Parcialito 2

Teoría de Algoritmos 2

Guido Botta

Padrón 102103

Enunciado

Teoría de Algoritmos II (75.30)

2.do Parcialito Domiciliario – 29/04/2022 - Fecha de Entrega: 20/05/2022

Aclaraciones: Cada ejercicio dice al final del mismo la cantidad de puntos que otorga por hacerse completamente bien (en total, 10). Se deben obtener al menos 5 puntos para aprobar, y se deben aprobar al menos 3 de los parcialitos para aprobar/regularizar la cursada. Para la fecha de entregar, enviar un mail a mbuchwald@fi.uba.ar con un pdf con la resolución, con nombre P2 - PADRON.pdf. Pueden incluir todo el material adicional que les parezca relevante (desde código hasta gráficos).

Nuevamente, considerando esta red que representa las conexiones de diferentes países por los vuelos (directos) realizados en- tre ellos, responder las siguientes preguntas. A los fines de estos ejercicios, se puede obviar la última columna del archivo csv.

1. a. Obtener una visualización de las comunidades presentes en dicha red (indicando el algoritmo utilizado).

   b. Considerando lo que respondiste en el parcialito 1 (ejercicio 2):

   i. Si mencionaste que había homofilia, ¿corresponde por el mismo tipo que mencionaste anteriormente? ¿porqué?

   ii. Si mencionaste que no había homofilia (o bien no realizaste el ejercicio), ¿qué tipo de homofilia se puede ver presente?

   c. Obtener los nodos correspondientes a una de las subredes (con al menos 20% de los nodos), y realizar una visualización de las sub-comunidades presentes.

[3 puntos]

1. a. Calcular los motifs de hasta 5 nodos de la subred definida en el punto 1.c.

   b. Calcular el promedio y desvío estandar de los motifs de una red de baseline. Calcular el significant profile de la red, y hacer un gráfico.

c. Intentar dar con una explicación del resultado obtenido en el punto anterior [+1 punto].

[4 puntos] (sin contar 2.c)

1. Detectar los roles en dicha red utilizando el algoritmo RolX, explicando el resultado obtenido.

[3 puntos]

Armado de Datos

In [10]:
```
# IMPORTS
import networkx as nx
import networkx.algorithms.community as nx_comm
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import socialnetworksutils.metricas as snu_metrics
import socialnetworksutils.modelos as snu_models
import socialnetworksutils.homofilia as snu_homofilia
import socialnetworksutils.embeddings as snu_embeddings
from socialnetworksutils.motifs import calculos as snu_motifs
from socialnetworksutils.motifs import graficos as snu_graficos
from pprint import pprint
from graphrole import RecursiveFeatureExtractor, RoleExtractor
```

In [2]:
```
df = pd.read_csv("World.csv")
df = df[['Origen', 'Destino']]

grafo = nx.from_pandas_edgelist(df, source="Origen", target="Destino")
print(f"Nodos: {len(grafo)}")
print(f"Aristas: {len(grafo.edges)}")
```

```
Nodos: 229
Aristas: 2852
```

Resolución

Ejercicio 1

1. a. Obtener una visualización de las comunidades presentes en dicha red (indicando el algoritmo utilizado).

   b. Considerando lo que respondiste en el parcialito 1 (ejercicio 2):

   i. Si mencionaste que había homofilia, ¿corresponde por el mismo tipo que mencionaste anteriormente? ¿porqué?

   ii. Si mencionaste que no había homofilia (o bien no realizaste el ejercicio), ¿qué tipo de homofilia se puede ver presente?

c. Obtener los nodos correspondientes a una de las subredes (con al menos 20% de los nodos), y realizar una visualización de las sub-comunidades presentes.

Comunidades

Las comunidades son nodos bien conectados entre ellos. Están basadas en la cercanía, proximidad o accesibilidad entre los nodos.

Una comunidad es un conjunto de vértices con muchas aristas hacia los vértices del conjunto, y pocas hacia afuera del mismo.

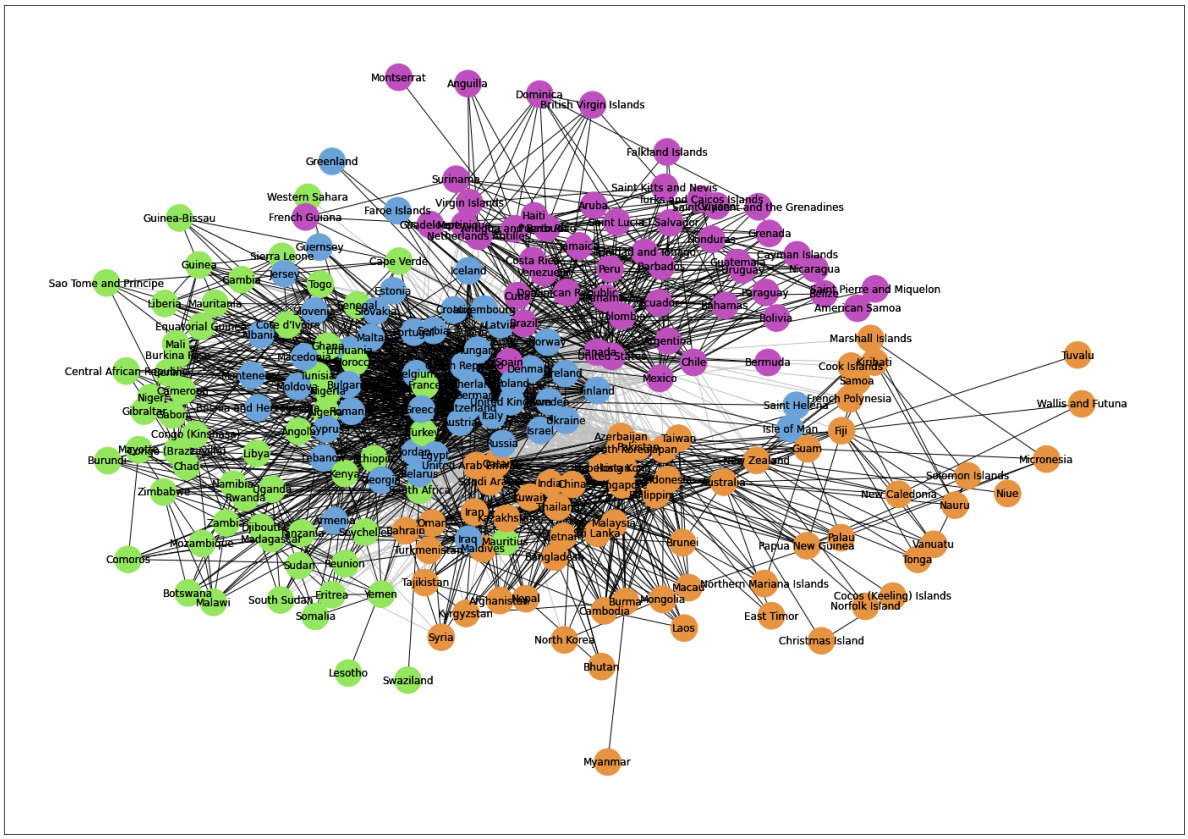
Visualización por Algoritmo de Louvain

Se utilizará el algoritmo de Louvain para encontrar comunidades.

In [39]:
```
# Fuente del código para graficar comunidades: https://graphsandnetworks.com/commu

def set_node_community(G, communities):
    '''Add community to node attributes'''
    for c, v_c in enumerate(communities):
        for v in v_c:
            # Add 1 to save 0 for external edges
            G.nodes[v]['community'] = c + 1

def set_edge_community(G):
    '''Find internal edges and add their community to their attributes'''
    for v, w, in G.edges:
        if G.nodes[v]['community'] == G.nodes[w]['community']:
            # Internal edge, mark with community
            G.edges[v, w]['community'] = G.nodes[v]['community']
        else:
            # External edge, mark as 0
            G.edges[v, w]['community'] = 0

def get_color(i, r_off=1, g_off=1, b_off=1):
    '''Assign a color to a vertex.'''
    r0, g0, b0 = 0, 0, 0
    n = 16
    low, high = 0.1, 0.9
    span = high - low
    r = low + span * (((i + r_off) * 3) % n) / (n - 1)
    g = low + span * (((i + g_off) * 5) % n) / (n - 1)
    b = low + span * (((i + b_off) * 7) % n) / (n - 1)
    return (r, g, b)

print("Busco comunidades con Louvain")
# Uso seed porque a veces da 4 comunidades y a veces 5, para mantener resultado
seed = 20000
comunidades = nx_comm.louvain_communities(grafo, seed=seed)
print(f"Hay {len(comunidades)} comunidades")

# Set node and edge communities
set_node_community(grafo, comunidades)
set_edge_community(grafo)
node_color = [get_color(grafo.nodes[v]['community']) for v in grafo.nodes]

# Set community color for edges between members of the same community (internal) a
external = [(v, w) for v, w in grafo.edges if grafo.edges[v, w]['community'] == 0]
internal = [(v, w) for v, w in grafo.edges if grafo.edges[v, w]['community'] > 0]
```

```
internal_color = ['black' for e in internal]

g_pos = nx.kamada_kawai_layout(grafo)
plt.rcParams.update({'figure.figsize': (25, 18)})
nx.draw_networkx(
    grafo,
    pos=g_pos,
    node_size=0,
    edgelist=external,
    edge_color="silver"
)
nx.draw_networkx(
    grafo,
    pos=g_pos,
    node_color=node_color,
    node_size=1000,
    edgelist=internal,
    edge_color=internal_color
)
```

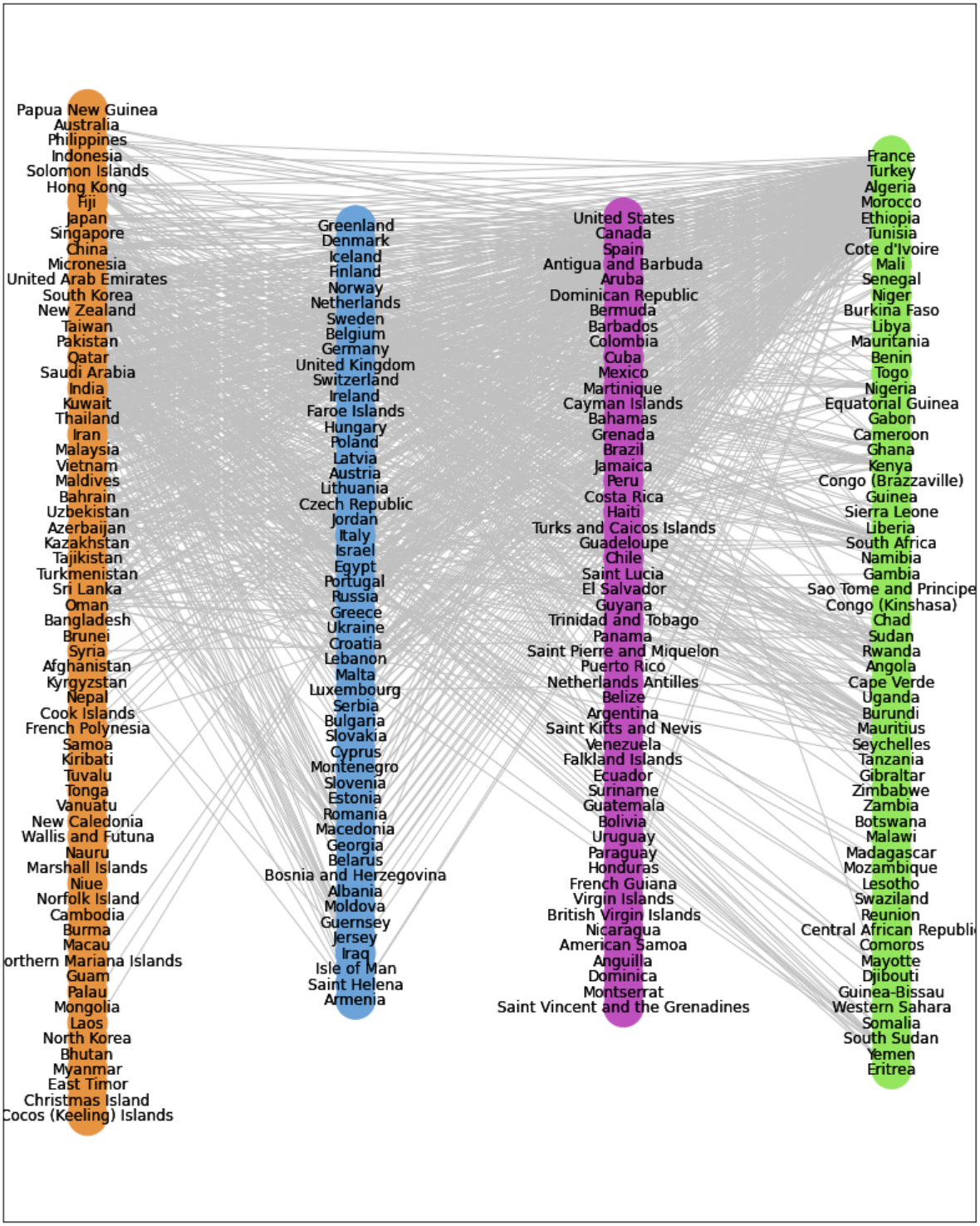
```
Busco comunidades con Louvain
Hay 4 comunidades
```



In [4]:

```
g_pos = nx.multipartite_layout(grafo, 'community')
plt.rcParams.update({'figure.figsize': (14, 18)})
nx.draw_networkx(
    grafo,
    pos=g_pos,
    node_size=0,
    edgelist=external,
    edge_color="silver"
)
nx.draw_networkx(
    grafo,
    pos=g_pos,
    node_color=node_color,
    node_size=1000,
    edgelist=internal,
```

```
    edge_color=internal_color
)
```



Comparación con homofilia del parcialito 1

En el parcialito 1 se habían planteado análisis para dos posibles tipos de homofilia:

- Por PBI (mayor y menor a la mediana en 2010)
- Por Continente

El análisis por continente había dado como resultado una gran posibilidad de homofilia. Analizando el resultado de ejecutar el Algoritmo de Louvain, se pueden observar comunidades divididas por continente.

Se puede observar:

- En el grupo naranja una mayoría de países de Oceanía y Asia.
- En el grupo azul una mayoría de países de Europa.
- En el grupo violeta una mayoría de países de America.
- En el grupo verde una mayoría de países de África.

Si bien existen algunas excepciones, como Francia en África, Groenlandia en Europa y posiblemente algunos otros, la mayoría de comunidades parecen estar bien definidas por su continente. Además, los casos de Francia y Groenlandia tienen sentido. Por el lado de Francia, tiene una gran conexión con países africanos, dado que la mayoría estuvieron bajo dominio Francés. Y por el lado de Groenlandia, forma parte del Reino de Dinamarca, por lo que tiene sentido su conexión con Europa.

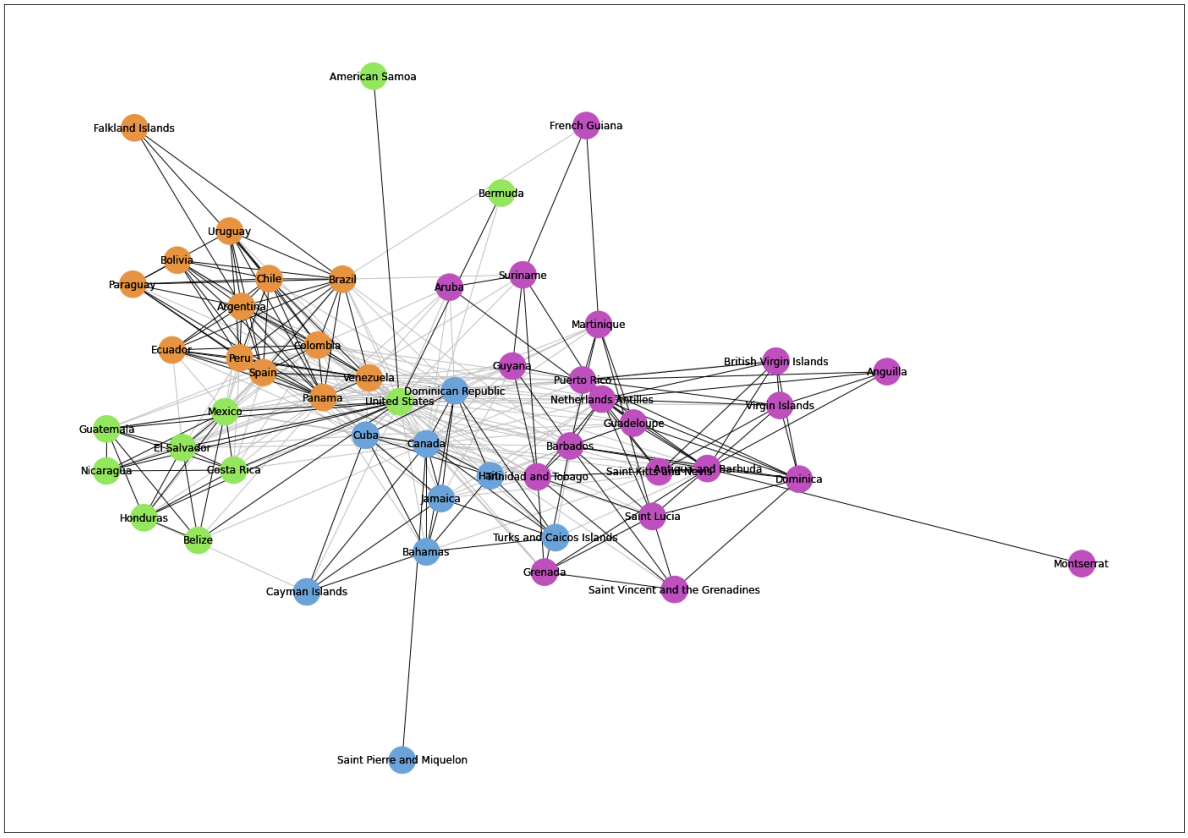
En conclusión, se puede observar un tipo de homofilia por continente.

Visualización de las sub-comunidades presentes en subred

In [41]:
```
print("-- Comunidad de America --")
print(f"Cantidad de nodos: {len(comunidades[2])}\nProporcion: {(len(comunidades[2])

com_america = grafo.subgraph(list(comunidades[2]))

# Fuente: https://graphsandnetworks.com/community-detection-using-networkx/

def set_node_community(G, communities):
    '''Add community to node attributes'''
    for c, v_c in enumerate(communities):
        for v in v_c:
            # Add 1 to save 0 for external edges
            G.nodes[v]['community'] = c + 1

def set_edge_community(G):
    '''Find internal edges and add their community to their attributes'''
    for v, w, in G.edges:
        if G.nodes[v]['community'] == G.nodes[w]['community']:
            # Internal edge, mark with community
            G.edges[v, w]['community'] = G.nodes[v]['community']
        else:
            # External edge, mark as 0
            G.edges[v, w]['community'] = 0

def get_color(i, r_off=1, g_off=1, b_off=1):
    '''Assign a color to a vertex.'''
    r0, g0, b0 = 0, 0, 0
    n = 16
    low, high = 0.1, 0.9
    span = high - low
    r = low + span * (((i + r_off) * 3) % n) / (n - 1)
    g = low + span * (((i + g_off) * 5) % n) / (n - 1)
    b = low + span * (((i + b_off) * 7) % n) / (n - 1)
    return (r, g, b)

print()
print("Busco subcomunidades con Louvain")
# Uso seed porque a veces da 4 comunidades y a veces 5, para mantener resultado
seed = 20000
sub_comunidades = nx_comm.louvain_communities(com_america, seed=seed)
print(f"Hay {len(sub_comunidades)} subcomunidades")

# Set node and edge communities
```

```
set_node_community(com_america, sub_comunidades)
set_edge_community(com_america)
node_color = [get_color(com_america.nodes[v]['community']) for v in com_america.no

# Set community color for edges between members of the same community (internal) a
external = [(v, w) for v, w in com_america.edges if com_america.edges[v, w]['commu
internal = [(v, w) for v, w in com_america.edges if com_america.edges[v, w]['commu
internal_color = ['black' for e in internal]

g_pos = nx.spring_layout(com_america)
plt.rcParams.update({'figure.figsize': (25, 18)})
nx.draw_networkx(
    com_america,
    pos=g_pos,
    node_size=0,
    edgelist=external,
    edge_color="silver"
)
nx.draw_networkx(
    com_america,
    pos=g_pos,
    node_color=node_color,
    node_size=1000,
    edgelist=internal,
    edge_color=internal_color
)
```

```
-- Comunidad de America --
Cantidad de nodos: 52
Proporcion: 22.71%

Busco subcomunidades con Louvain
Hay 4 subcomunidades
```



Se puede observar una división de comunidades por región:

- La región naranja representa a Sudamérica, con Argentina, Chile, Brasil, Ecuador, Paraguay, entre otros. También se encuentran Panamá, país que limita con el norte de

Colombia, y el infiltrado España.

- La región verde representa a Centroamérica, con Honduras, Nicaruagua, México, entre otros. También se encuentra Estados Unidos que, si bien tendría más sentido en la región azul, limita con México.
- La región azul representa a Norteamérica y a algunas islas del Caribe como Cuba, Jamaica, Bahamas, entre otras con mucha influencia norteamericana.
- La región violeta representa a las islas del Caribe y a algunos países del norte de Sudamérica como Guyana o Surinam.

Ejercicio 2

1. a. Calcular los motifs de hasta 5 nodos de la subred definida en el punto 1.c.

   b. Calcular el promedio y desvío estandar de los motifs de una red de baseline. Calcular el significant profile de la red, y hacer un gráfico.

   c. Intentar dar con una explicación del resultado obtenido en el punto anterior [+1 punto].

Motif

Un motif es un patrón recurrente y significante de interconexiones. Siendo un patrón un subgrafo inducido o no inducido, recurrente que fue encontrado con alta frecuencia y significantes que fue encontrado más veces de lo esperado.

Cálculo de motif de hasta 5 nodos de la subred de America

In [6]:
```
MAX_NODOS = 5
motifs = snu_motifs.calcular_motifs(com_america, MAX_NODOS)
print(motifs)
```

```
[  3925    949  14476  23068   1180  24525   7397   1674  35714 108701
 107732  64950  53814 207513   1983  22990 161393  25871  53569   1627
  14622  25325  68560  54892   5852  50265   3156  12129   2024]
```

Promedio y desvío estándar de los motifs de una red de baseline

In [7]:
```
ITERS=5

dist = snu_metrics.distribucion_grados(com_america)
GENERADOR = lambda: snu_models.configuration_model(dist)
proms, std_devs = snu_motifs.motif_grafo_eleatorios(GENERADOR, MAX_NODOS, ITERS)
print(f"Promedios: {proms}\nDesvíos estándar: {std_devs}")
```

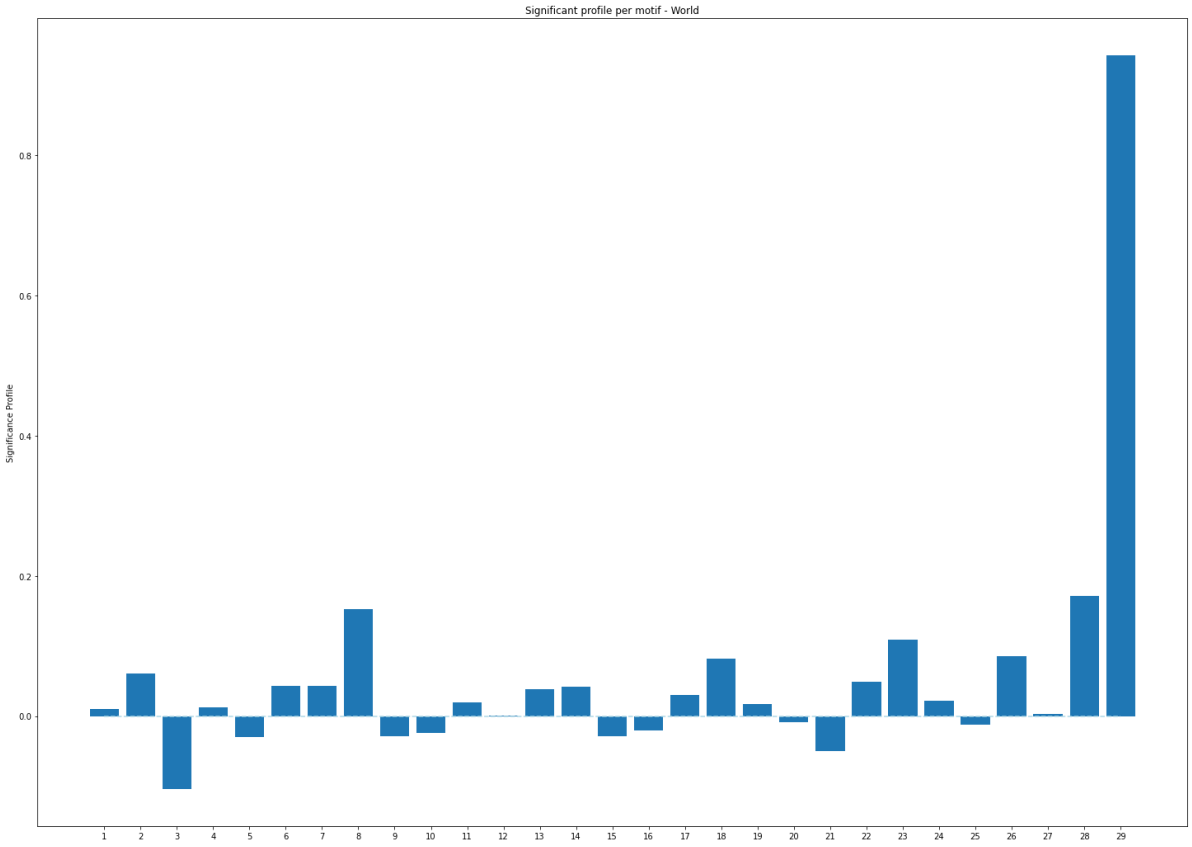
```
Iteracion 1
Iteracion 2; anterior: 234.68 segs
Iteracion 3; anterior: 236.63 segs
Iteracion 4; anterior: 239.54 segs
Iteracion 5; anterior: 210.91 segs
Promedios: [3.561000e+03 2.996000e+02 2.271560e+04 1.564180e+04 4.250400e+03
 1.057860e+04 2.152800e+03 9.620000e+01 9.688680e+04 1.761600e+05
 4.223420e+04 6.408260e+04 3.563500e+04 6.364240e+04 1.761080e+04
 1.025498e+05 5.791180e+04 4.996000e+03 3.030520e+04 1.204900e+04
 4.446320e+04 4.892000e+03 5.850800e+03 2.043920e+04 1.265380e+04
 4.247000e+03 2.531400e+03 4.742000e+02 7.400000e+00]
Desvíos estándar: [7.80256368e+01 2.18778427e+01 1.60806219e+02 1.18057738e+03
 2.07870729e+02 6.64777436e+02 2.46034469e+02 2.10371101e+01
 4.37050919e+03 5.75178094e+03 6.82762865e+03 2.98381692e+03
 9.56800711e+02 6.97096431e+03 1.12649055e+03 7.72619638e+03
 7.00184554e+03 5.20118833e+02 2.69853497e+03 2.43589417e+03
 1.22119178e+03 8.63028852e+02 1.17093388e+03 3.18247988e+03
 1.10245860e+03 1.09504648e+03 4.57191688e+02 1.38792507e+02
 4.36348485e+00]
```

Significant profile

In [8]:
```
sign_prof = snu_motifs.significance_profile(motifs, proms, std_devs)
print(f"Significant Profile: {sign_prof}")
```

```
Significant Profile: [ 9.51805870e-03  6.05588782e-02 -1.04541920e-01  1.28339900e
-02
 -3.01362640e-02  4.28030855e-02  4.34881976e-02  1.53015389e-01
 -2.85572317e-02 -2.39291553e-02  1.95725152e-02  5.93112370e-04
  3.87648341e-02  4.21084268e-02 -2.83047804e-02 -2.10095992e-02
  3.01535710e-02  8.18866215e-02  1.75890477e-02 -8.72935907e-03
 -4.98565174e-02  4.83054575e-02  1.09266844e-01  2.20876038e-02
 -1.25878355e-02  8.57402354e-02  2.78735694e-03  1.71326901e-01
  9.42706835e-01]
```

In [11]:
```
snu_graficos.graficar_significant_profile(sign_prof, 'World')
```



Explicación de Resultado

El motif 29 es un subgrafo de 5 nodos completo. Se podría entender como que la red está tan conectada que en la mayoría de los casos si se toman subgrafos de 5 nodos en la red están completamente enlazados entre sí.

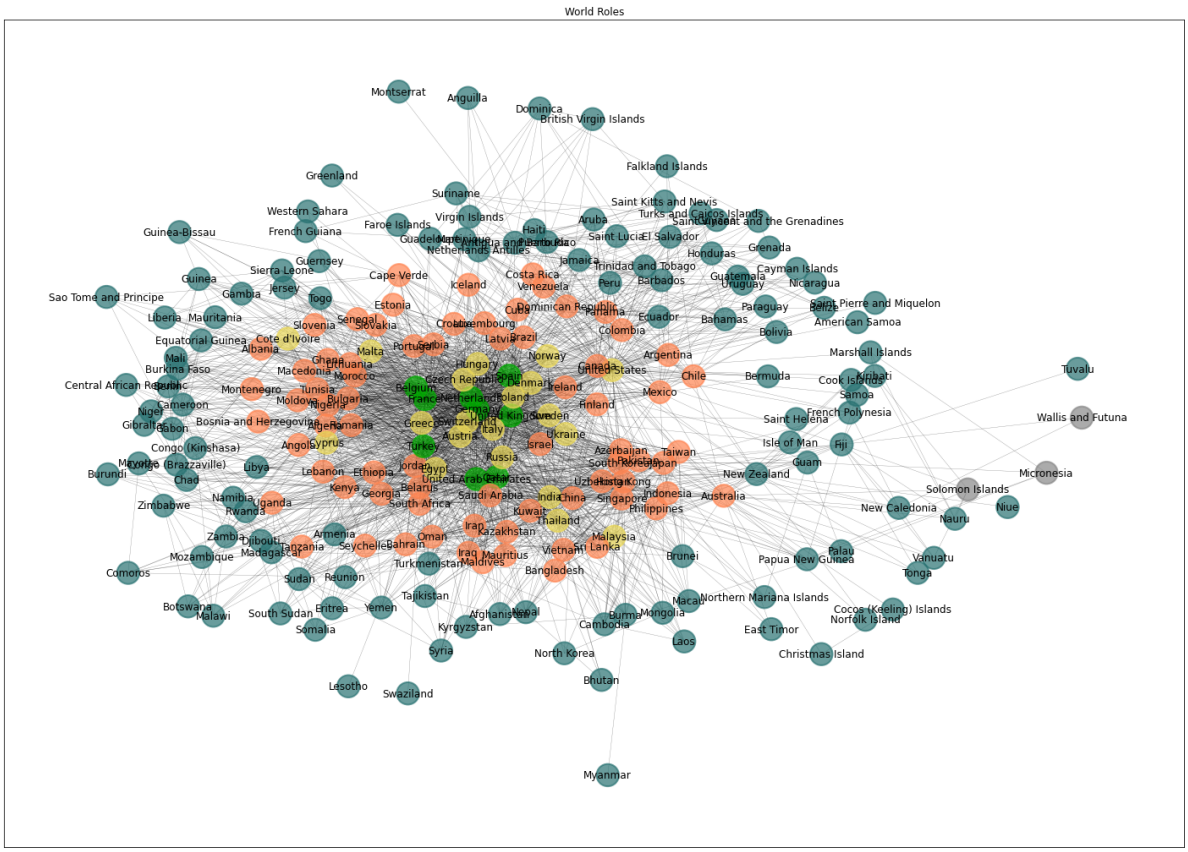
Ejercicio 3

1. Detectar los roles en dicha red utilizando el algoritmo RolX, explicando el resultado obtenido.

Roles

Un rol es un conjunto de nodos que tienen un vecindario/posición similar dentro de la red.

In [24]:
```
feature_extractor = RecursiveFeatureExtractor(grafo)
features = feature_extractor.extract_features()

role_extractor = RoleExtractor(n_roles=None)
role_extractor.extract_role_factors(features)

available_colors = {'role_0':'#E9D758', 'role_1':'#297373', 'role_2':'#ff8552', 'r

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

pos = nx.kamada_kawai_layout(grafo)
plt.figure(figsize=(25,18))
plt.title("World Roles")

nx.draw_networkx_nodes(grafo, pos, nodelist=grafo.nodes(), node_color=colors, alpha
nx.draw_networkx_edges(grafo, pos, width=0.3, alpha=0.5)
nx.draw_networkx_labels(grafo, pos)
```



In [44]:

```
roles = {}

for node in role_extractor.roles:
    if role_extractor.roles[node] in roles: roles[role_extractor.roles[node]].appe
    else: roles[role_extractor.roles[node]] = [node]

print(f"Hay {len(roles)} roles:")
for role in roles: print(f"- {role} tiene {len(roles[role])} países")
print()
pprint(roles, compact=True, width=100, indent=2)
```

```
Hay 5 roles:
- role_1 tiene 124 países
- role_2 tiene 73 países
- role_0 tiene 20 países
- role_4 tiene 9 países
- role_3 tiene 3 países

{ 'role_0': [ 'Austria', "Cote d'Ivoire", 'Cyprus', 'Czech Republic', 'Denmark',
'Egypt', 'Greece',
              'Hungary', 'India', 'Italy', 'Malaysia', 'Malta', 'Norway', 'Polan
d', 'Russia',
              'Sweden', 'Switzerland', 'Thailand', 'Ukraine', 'United States'],
  'role_1': [ 'Afghanistan', 'American Samoa', 'Anguilla', 'Antigua and Barbuda',
'Armenia',
              'Aruba', 'Bahamas', 'Barbados', 'Belize', 'Benin', 'Bermuda', 'Bhuta
n', 'Bolivia',
              'Botswana', 'British Virgin Islands', 'Brunei', 'Burkina Faso', 'Bur
ma', 'Burundi',
              'Cambodia', 'Cameroon', 'Cayman Islands', 'Central African Republi
c', 'Chad',
              'Christmas Island', 'Cocos (Keeling) Islands', 'Comoros', 'Congo (Br
azzaville)',
              'Congo (Kinshasa)', 'Cook Islands', 'Djibouti', 'Dominica', 'East Ti
mor', 'Ecuador',
              'El Salvador', 'Equatorial Guinea', 'Eritrea', 'Falkland Islands',
'Faroe Islands',
              'Fiji', 'French Guiana', 'French Polynesia', 'Gabon', 'Gambia', 'Gib
raltar',
              'Greenland', 'Grenada', 'Guadeloupe', 'Guam', 'Guatemala', 'Guernse
y', 'Guinea',
              'Guinea-Bissau', 'Guyana', 'Haiti', 'Honduras', 'Isle of Man', 'Jama
ica', 'Jersey',
              'Kiribati', 'Kyrgyzstan', 'Laos', 'Lesotho', 'Liberia', 'Libya', 'Ma
cau',
              'Madagascar', 'Malawi', 'Mali', 'Marshall Islands', 'Martinique', 'M
auritania',
              'Mayotte', 'Mongolia', 'Montserrat', 'Mozambique', 'Myanmar', 'Namib
ia', 'Nauru',
              'Nepal', 'Netherlands Antilles', 'New Caledonia', 'New Zealand', 'Ni
caragua', 'Niger',
              'Niue', 'Norfolk Island', 'North Korea', 'Northern Mariana Islands',
'Palau',
              'Papua New Guinea', 'Paraguay', 'Peru', 'Puerto Rico', 'Reunion', 'R
wanda',
              'Saint Helena', 'Saint Kitts and Nevis', 'Saint Lucia', 'Saint Pierr
e and Miquelon',
              'Saint Vincent and the Grenadines', 'Samoa', 'Sao Tome and Princip
e', 'Sierra Leone',
              'Somalia', 'South Sudan', 'Sudan', 'Suriname', 'Swaziland', 'Syria',
'Tajikistan',
              'Togo', 'Tonga', 'Trinidad and Tobago', 'Turkmenistan', 'Turks and C
aicos Islands',
              'Tuvalu', 'Uruguay', 'Vanuatu', 'Virgin Islands', 'Western Sahara',
'Yemen', 'Zambia',
              'Zimbabwe'],
  'role_2': [ 'Albania', 'Algeria', 'Angola', 'Argentina', 'Australia', 'Azerbaija
n', 'Bahrain',
              'Bangladesh', 'Belarus', 'Bosnia and Herzegovina', 'Brazil', 'Bulgar
ia', 'Canada',
              'Cape Verde', 'Chile', 'China', 'Colombia', 'Costa Rica', 'Croatia',
'Cuba',
              'Dominican Republic', 'Estonia', 'Ethiopia', 'Finland', 'Georgia',
'Ghana',
              'Hong Kong', 'Iceland', 'Indonesia', 'Iran', 'Iraq', 'Ireland', 'Isr
```

```
ael', 'Japan',
              'Jordan', 'Kazakhstan', 'Kenya', 'Kuwait', 'Latvia', 'Lebanon', 'Lit
huania',
              'Luxembourg', 'Macedonia', 'Maldives', 'Mauritius', 'Mexico', 'Moldo
va', 'Montenegro',
              'Morocco', 'Nigeria', 'Oman', 'Pakistan', 'Panama', 'Philippines',
'Portugal',
              'Romania', 'Saudi Arabia', 'Senegal', 'Serbia', 'Seychelles', 'Singa
pore', 'Slovakia',
              'Slovenia', 'South Africa', 'South Korea', 'Sri Lanka', 'Taiwan', 'T
anzania',
              'Tunisia', 'Uganda', 'Uzbekistan', 'Venezuela', 'Vietnam'],
  'role_3': ['Micronesia', 'Solomon Islands', 'Wallis and Futuna'],
  'role_4': [ 'Belgium', 'France', 'Germany', 'Netherlands', 'Qatar', 'Spain', 'Tu
rkey',
              'United Arab Emirates', 'United Kingdom']}
```

Quedan definidos 5 roles. Los roles parecen quedar definidos según la importancia de cada país.

Se pueden observar en el grupo más central, el de color verde, los siguientes países:

- Alemania, España, Holanda, Francia, Bélgica, Qatar, Turquía, Reino Unido y Emiratos Arabes Unidos.

Estos son países de mucha importancia económica y turística y tienen una ubicación buena en cuanto al tráfico aéreo se refiere. La mayor parte de estos se encuentran en Europa, continente el cual es muy visitado y también es muy utilizado para hacer escalas. Además, se encuentran países como Qatar y Emiratos Arabes Unidos, que cuentan con dos aerolíneas muy grandes, con un turismo que creció mucho en los últimos años (pre-pandemia) y probablemente también se los utilicen como puntos intermedios dado que se encuentran en un punto medio entre Europa occidental y Oceanía o el sureste asiático.

Luego, está el grupo de color amarillo, con países como:

- Austria, Republica Chena, Dinamarca, Grecia, Egipto, Polonia, Costa de Marfil, Noruega, entre otros.

En este caso parece haber también países de mucha relevancia tanto económica, turística y de posición geográfica, pero en menor parte que los anteriormente mencionados. Cabe destacar que también se encuentran países como Estados Unidos, Rusia, India e Italia, que uno esperaría que tuvieran una mayor relevancia.

Más alejados del centro del gráfico, se encuentra el grupo de color rojo. En este se encuentran países como:

- Argentina, Austrania, Brasil, Croacia, Chile, Cuba, Finlancia, Indonesia, Iran, Irlanda, Serbia, Taiwan, Vietnam, entre otros.

Es un grupo de países más grande que los anteriores, donde se vuelve a ver el patrón de países con una relevancia un poco menor. En este caso, es destacable la aparición de China, la cual uno esperaría que se encontrase en un rol de mayor importancia.

Este patrón sigue apareciendo en los grupos restantes, tanto en el azul como en el gris.