

```
In [7]: import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
import matplotlib.colors as mcolours
# import plotly.graph_objects as go
from numpy import linalg as LA
from collections import defaultdict
import numpy as np
import random
from community import community_louvain
from graphrole import RecursiveFeatureExtractor, RoleExtractor
import multiprocessing as mp
import itertools
from numpy import linalg as LA
from os import cpu_count
import time
from modelos import configuration_model
from metrics import distribution_grados
from motifs_calculos import motif_grafo_electorios
from motifs_graficos import graficar_significant_profile
from motifs_calculos import significant_profile

In [8]: world = pd.read_csv("World.csv")
world = world.drop(['ConexionAeropuertos'], axis=1)
world

Out[8]:
```

1	Papua New Guinea	Philippines
2	Papua New Guinea	Indonesia
3	Papua New Guinea	Solomon Islands

...
2847	Lithuania	Georgia
2848	Armenia	Georgia
2849	Entrea	Yemen
2850	Yemen	Djibouti
2851	Solomon Islands	Nauru

2852 rows x 2 columns

Ejercicio 1.9

```
'''
algoritmo basado al concentrado en:
```

<https://stackoverflow.com/questions/43541376/how-to-draw-communities-with-networkx>

```
def community_layout(g, partition):
    pos_communities = _position_communities(g, partition, scale=3.)
    pos_nodes = _position_nodes(g, partition, scale=1.)
    return dict()
```

```
for node in g.nodes():
    pos[node] = pos_communities[node] + pos_nodes[node]

return pos
```

```

between_community_edges = _find_between_community_edges(partition)

communities = set(partition.values())
hypergraph = nx.DiGraph()
hypergraph.add_nodes_from(communities)
for (ci, cj), edges in between_community_edges.items():
    hypergraph.add_edge(ci, cj, weight=len(edges))

pos_communities = nx.spring_layout(hypergraph, **kwargs)

pos = dict()
for node, community in partition.items():
    pos[node] = pos_communities[community]

return pos

def _find_between_community_edges(g, partition):
    edges = dict()
    for (ni, nj) in g.edges():
        ci = partition[ni]
        cj = partition[nj]

        if ci != cj:
            try:
                edges[(ci, cj)] += [(ni, nj)]
            except KeyError:
                edges[(ci, cj)] = [(ni, nj)]

    return edges

def _position_nodes(g, partition, **kwargs):
    communities = dict()
    for node, community in partition.items():
        try:
            communities[community] += [node]
        except KeyError:
            communities[community] = [node]

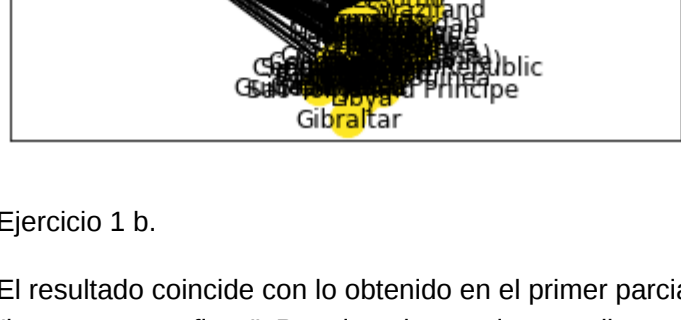

    pos = dict()
    for ci, nodes in communities.items():
        subgraph = g.subgraph(nodes)
        pos_subgraph = nx.spring_layout(subgraph, **kwargs)
        pos.update(pos_subgraph)

    return pos

def graficar_comunidades(g):
    partition = community_louvain.best_partition(g)
    pos = community_layout(g, partition)
    nx.draw_networkx(g, pos, node_color=list(partition.values()), with_labels=True);
    plt.rcParams["figure.figsize"] = (20,8)
    plt.show()
    return

graficar_comunidades(graf1)

```



El algoritmo utilizado es Louvain.

```
comunidades = community_louvain.best_partition(grafos)
grupo_asia_oceania = dict()
```

```
for key, value in count.items():
    if value == 0:
```

```

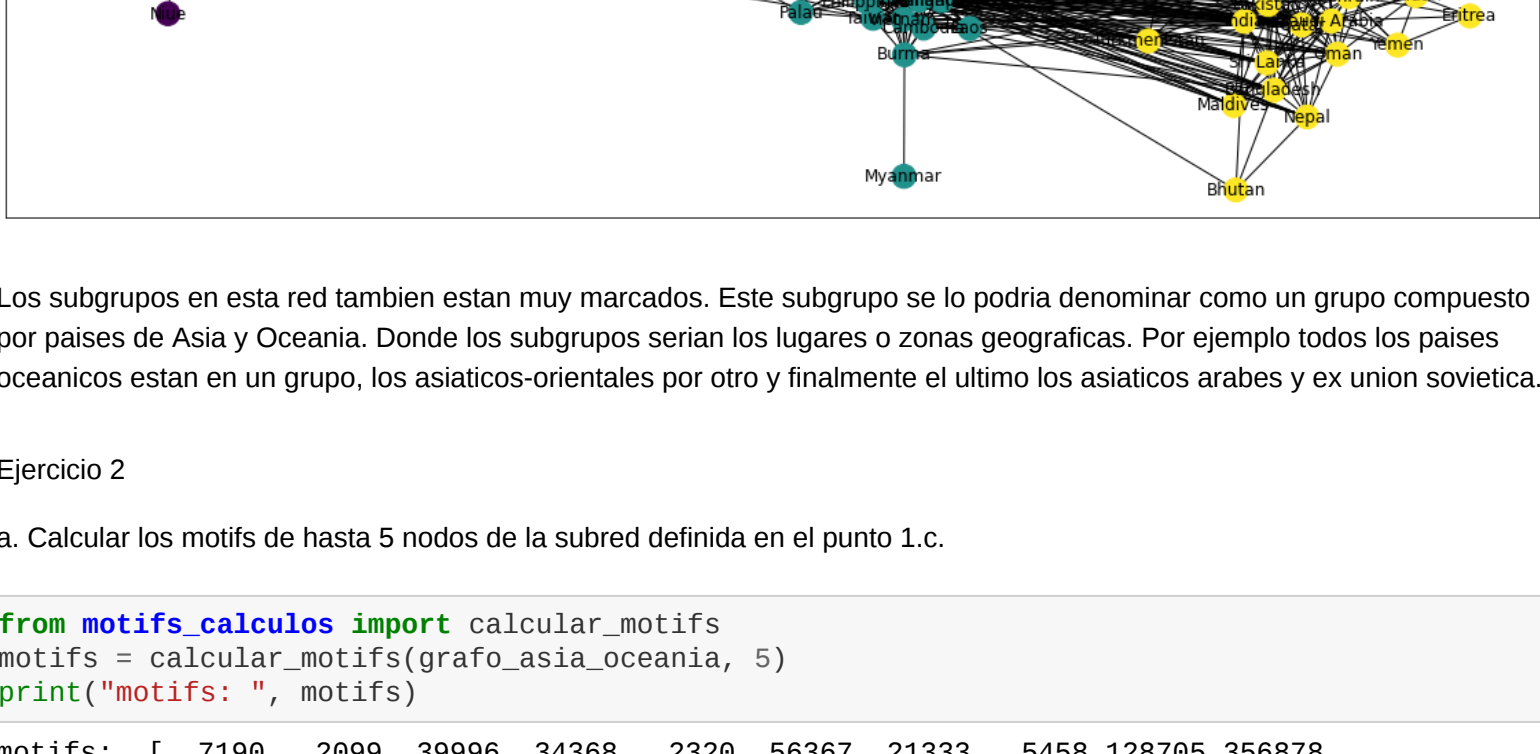
grupo_asia_oceania[key] = value

grafo_asia_oceania = grafo

In [ ]: for node in list(grafo_asia_oceania.nodes()):
        if grupo_asia_oceania.get(node) == None:
            grafo_asia_oceania.remove_node(node)

graficar_comunidades(grafo_asia_oceania)

```



```
1065247 1392068 1064371 312831 56039 5071060 613374 2174361 2300
39991 73231 231422 178720 16627 186573 15283 57637 9945]
```

```
print("p
```

```
promedios: [7.275700e+03 6.766000e+02 5.939560e+04 3.921
```

```

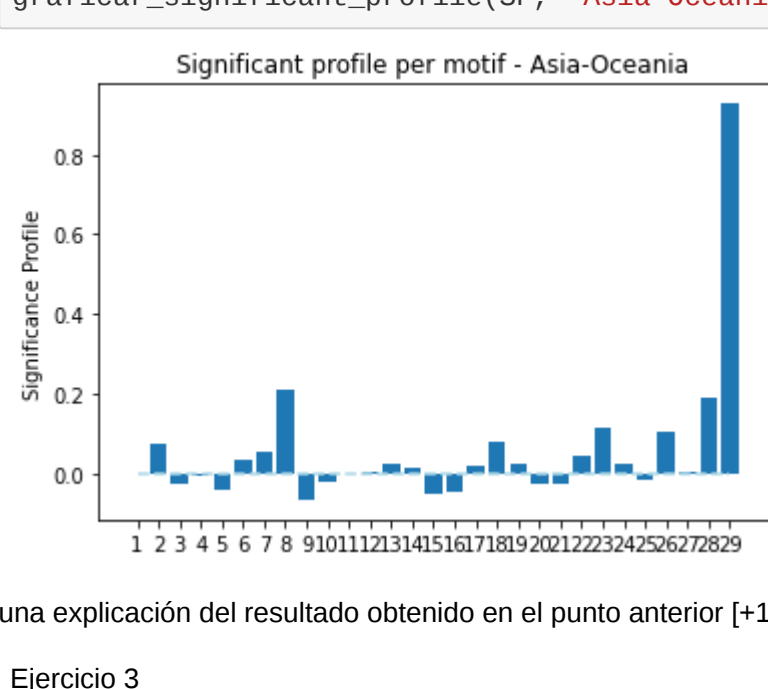
1.250220e+05  2.393289e+05  1.262296e+05  1.999310e+05  4.884290e+04
1.351011e+05  1.963821e+05  1.592446e+04  1.224808e+05  3.992290e+04
1.261411e+05  1.986360e+04  3.129700e+03  7.155230e+04  4.999100e+04
2.659880e+04  1.090700e+04  3.129700e+03  9.510800e+01

```

3 280341136e+04 2 000329835e+03 1 49813249e+04 3 89131437e+03

```
1.26937699e+04 4.26643475e+03 3.63141719e+03 1.37820734e+04
6.01486603e+03 5.04761858e+03 2.59963201e+03 9.50796298e+02
3.52035510e+01]
```

```
print("SP", SP)
```



Detectar los roles en dicha red utilizando el algoritmo RolX, explicando

algoritmo basado al encontrado en:

```

def extract_roles_and_plot(G, title='', save = False, file_name='', big=False):
    feature_extractor = RecursiveFeatureExtractor(G)
    features = feature_extractor.extract_features()
    role_extractor = RoleExtractor(n_roles=None)
    role_extractor.extract_role_factors(features)
    labels = {node: node for node in G.nodes()}

    available_colors = \
        ('role_0': '#E9D758', 'role_1': '#297373', 'role_2': '#FF8552', 'role_3': '#888888', 'role_4':
         '#00aa00', 'role_5': '#aaaa00', 'role_6': '#aa0000', 'role_7': '#0000aa'})

    colors = [available_colors[role_extractor.roles[node]] for node in G.nodes()]

    pos = nx.kamada_kawai_layout(G)
    if big:
        plt.figure(figsize=(25,25))
    else:
        plt.figure(figsize=(20,20))
    plt.title(title)

    nx.draw_networkx_nodes(G, pos, nodelist=G.nodes(), node_color=colors, alpha=0.7, node_size=
e=200, linewidths=2)
    nx.draw_networkx_edges(G, pos, width=0.3, alpha=0.05)
    nx.draw_networkx_labels(G, pos, labels=labels)

    if save:
        plt.savefig(file_name, format = 'svg', dpi=300)
        plt.figure(figsize=(10,10))
        plt.title(title)

    nx.draw_networkx_nodes(G, pos, nodelist=G.nodes(), node_color=colors, alpha=0.7, node_size=
e=200, linewidths=2)
    nx.draw_networkx_edges(G, pos, width=0.3, alpha=0.05)
    nx.draw_networkx_labels(G, pos, labels=labels)

    if save:
        plt.savefig(file_name, format = 'svg', dpi=300)

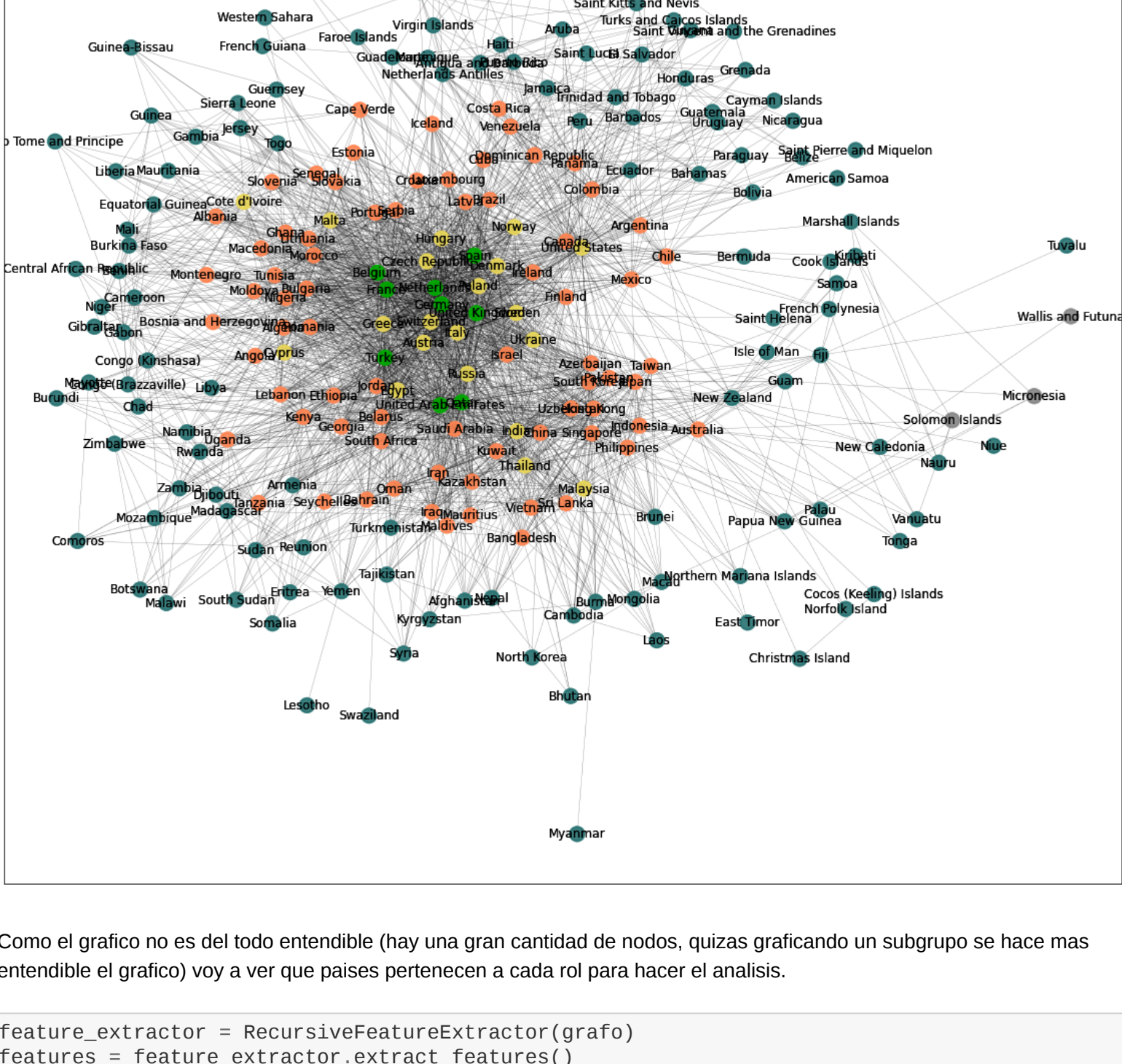
```

```

In [ ]: grafo = nx.from_pandas_edgelist(world, 'Origin', 'Destino')

extract_roles_and_plot(grafo, title='Roles', save=False, file_name='img/roles_ep4.svg')

```



```
role_extractor.extract_role_factors(features)

países_por_rol = [[], [], [], [], [], [], [], []]
```

```
for i in range(0,5):
    for key, value in role_extractor.roles.items():
        if i == int(value[len(value)-1]):
            paises_por_rol[i].append(key)
```

```
for i in range(0,5):
    print('Los países pertenecientes al rol', i, 's')
    for value in paises_por_rol[i]:
        print(value, end=', ')
```

Los países pertenecientes al rol 0 son:

Austria, Côte d'Ivoire, Cyprus, Czech Republic, Denmark, Egypt, Greece, Hungary, India, Italy, Japan, Jordan, Kazakhstan, Korea, Kuwait, Lebanon, Lithuania, Luxembourg, Macedonia, Malaysia, Malta, Norway, Poland, Russia, Sweden, Switzerland, Thailand, Ukraine, United States, Uzbekistan, Venezuela.

Los países pertenecientes al rol 1 son:

Afghanistan, American Samoa, Anguilla, Antigua and Barbuda, Armenia, Aruba, Bahamas, Barbados, Bangladesh, Belgium, Belize, Benin, Bermuda, Bhutan, Bolivia, Botswana, British Virgin Islands, Brunei, Burkina Faso, Burundi, Cambodia, Cameroon, Canada, Cape Verde, Cayman Islands, Central African Republic, Chad, Chile, China, Christmas Island, Cocos (Keeling) Islands, Colombia, Costa Rica, Cote d'Ivoire, Czech Republic, Denmark, Djibouti, Dominica, Dominican Republic, Ecuador, Egypt, El Salvador, Equatorial Guinea, Estonia, Ethiopia, Falkland Islands (Malvinas), Faroe Islands, Finland, France, French Polynesia, Gabon, Gambia, Germany, Ghana, Gibraltar, Greece, Greenland, Grenada, Guadeloupe, Guam, Guatemala, Guinea, Guinea-Bissau, Guyana, Haiti, Honduras, Hungary, Iceland, India, Indonesia, Iraq, Ireland, Israel, Italy, Jamaica, Japan, Jordan, Kazakhstan, Kenya, Korea, Kuwait, Kyrgyzstan, Laos, Latvia, Lebanon, Lesotho, Liberia, Lithuania, Luxembourg, Macedonia, Madagascar, Malawi, Malaysia, Maldives, Mali, Malta, Marshall Islands, Mauritania, Mauritius, Mexico, Micronesia, Moldova, Monaco, Mongolia, Montenegro, Morocco, Mozambique, Myanmar, Namibia, Nepal, Netherlands, New Zealand, Nicaragua, Niger, Nigeria, North Macedonia, Norway, Oman, Pakistan, Panama, Papua New Guinea, Paraguay, Peru, Philippines, Poland, Portugal, Qatar, Romania, Rwanda, Saint Kitts and Nevis, Saint Lucia, Saint Vincent and the Grenadines, Samoa, San Marino, Sao Tome and Principe, Saudi Arabia, Senegal, Serbia, Seychelles, Sierra Leone, Singapore, Slovakia, Slovenia, South Africa, South Korea, South Sudan, Spain, Sri Lanka, Sudan, Suriname, Swaziland, Sweden, Switzerland, Taiwan, Tajikistan, Tanzania, Thailand, Timor-Leste, Togo, Tonga, Trinidad and Tobago, Tunisia, Turkey, Turkmenistan, Turks and Caicos Islands, Uganda, Ukraine, United Arab Emirates, United Kingdom, United States, Uruguay, Uzbekistan, Vanuatu, Venezuela, Viet Nam, Virgin Islands, Yemen, Zambia, Zimbabwe.

Christmas Island, Cocos(Keeling) Islands, Comoros, Congo (Brazzaville), Congo (Kinshasa), Cook Islands, Djibouti, Dominica, East Timor, Ecuador, El Salvador, Equatorial Guinea, Eritrea, Ethiopia, Falkland Islands, Faroe Islands, FI, French Guiana, French Polynesia, Gabon, Gambia, Ghana, Guinea, Guinea-Bissau, Guyana, Honduras, Hungary, Iceland, India, Indonesia, Iran, Iraq, Israel, Italy, Jamaica, Japan, Jersey, Jordan, Kazakhstan, Kenya, Kiribati, Kyrgyzstan, Laos, Latvia, Lebanon, Lesotho, Liberia, Lithuania, Luxembourg, Macao, Madagascar, Malawi, Mali, Marshall Islands, Martinique, Mauritania, Mauritius, Mexico, Micronesia, Moldova, Monaco, Mongolia, Montserrat, Mozambique, Myanmar, Namibia, Nauru, Nepal, Netherlands Antilles, Netherlands, New Caledonia, New Zealand, Nicaragua, Niger, Niue, Norfolk Island, North Korea, Northern Mariana Islands, Norway, Oman, Pakistan, Palau, Papua New Guinea, Paraguay, Peru, Puerto Rico, Reunion, Rwanda, Saudi Arabia, Senegal, Serbia, Seychelles, Sierra Leone, Singapore, Slovakia, Slovenia, South Africa, South Korea, South Sudan, Spain, Sri Lanka, St. Kitts and Nevis, St. Lucia, St. Vincent and the Grenadines, Sudan, Suriname, Swaziland, Sweden, Switzerland, Taiwan, Tajikistan, Tanzania, Thailand, Timor-Leste, Tonga, Trinidad and Tobago, Tunisia, Turkey, Turkmenistan, Tuvalu, Uganda, Ukraine, United Kingdom, United States, Uruguay, Uzbekistan, Vanuatu, Venezuela, Viet Nam, Virgin Islands, Wallis and Futuna, Yemen, Zambia, Zimbabwe.

he Grenadines, Samoa, Sao Tome and Principe, Sierra Leone, Somalia, South Sudan, Sudan, Suriname, name, Swaziland, Syria, Tajikistan, Tonga, Tonga, Trinidad and Tobago, Turkmenistan, Turks and Caicos Islands, Tuvalu, Uruguay, Vanuatu, Virgin Islands, Western Sahara, Yemen, Zambia, Zimbabue,

Los países pertenecientes al rol 2 son:

Azerbaijan, Bahrain, Bangladesh, Belarus, Bosnia and Herzegovina, Brazil, Bulgaria, Canada, Croatia, Cuba, Dominican Republic, Estonia, Ethiopia, Finland, China, Colombia, Costa Rica, Cape Verde, Chile, China, Georgia, Ghana, Hong Kong

Latvia, Lebanon, Lithuania, Luxembourg, Macedonia, Maldives, Mauritius, Mexico, Moldova, Montenegro, Morocco, Nigeria, Oman, Pakistan, Panama, Philippines, Portugal, Romania, Saudi Arabia, Senegal, Serbia, Seychelles, Singapore, Slovakia, Slovenia, South Africa, South Korea, Sri Lanka, Taiwan, Tanzania, Tunisia, Uganda, Uzbekistan, Venezuela, Vietnam,

Los países pertenecientes al rol 3 son:

Micronesia, Solomon Islands, Wallis and Futuna,

Los países pertenecientes al rol 4 son:

Combinando la información recién impresa y el gráfico podemos determinar que los países pertenecientes a los roles 1 y 3 son los nodos periféricos. Lo cual tiene sentido ya que son los países con menor conexión de vuelos. Por otra parte, los nodos del rol 4 (color verde claro) son los nodos centrales. Estos países son de los mas desarrollados, con mejor ubicación y lugares estratégicos claves para la conexión de muchos vuelos a todos los países del mundo.