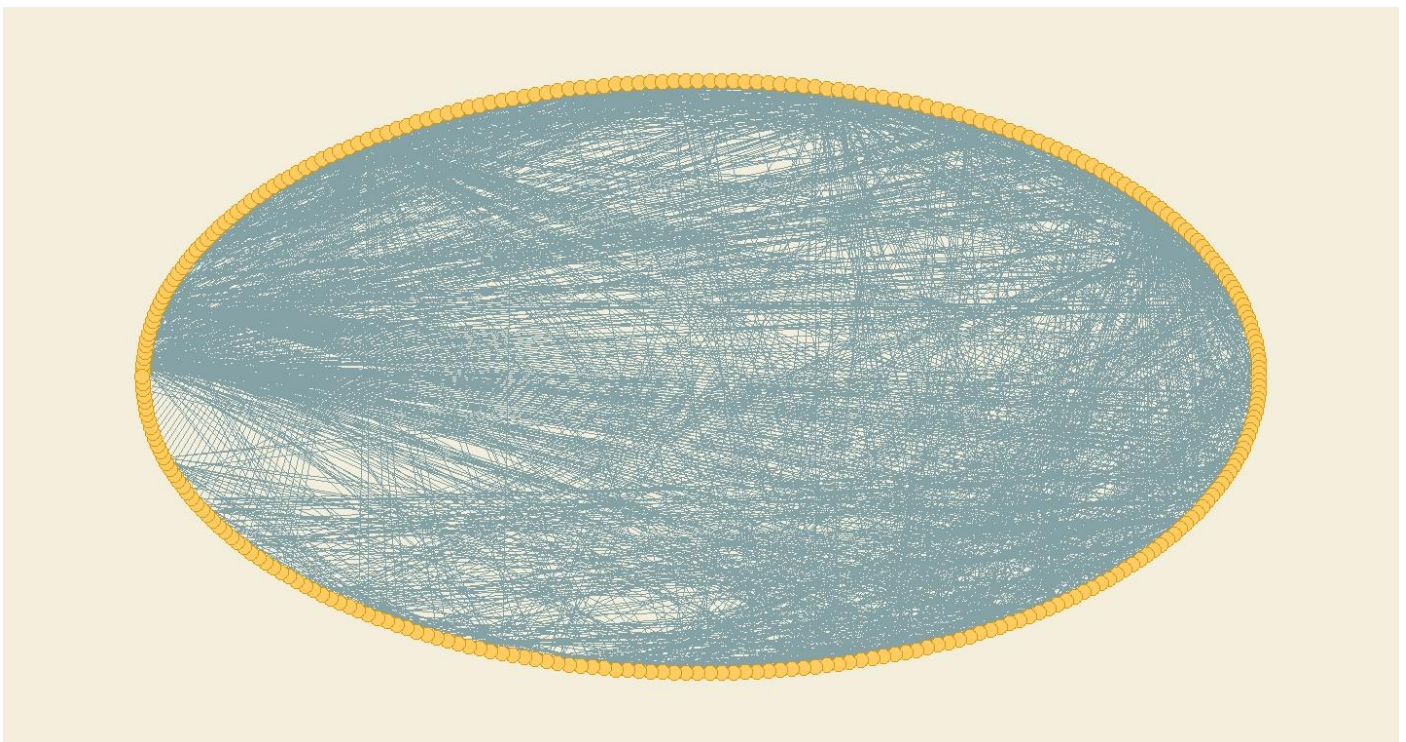
Do wykonania polecenia została wykorzystana sieć nr 4, czyli „Połączenia nerwowe nicienia *Caenorhabditis elegans*”. Wybrano zestaw nr 4 na podstawie obliczeń wykonanych poniżej.

$$269056 \bmod 7 = 38436 + 4$$

Zadania do wykonania:

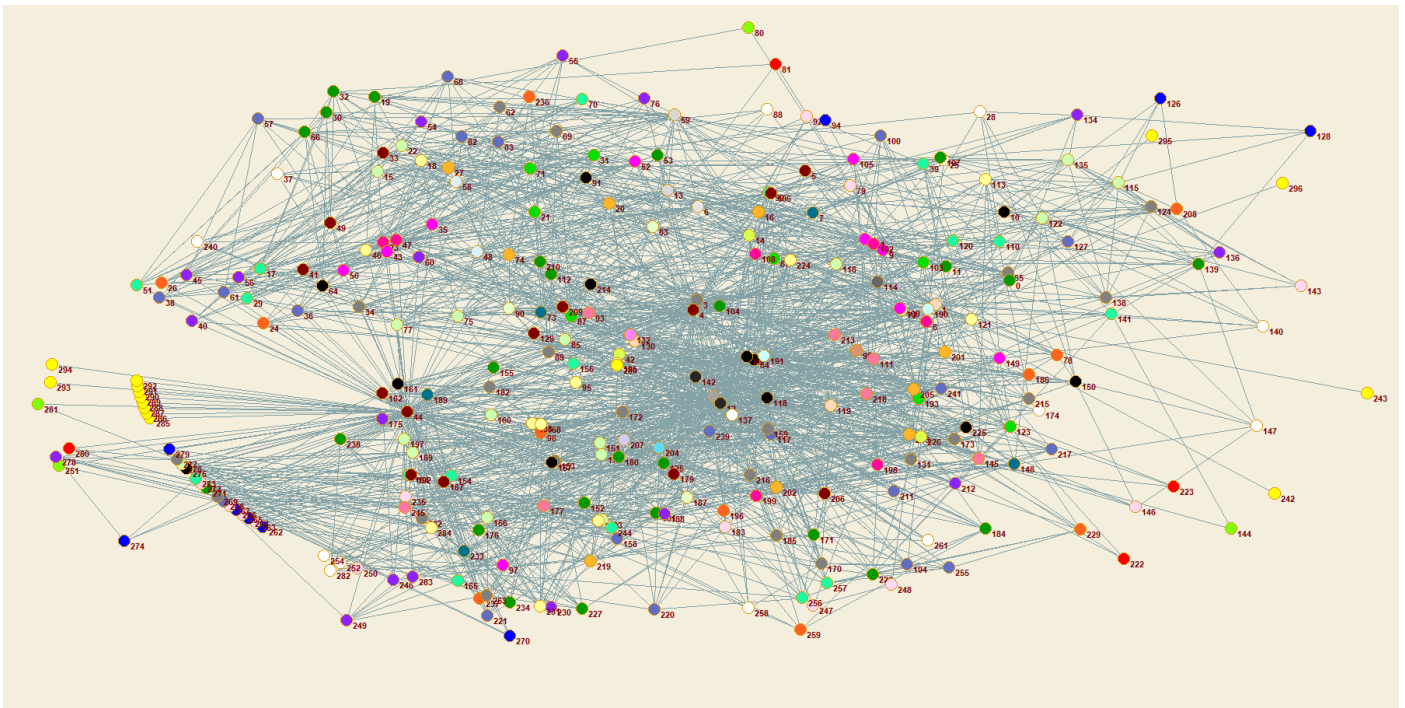
- zbadaj, jaki jest rząd i rozmiar całej sieci, a następnie wyodrębnij największą składową spójną, zbadaj jej rząd i rozmiar (1 pkt.);
- wykreśl największą składową spójną i skomentuj wynik (1);
- przeprowadź grupowanie metodą Warda z metryką d1 (odległość dwóch węzłów to liczba sąsiadów połączonych tylko z jednym z nich) (1);
- wykreśl dendrogram (1) i zaproponuj cięcie (1);
- wykreśl wyodrębnione grupy (2).

Schemat otrzymany przy użyciu funkcji: Draw -> Network (->Layout -> Circular -> Original)



Rysunek 1 Schemat sieci

Schemat sieci (wraz z stopniami wierzchołków) otrzymany przy użyciu funkcji: Draw -> Network -> Layout -> Energy -> Kawada - Kamai -> Separate Components



Rysunek 2 Schemat sieci przy użyciu opisanej funkcji

1. Zbadaj, jaki jest rząd i rozmiar całej sieci, a następnie wyodrębnij największą składową spójną, zbadaj jej rząd i rozmiar (1)

Rząd sieci: 297 węzłów

Rozmiar sieci: 2148 krawędzi

Zrzut ekranu z informacjami o sieci otrzymany z funkcji „Info Network” i wpisaniu w polu zakresu "0 3000"



Rysunek 3 Wyświetlenie opcji sieci

1. C:\Users\Sebastian\OneDrive - Politechnika Warszawska\Pulpit\4.net (297)		
Number of vertices (n): 297		
-----	Arcs	Edges
Total number of lines	0	2148

Number of loops	0	0
Number of multiple lines	0	0

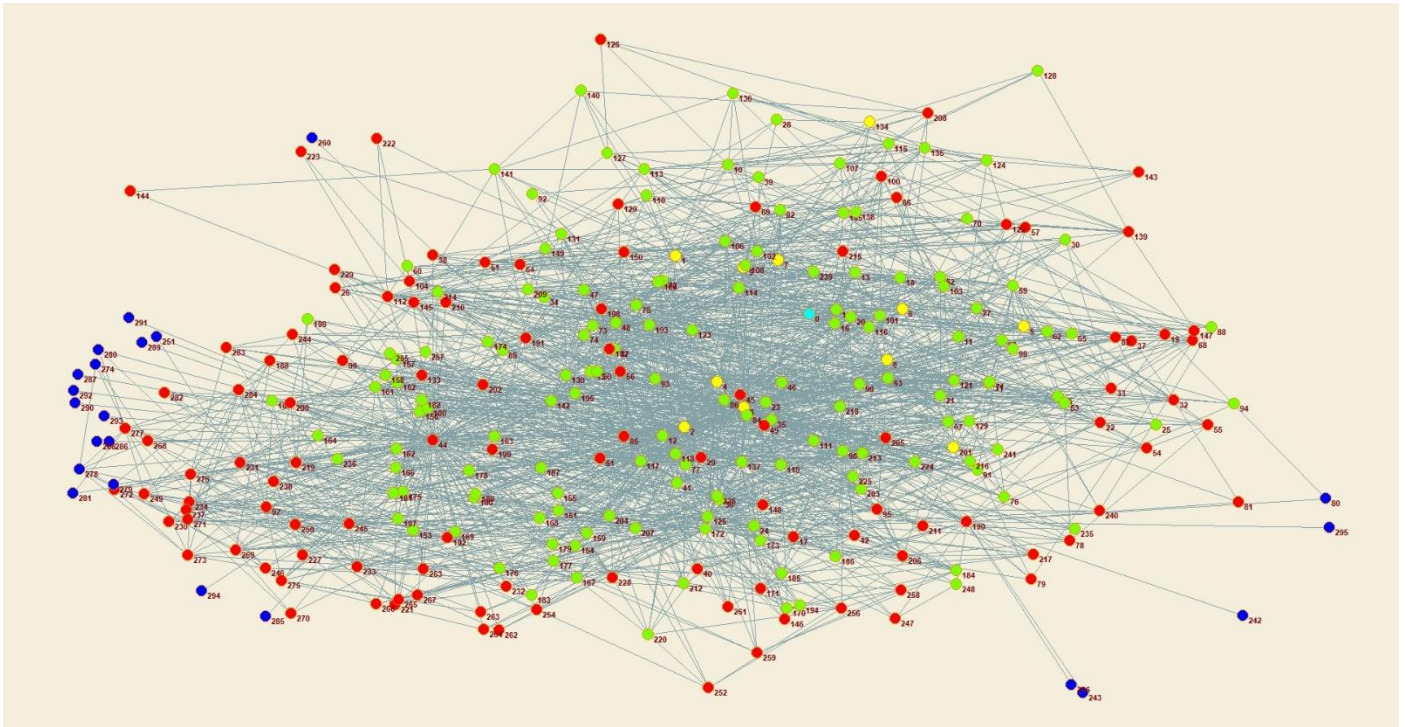
Density1 [loops allowed] = 0.04870251		
Density2 [no loops allowed] = 0.04886705		
Average Degree = 14.46464646		

Tabela 1 Wynik działania funkcji

Składowa spójna odpowiada pierwotnemu grafowi, a więc graf możemy nazwać spójnym. Parametry są następujące:






Rząd sieci: 297 węzłów

Rozmiar sieci: 2148 krawędzi



Rysunek 4 Graf spójny

Oznaczenia kolorów:

-  - węzeł początkowy,
-  - węzeł połączony bezpośrednio z początkowym,
-  - węzeł połączony przez żółty z początkowym,
-  - węzeł połączony przez zielony z początkowym,
-  - węzeł połączony przez czerwony z początkowym.

Oczywiście poszczególne węzły mogą być również połączone z początkowym przez inne węzły, ale to miało tylko objaśnić „hierarchię”.

Poniższy opis otrzymano za pomocą Network -> Create partition -> Components -> Strong

=====

Strong Components

=====

Working...

Number of components: 1

Size of the largest component: 297 vertices (100.000%).

Time spent: 0:00:00

2. Wykreśl największą składową spójną i skomentuj wynik (1)

Odpowiedź zależy od tego jak rozumiemy “wykreślić”.

Jeśli przez usunięcie składowej spójnej, to sieć nie będzie miała wierzchołków, bo graf jest spójny, jeśli cały graf jest składową spójną. Wtedy zasadniczo usuwamy całość.

Jeśli przez “wyrysować/wykreślić/wyznaczyć”, to wtedy cały graf jest jednocześnie składową spójną i ma 297 wierzchołków.

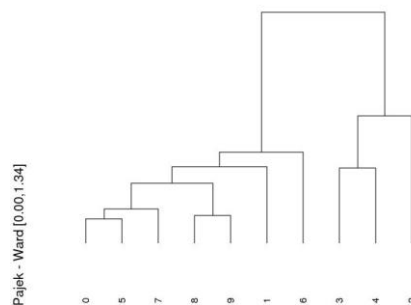
3. Przeprowadź grupowanie metodą Warda z metryką d1 (odległość dwóch węzłów to liczba sąsiadów połączonych tylko z jednym z nich) (1)

Wykonano następujące operacje na wczytanej sieci z pliku:

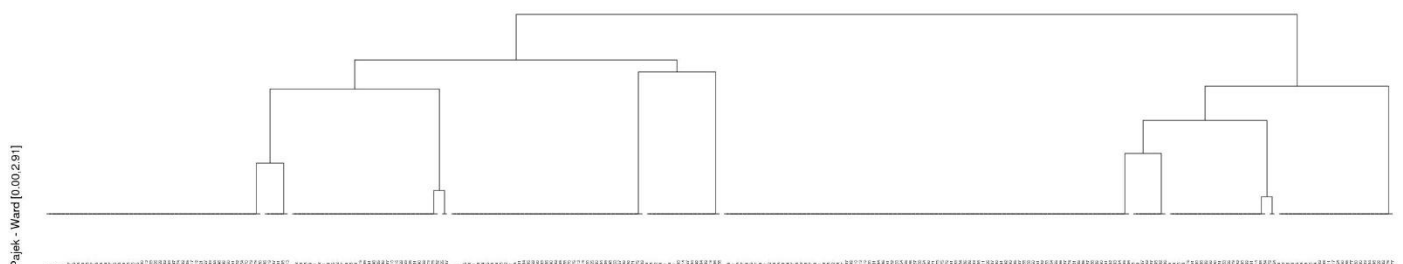
Cluster->Create Complete Cluster – następnie wybrano 297, czyli ilość wierzchołków w pliku. Jeśli interesuje nas bardziej zwężta forma dendrogramu, to należy wskazać niższą wartość w polu wymiaru. Następnie wybrano opcję Operations->Network + Cluster->Dissimilarity->Network based->d1->All, aby otrzymać dendrogram.

4. Wykreśl dendrogram (1) i zaproponuj cięcie (1)

Na następnej stronie został umieszczony dendrogram uzyskany jako wynik wykonania powyższych operacji. Ze względu na jego rozbudowanie można zawsze próbować stworzyć inny dendrogram bardziej odpowiadający wybranej przez nas skali np. dwa poniższe, gdzie ograniczono szczegółowość dendrogramu.



Rysunek 5 Inny wariant dendrogramu



Rysunek 6 Inny wariant dendrogramu

Przyglądając się dendrogramowi po prawej można zauważyć kilka zależności:

- zdecydowana większość węzłów w grafie jest do siebie całkiem podobna,
- istnieje część węzłów, które znacząco odstają od reszty tzn. mają akurat w tym przypadku zdecydowanie większy rząd od pozostałych węzłów,
-

Bazując na powyższych dendrogramach oraz tym najbardziej szczegółowym przedstawionym na tej stronie można zaproponować cięcie w miejscu oznaczonym czerwoną linią. Jest to chyba najbardziej intuicyjne cięcie pozwalające na podzielenie grafu na część posiadającą węzły o rzędach znacznie wyższych (rzędu kilkudziesięciu – kilkuset) oraz resztę, posiadającą rzędy przeważnie znacznie mniejsze. Szczególną uwagę należy zwrócić na węzeł nr 44, który ma najwyższy możliwy rząd ze wszystkich węzłów w grafie.

Ze względu na mnogość różnych rzędów węzłów w zadanym przykładzie niezwykle ciężko jest wskazać miejsce najbardziej interesującego cięcia.

Inną opcją przy tym poleceniu mogłoby być sugerowanie się metodą podziału Louviana, która przyniosła podzielenie grafu na 6 grup i której efekty możemy zobaczyć poniżej.

Na następnych rysunkach można zaobserwować sieci, które powstały na skutek wykonanego cięcia.

Viewing Hierarchy: 1. Hierarchical Clustering [Ward] of N2 (297)

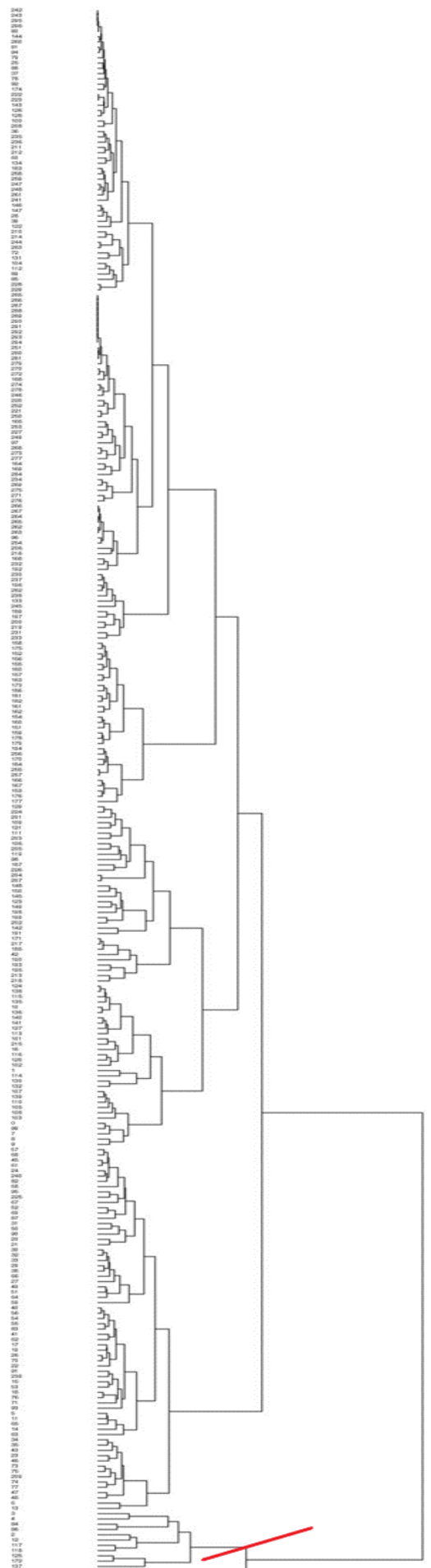
File Edit

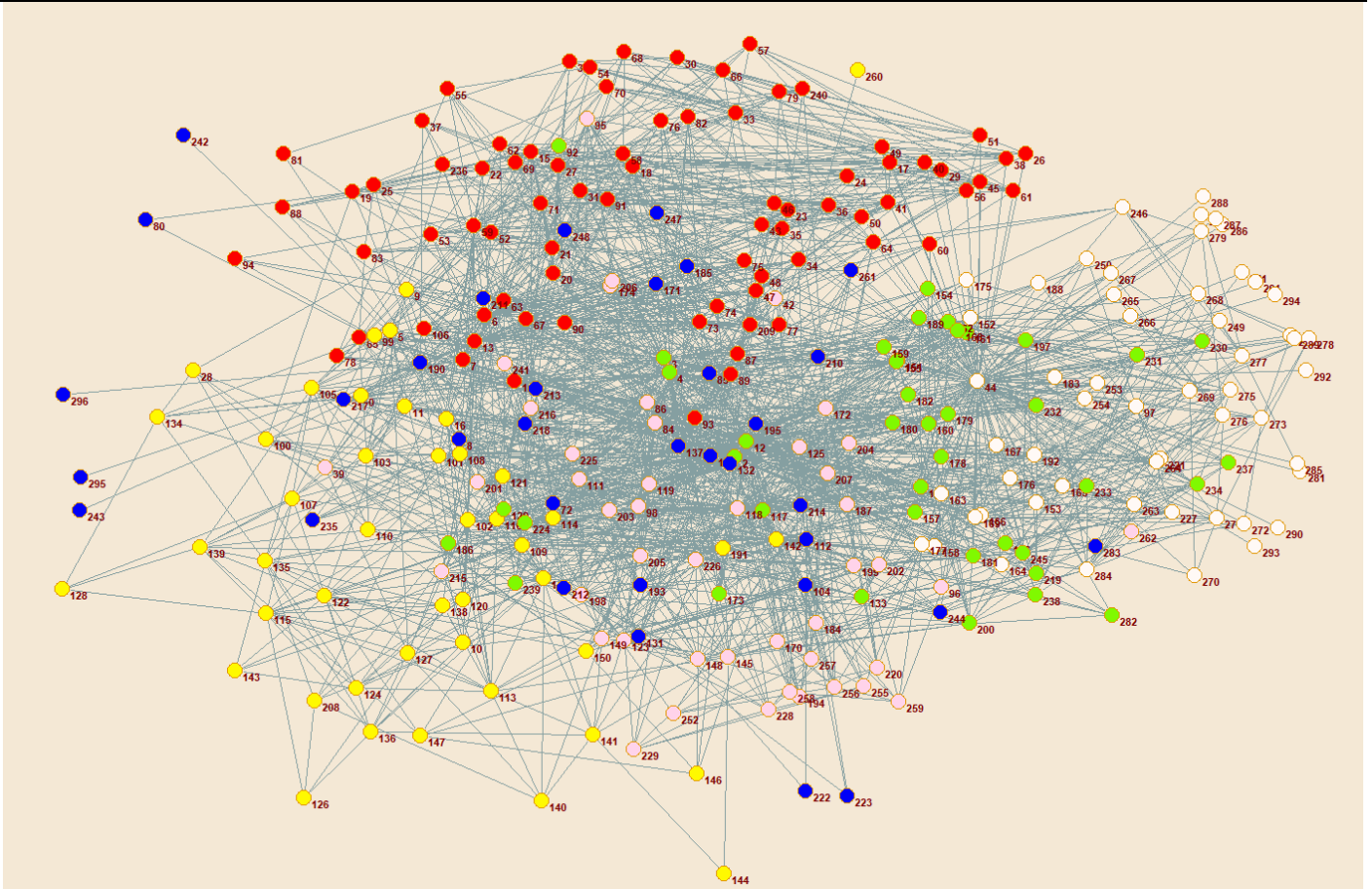
```

✓ Root [4.49] (297)
  > 100295 [2.27] (285)
  > 100294 [2.05] (12)
    > 100290 [1.28] (11) (Cut)
      v45 (1)
  
```

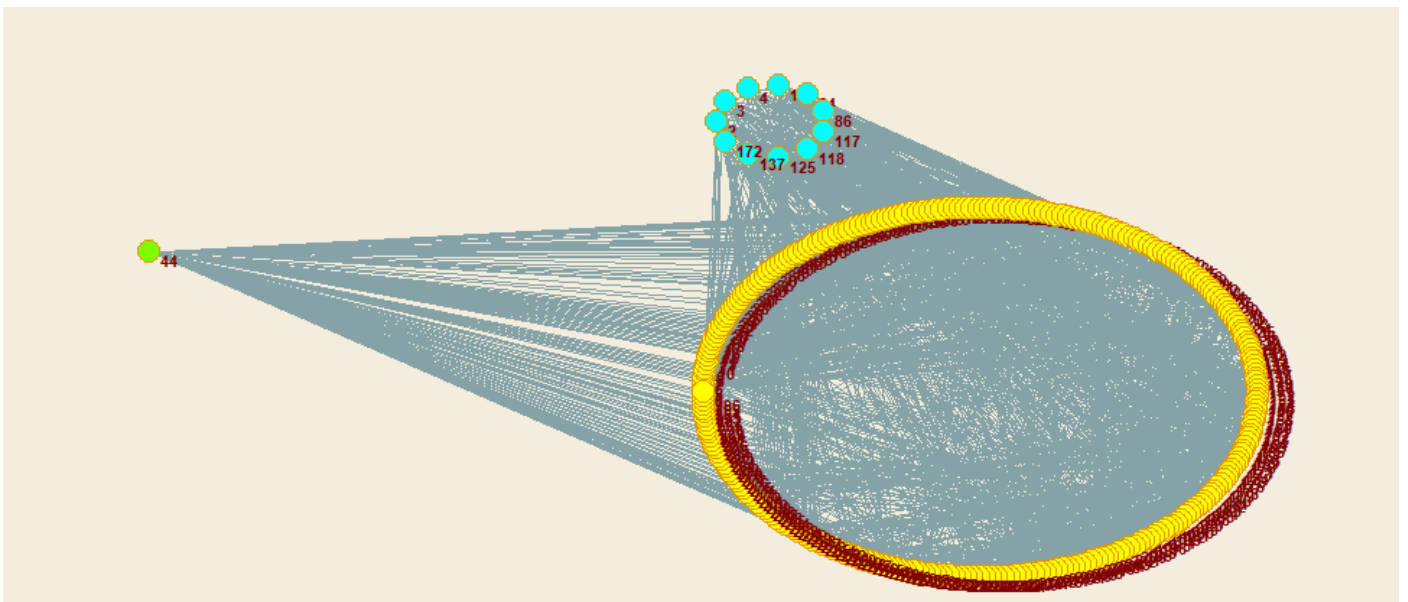
Rysunek 7 Widok drzewa i wykonanych działań

Pajek - Ward [0.00,4.49]

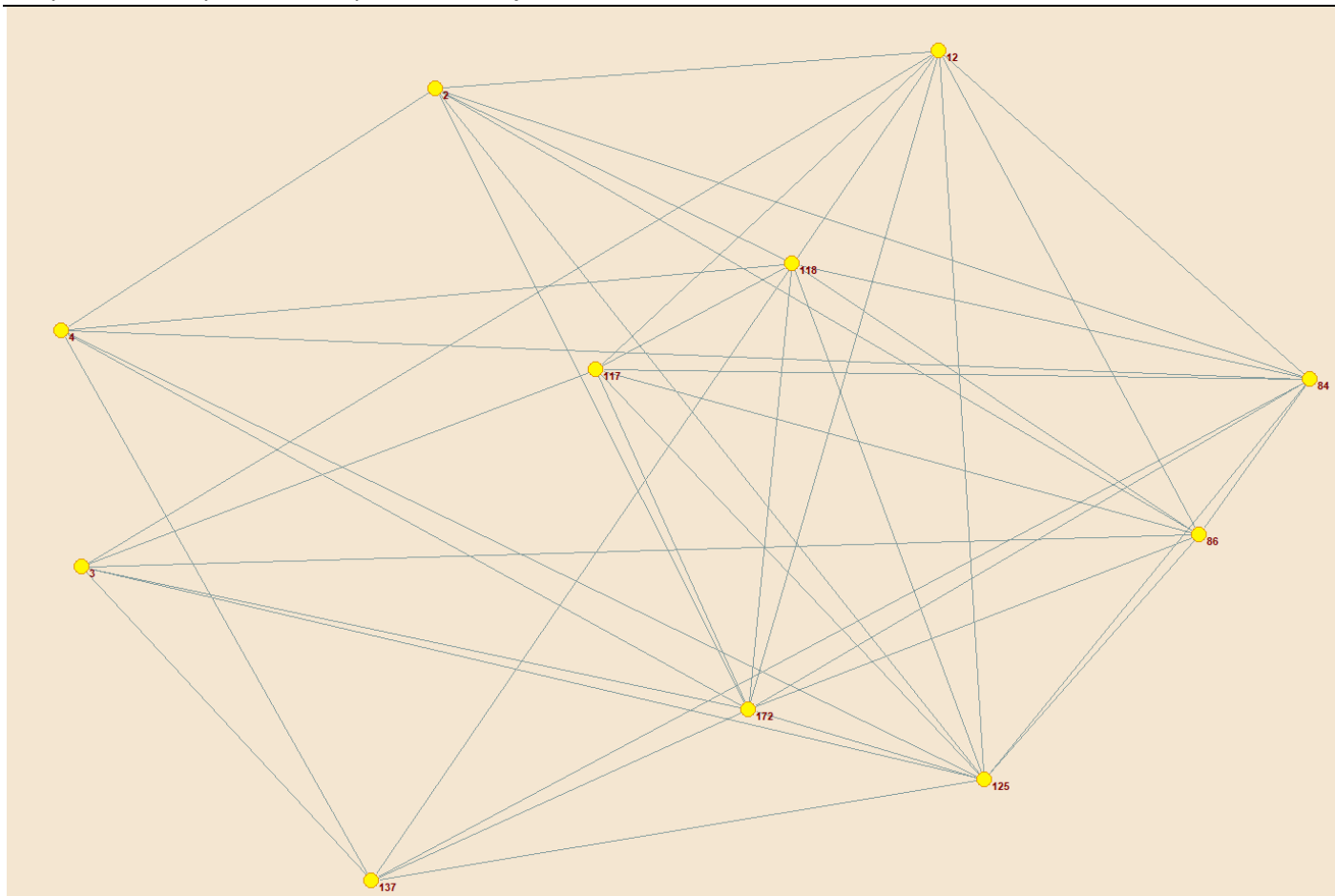




Rysunek 8 Metoda Louvaina



Rysunek 9 Widok poszczególnych grup



Rysunek 10 Sieć niebieska odcięta

Aby otrzymać następujące schematy wykonano operacje Hierarchy -> Create network. Następnie Operations -> Network + Partition -> Extract -> SubNetwork Induced by Union of Selected Clusters. W celu wykonania cięcia otwarto lupę zaprezentowaną poniżej oraz po naciśnięciu na odpowiednią gałąź wykorzystano skrót klawiszowy Ctrl + T i naciskano go do momentu, aż przy danej gałęzi pojawi się napis (Cut). Schematy otrzymano przy użyciu opcji Draw z menu.



Rysunek 11 Menu edycji hierarchii

Zadanie B

Analiza sieci przy pomocy biblioteki NetworkX

Do wykonania polecenia została wykorzystana sieć nr 4, czyli „Sieć zależności artefaktów Mavena”. Wybrano zestaw nr 4 na podstawie obliczeń wykonanych poniżej.

$$269056 \bmod 6 = 38436 + 4$$

Zadania do wykonania:

- zbadaj jaki jest rząd i rozmiar całej sieci: pierwotnej oraz po usunięciu pętli i duplikatów krawędzi (1);
- wyodrębnij największą składową spójną, zbadaj jej rząd i rozmiar (1);
- wyznacz aproksymacje średniej długości ścieżki, operując na próbie losowej 100, 1000 i 10 tys. par wierzchołków (2);
- wyznacz liczbę rdzeni o największym możliwym rzędzie, o drugim możliwie największym rzędzie o trzecim możliwie największym rzędzie; jakie to są rzędy? (3);
- wykreśl rozkład stopni wierzchołków (1);
- wyznacz wykładnik rozkładu potęgowego metodą regresji dla dopełnienia dystrybuanty rozkładu stopni, dla przedziałów rozłożonych logarytmicznie (3);
- wyznacz wykres Hilla (3).

Analizę wykonano w edytorze VS Code przy użyciu Jupyter Notebook.

Ogólne zastrzeżenie: wyniki mogą się różnić w niektórych przypadkach użycia skryptów, gdyż np. przy losowaniu losowane są różne wierzchołki.

1. Zbadaj jaki jest rząd i rozmiar całej sieci: pierwotnej oraz po usunięciu pętli i duplikatów krawędzi (1)

Sieć wczytano przy użyciu następujących skryptów:

```
import networkx as nx
import requests
import matplotlib as plt
import powerlaw

url = 'https://www.ia.pw.edu.pl/~mkamola/dataset-big/4.txt'
r = requests.get(url, allow_redirects=True)
open('4.txt', 'wb').write(r.content)
```

Plik został pobrany z serwera studia.

```
G_multi = nx.read_edgelist("4.txt", create_using=nx.MultiGraph)
G = nx.read_edgelist("4.txt", create_using=nx.Graph)
I1 = nx.info(G_multi)
print(I1 + '\n')
I2 = nx.info(G)
print(I2 + '\n')
```


Wynik:

```
Name: Type: MultiGraph Number of nodes: 6988 Number of edges: 5537 Average degree: 1.5847
Name: Type: Graph Number of nodes: 6988 Number of edges: 5537 Average degree: 1.5847
```

Wczytano dane do dwóch rodzajów grafów, aby określić czy występują zduplikowane krawędzie lub pętle przy węzłach.

```
nx.number_of_selfloops(G_multi)
```

```
256
```

```
nx.number_of_selfloops(G)
```

```
256
```

Czyszczenie z pętli i wielokrotnych krawędzi:

```
G_multi=nx.Graph(G_multi)
G_multi.remove_edges_from(nx.selfloop_edges(G_multi))
G.remove_edges_from(nx.selfloop_edges(G))
```

```
nx.number_of_selfloops(G_multi)
```

```
0
```

```
nx.number_of_selfloops(G)
```

```
0
```

Usuniętych zostało 256 pętli. Finalnie nasze grafy prezentują się następująco:

```
N1 = nx.info(G_multi)
print(N1 + '\n')
N2 = nx.info(G)
print(N2 + '\n')
```

```
Name: Type: Graph Number of nodes: 6988 Number of edges: 5281 Average degree: 1.5114 Name:
Type: Graph Number of nodes: 6988 Number of edges: 5281 Average degree: 1.5114
```

2. Wyodrębnij największą składową spójną, zbadaj jej rząd i rozmiar (1)

Musimy na początku sprawdzić czy nasze grafy są spójne.

```
nx.is_connected(G)
```

```
False
```

```
nx.is_connected(G_multi)
```

```
False
```

Jak widać, nasz pierwotny graf nie jest spójny. Zastosowano więc poniższy skrypt w celu znalezienia największej składowej spójnej.

```
largest_cc = max(nx.connected_components(G_multi), key=len)
G_biggest=G_multi.subgraph(largest_cc)
Inf = nx.info(G_biggest)
print(Inf)
```

Wynik:

```
Name: Type: Graph Number of nodes: 626 Number of edges: 742 Average degree: 2.3706
```

Największa składowa spójna w naszym grafie ma 626 węzłów i 742 krawędzie. Jest ona znacznie mniejsza od pierwotnego grafu, więc możemy zakładać, że sieć posiada wiele różnych grup niepołączonych ze sobą.

3. Wyznacz aproksymację średniej długości ścieżki, operując na próbie losowej 100, 1000 i 10 tys. par wierzchołków (2)

W tym punkcie metodą, którą zastosowałem, było losowanie węzłów oraz pobieranie ich zależności z innymi węzłami, gdyż NetworkX nie pozwalał mi, przynajmniej bezpośrednio, pobierać samych krawędzi.

```
n_set = [200, 310, 620]
random_graph=[None] * len(n_set)
create_graph=[None] * len(n_set)
for i in n_set:
    random_graph = random.sample(G_biggest.nodes, i)
    create_graph = G_multi.subgraph(random_graph)
    lar_cc = max(nx.connected_components(create_graph), key=len)
    G_big=create_graph.subgraph(lar_cc)
    print(nx.info(G_big))
    try:
        ave = nx.average_shortest_path_length(G_big)
        print(i,ave)
    except nx.NetworkXError:
        print('Cannot count length')
```

Wynik:

```
Name: Type: Graph Number of nodes: 32 Number of edges: 34 Average degree: 2.1250 Probes: 200
```

Średnia najkrótsza długość ścieżki dla przypadku nr 1:

```
ASPL: 4.094758064516129
```

```
Name: Type: Graph Number of nodes: 128 Number of edges: 152 Average degree: 2.3750 Probes: 310
```

Średnia najkrótsza długość ścieżki dla przypadku nr 2:

```
ASPL: 4.742864173228346
```

```
Name: Type: Graph Number of nodes: 620 Number of edges: 736 Average degree: 2.3742 Probes: 620
```

Średnia najkrótsza długość ścieżki dla przypadku nr 3:

```
ASPL: 6.34676637656991
```

Dla największej składowej spójnej:

```
nx.average_shortest_path_length(G_biggest)
```

```
6.3478747603833865
```

Średnia najkrótsza długość ścieżki jest o ok. 1/3 mniejsza od maksymalnej w ramach największej składowej spójnej.

4. Wyznacz liczbę rdzeni o największym możliwym rzędzie, o drugim możliwie największym rzędzie o trzecim możliwie największym rzędzie; jakie to są rzędy? (3)

```
from collections import Counter
core_numbers = nx.core_number(G_biggest)

hist_data = sorted(Counter(core_numbers).items())

for i in list(sorted_freqs)[:5]:
    print(str(i) + '-core info')
    print(nx.info(nx.k_core(G_biggest, k=i)))
    print('\n')
```

Zastosowano powyższy skrypt w celu znalezienia najwyższych stopni k-rdzeni przy wykorzystaniu procesu degeneracji grafu, czyli odcinaniu kolejnych węzłów niższych od k.

Wyniki dla największej składowej spójnej:

3-core info

Name:

Type: Graph

Number of nodes: 18

Number of edges: 39

Average degree: 4.3333

2-core info

Name:

Type: Graph

Number of nodes: 149

Number of edges: 265

Average degree: 3.5570

1-core info

Name: Type: Graph

Number of nodes: 626

Number of edges: 742

Average degree: 2.3706

Wyniki dla całego grafu:

3-core info

Name:

Type: Graph

Number of nodes: 26

Number of edges: 54

Average degree: 4.1538

2-core info

Name:

Type: Graph

Number of nodes: 262

Number of edges: 415

Average degree: 3.1679

1-core info

Name:

Type: Graph

Number of nodes: 6811

Number of edges: 5281

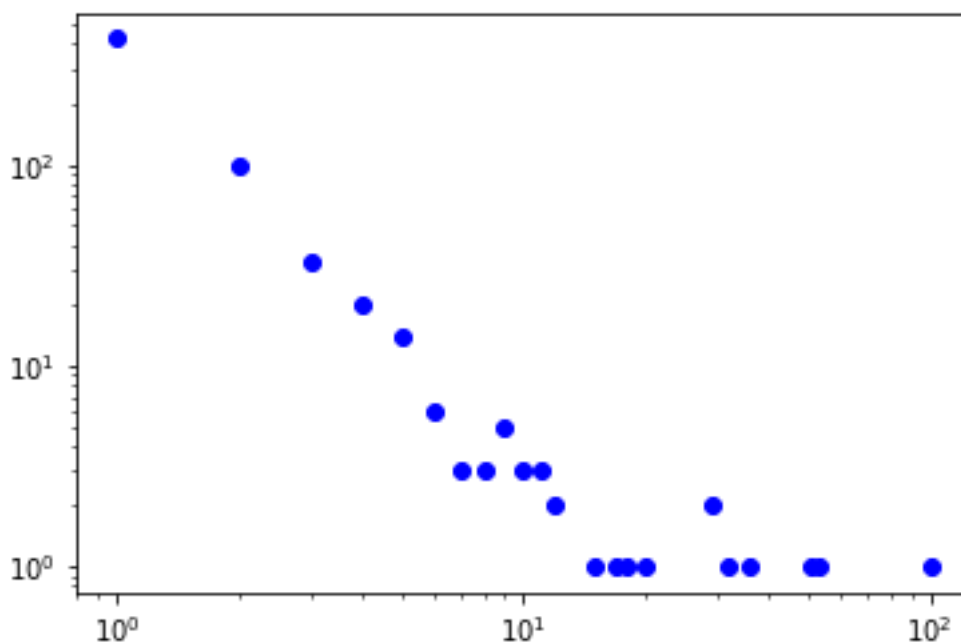
Average degree: 1.5507

Po symulacjach możemy jasno stwierdzić, że najwyższymi stopniami są 3, 2 oraz 1.

5. Wykreśl rozkład stopni wierzchołków

Schemat występowania stopni wierzchołków:

```
deg_dict = Counter(dlist)
degs = deg_dict.keys()
freqs = deg_dict.values()
plt.loglog(degs, freqs, 'bo')
plt.gcf().savefig('degree_freq.png')
plt.close()
```



Rysunek 12 Schemat rozkładu stopni wierzchołków

Rozkład stopni w pewnym momencie spada do wartości 10^0 , z czego wynika duża ilość węzłów niskich rzędów w porównaniu z węzłami wyższych rzędów.

6. Wyznacz wykładnik rozkładu potęgowego metodą regresji dla dopełnienia dystrybucji rozkładu stopni, dla przedziałów rozlokowanych logarytmicznie (3)

W celu otrzymania wartości wykładnika skorzystano z biblioteki SciPy

```
import scipy.stats as st
```

```
dlist = G_biggest.degree()
dlist = [elem[1] for elem in dlist]

dlist = dict(Counter(dlist))

log_dlist = dlist.keys()

print(type(log_dlist))

log_dlist = np.log10(list(dlist.keys()))
log_ylist = np.log10(list(dlist.values()))

st.linregress(log_dlist, log_ylist)
```

Jako wynik otrzymano następującą linijkę:

```
LinregressResult
(slope=-1.3386952271812083,
intercept=2.007638280919855,
rvalue=-0.8974471120557004,
pvalue=1.5118509900124536e-08,
stderr=0.147135219003944)
```

Jako wartość wykładnika należy potraktować wartość slope (nachylenia), czyli 1,3387.

7. Wyznacz wykres Hilla (3)

Wykres Hilla został wygenerowany za pomocą następującego skryptu:

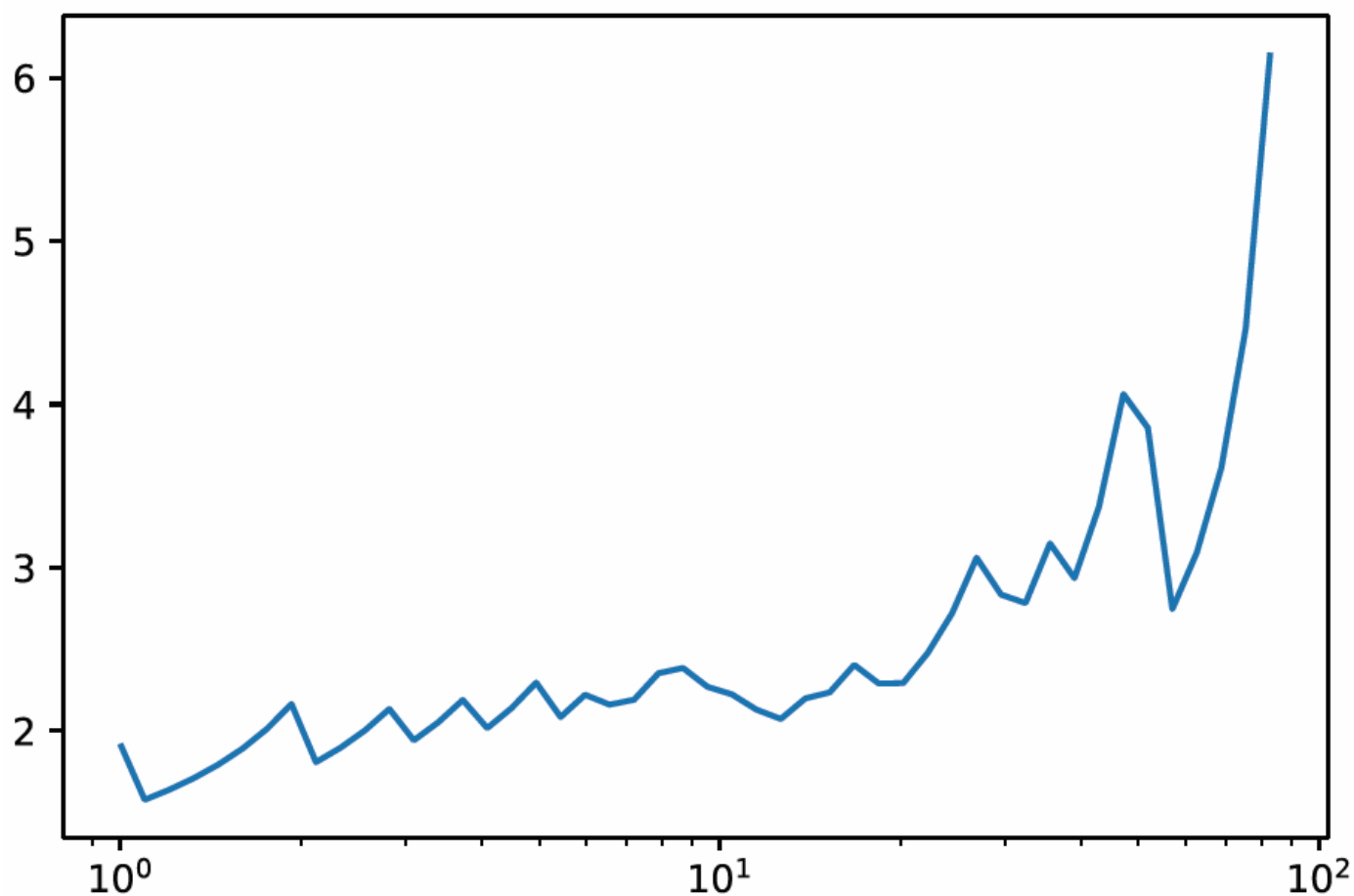
```
import matplotlib.pyplot as plt
dlist = G_biggest.degree()
dlist = [elem[1] for elem in dlist]

NBINS = 50
bins=np.logspace(np.log10(min(dlist)),
                 np.log10(max(dlist)), num=NBINS)
bcnt, bedge=np.histogram(np.array(list(dlist)),bins=bins)
alpha=np.zeros(len(bedge[:-2]))

for i in range(0,len(bedge)-2):
    fit=powerlaw.Fit(list(dlist), xmin=bedge[i], discrete=True)
    alpha[i]=fit.alpha

plt.semilogx(bedge[:-2],alpha)
```

Wykres wygląda następująco:



Rysunek 13 Wykres Hilla