FERNANDEZ_Exercise5

October 25, 2024

Exercise 5

FERNANDEZ FRAGOSO Aldo Enrique

[1]:
```
## Importing necessary packages
import networkx as nx
import pandas as pd
import requests
import matplotlib.pyplot as plt
```

[2]:
```
urls = ["https://raw.githubusercontent.com/1250326/exercise_complex_network/
 ↪e478a2b7e0cc8c0746d4c689a7c52504277a80e1/Datasets/Group3/Facebook-Ego/348.
 ↪edges",
       "https://raw.githubusercontent.com/1250326/exercise_complex_network/
 ↪e478a2b7e0cc8c0746d4c689a7c52504277a80e1/Datasets/Group3/Facebook-Ego/348.
 ↪egofeat",
       "https://raw.githubusercontent.com/1250326/exercise_complex_network/
 ↪e478a2b7e0cc8c0746d4c689a7c52504277a80e1/Datasets/Group3/Facebook-Ego/348.
 ↪feat",
       "https://raw.githubusercontent.com/1250326/exercise_complex_network/
 ↪e478a2b7e0cc8c0746d4c689a7c52504277a80e1/Datasets/Group3/Facebook-Ego/348.
 ↪featnames"]

def download_data(urls):
  name = []
  i = 0
  for url in urls:
    response = requests.get(url)
    name.append(response.url.split('/')[-1])

    if response.status_code == 200:
        with open(name[i], "wb") as file:
            file.write(response.content)
            i = i+1
        print("File downloaded successfully.")

    else:
        print("Failed to download the file.")
```

```
  return "Finished downloading"

download_data(urls)

## Extracting datasets
features_file = "348.edges"
edges_file = "348.edges"
egofeat_file = "348.egofeat"
featnames_file = "348.featnames"

G_facebook = nx.read_edgelist(edges_file, nodetype=int)
features = pd.read_csv(features_file, sep=" ", header=None)
ego_features = pd.read_csv(egofeat_file, sep=" ", header=None)
feat_names = pd.read_csv(featnames_file, sep=" ", header=None)

# Add features to each node in the graph
for i, node in enumerate(G_facebook.nodes()):
    G_facebook.nodes[node]['features'] = features.iloc[i].values


# (a) How many nodes and edges are there in the networks?
print(f'Number of edges: {G_facebook.number_of_edges()}')
print(f'Number of nodes in the graph: {len(G_facebook)}')


### Twitter

urls = ["https://raw.githubusercontent.com/1250326/exercise_complex_network/
 ↪e478a2b7e0cc8c0746d4c689a7c52504277a80e1/Datasets/Group3/Twitter-Ego/789071.
 ↪circles",
       "https://raw.githubusercontent.com/1250326/exercise_complex_network/
 ↪e478a2b7e0cc8c0746d4c689a7c52504277a80e1/Datasets/Group3/Twitter-Ego/789071.
 ↪edges",
       "https://raw.githubusercontent.com/1250326/exercise_complex_network/
 ↪e478a2b7e0cc8c0746d4c689a7c52504277a80e1/Datasets/Group3/Twitter-Ego/789071.
 ↪egofeat",
       "https://raw.githubusercontent.com/1250326/exercise_complex_network/
 ↪e478a2b7e0cc8c0746d4c689a7c52504277a80e1/Datasets/Group3/Twitter-Ego/789071.
 ↪feat",
       "https://raw.githubusercontent.com/1250326/exercise_complex_network/
 ↪e478a2b7e0cc8c0746d4c689a7c52504277a80e1/Datasets/Group3/Twitter-Ego/789071.
 ↪featnames"]

download_data(urls)

## Extracting datasets
features_file = "789071.feat"
```

```
edges_file = "789071.edges"
egofeat_file = "789071.egofeat"
featnames_file = "789071.featnames"
circles_file = "789071.circles"

G_twitter = nx.read_edgelist(edges_file, nodetype=int,create_using=nx.DiGraph())
features = pd.read_csv(features_file, sep=" ", header=None)
ego_features = pd.read_csv(egofeat_file, sep=" ", header=None)
feat_names = pd.read_csv(featnames_file, sep=" ", header=None)
circles = pd.read_csv(circles_file, sep=" ", header=None)

# Add features to each node in the graph
for i, node in enumerate(G_twitter.nodes()):
    G_twitter.nodes[node]['features'] = features.iloc[i].values

    ## Extracting datasets
features_file = "789071.feat"
edges_file = "789071.edges"
egofeat_file = "789071.egofeat"
featnames_file = "789071.featnames"
circles_file = "789071.circles"

G_twitter = nx.read_edgelist(edges_file, nodetype=int,create_using=nx.DiGraph())
features = pd.read_csv(features_file, sep=" ", header=None)
ego_features = pd.read_csv(egofeat_file, sep=" ", header=None)
feat_names = pd.read_csv(featnames_file, sep=" ", header=None)
circles = pd.read_csv(circles_file, sep=" ", header=None)

# Add features to each node in the graph
for i, node in enumerate(G_twitter.nodes()):
    G_twitter.nodes[node]['features'] = features.iloc[i].values

  # (a) How many nodes and edges are there in the networks?
number_edges = G_twitter.number_of_edges()
number_nodes = G_twitter.number_of_nodes()
print(f'Number of edges: {number_edges}')
print(f'Number of nodes in the graph: {number_nodes}')
```

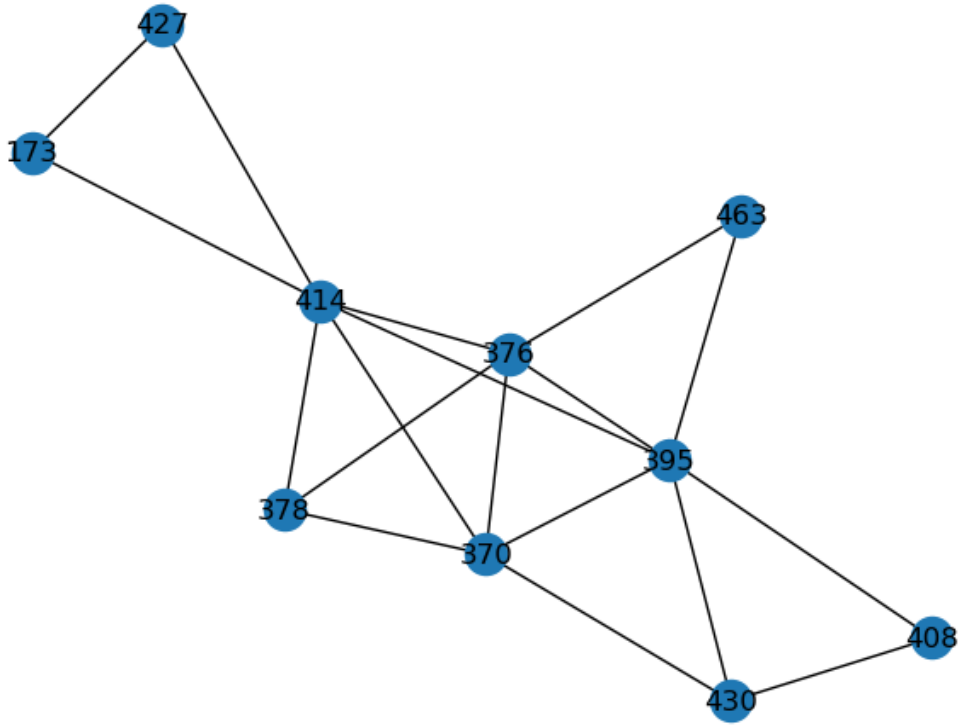
```
File downloaded successfully.
File downloaded successfully.
File downloaded successfully.
File downloaded successfully.
Number of edges: 1248
Number of nodes in the graph: 148
File downloaded successfully.
File downloaded successfully.
File downloaded successfully.
File downloaded successfully.
```

```
File downloaded successfully.
Number of edges: 1128
Number of nodes in the graph: 138
```

(a) Please choose ten nodes from either dataset, and return a SubGraph View of the subgraphs induced on the chosen ten nodes

[3]:
```
print(G_facebook.nodes)
list_nodes = [395, 370, 463, 430, 427, 378, 414, 173, 376, 408]

subgraph_facebook = G_facebook.subgraph(list_nodes)
print("Nodes in Subgraph:", subgraph_facebook.nodes())
print("Edges in Subgraph:", subgraph_facebook.edges())

nx.draw(subgraph_facebook, with_labels=True)
plt.show()
```

```
[436, 428, 471, 461, 465, 475, 446, 367, 452, 450, 361, 359, 378, 482, 484, 492,
479, 373, 412, 398, 395, 370, 416, 368, 423, 457, 477, 364, 441, 496, 382, 474,
396, 451, 371, 460, 420, 432, 448, 397, 439, 443, 453, 403, 422, 483, 402, 413,
495, 376, 445, 444, 417, 418, 476, 463, 388, 438, 488, 442, 491, 419, 387, 456,
458, 424, 470, 391, 426, 404, 392, 353, 360, 421, 408, 430, 427, 464, 400, 407,
431, 350, 369, 493, 375, 466, 429, 486, 459, 394, 355, 354, 409, 494, 481, 487,
374, 198, 399, 352, 489, 473, 454, 351, 366, 490, 363, 434, 390, 372, 410, 414,
415, 468, 406, 437, 362, 173, 462, 478, 349, 425, 357, 455, 449, 469, 467, 435,
385, 440, 380, 389, 405, 381, 485, 34, 356, 411, 480, 433, 393, 472, 365, 384,
401, 386, 377, 383]
Nodes in Subgraph: [408, 395, 427, 173, 430, 463, 370, 376, 378, 414]
Edges in Subgraph: [(408, 430), (408, 395), (395, 370), (395, 414), (395, 463),
(395, 376), (395, 430), (427, 173), (427, 414), (173, 414), (430, 370), (463,
376), (370, 414), (370, 378), (370, 376), (376, 414), (376, 378), (378, 414)]
```



(b) Please list three algorithms for community detection.

Agglomerative: begin with singleton groups and join successively by similarity Divisive: begin with one group containing all points and divide successivel

1. Girvan-Newman (Agglomerative)
2. Louvain (Divisive)
3. Leiden

(c) Please choose one of the datasets, and utilize all of the algorithms listed in (b) to detect communities in the dataset. Please plot graphs to show the communities (mark nodes with different colors / draw communities unions….

[5]:
```
!pip install community
## import algorithms packages
import community as community_louvain
from networkx.algorithms.community import girvan_newman
from networkx.algorithms.community import label_propagation_communities
import numpy as np
```

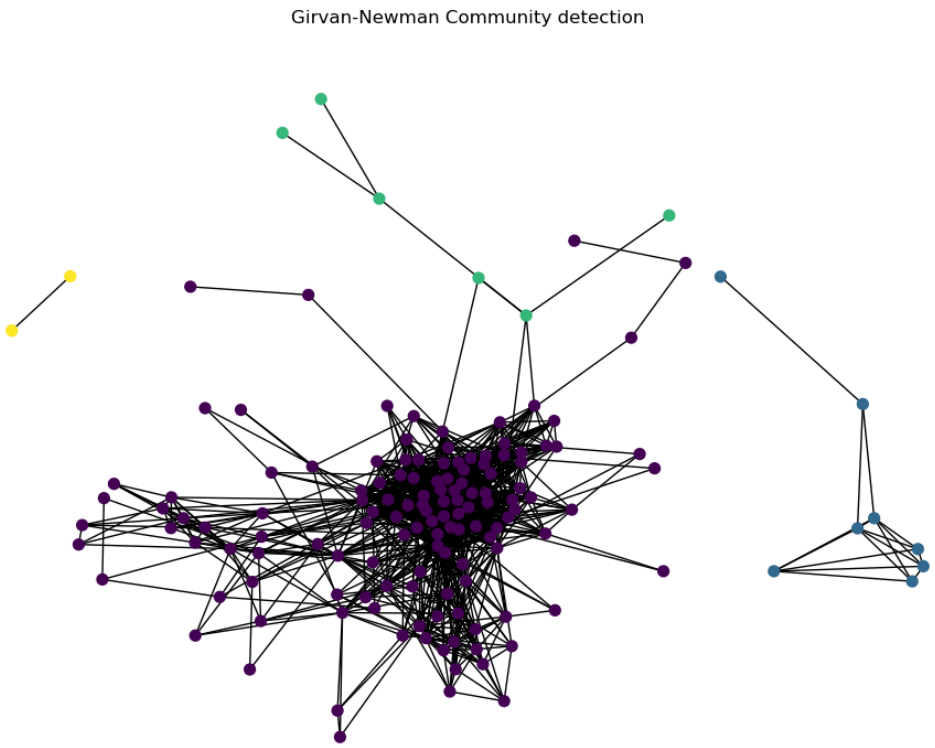
```
Collecting community
  Downloading community-1.0.0b1.tar.gz (2.2 kB)
  Preparing metadata (setup.py): started
```

```
  Preparing metadata (setup.py): finished with status 'done'
Requirement already satisfied: Flask in c:\users\aldoe\anaconda3\lib\site-
packages (from community) (2.2.2)
Requirement already satisfied: requests in c:\users\aldoe\anaconda3\lib\site-
packages (from community) (2.31.0)
Requirement already satisfied: Werkzeug>=2.2.2 in
c:\users\aldoe\anaconda3\lib\site-packages (from Flask->community) (2.2.3)
Requirement already satisfied: Jinja2>=3.0 in c:\users\aldoe\anaconda3\lib\site-
packages (from Flask->community) (3.1.2)
Requirement already satisfied: itsdangerous>=2.0 in
c:\users\aldoe\anaconda3\lib\site-packages (from Flask->community) (2.0.1)
Requirement already satisfied: click>=8.0 in c:\users\aldoe\anaconda3\lib\site-
packages (from Flask->community) (8.0.4)
Requirement already satisfied: charset-normalizer<4,>=2 in
c:\users\aldoe\anaconda3\lib\site-packages (from requests->community) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in
c:\users\aldoe\anaconda3\lib\site-packages (from requests->community) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in
c:\users\aldoe\anaconda3\lib\site-packages (from requests->community) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in
c:\users\aldoe\anaconda3\lib\site-packages (from requests->community)
(2023.7.22)
Requirement already satisfied: colorama in c:\users\aldoe\anaconda3\lib\site-
packages (from click>=8.0->Flask->community) (0.4.6)
Requirement already satisfied: MarkupSafe>=2.0 in
c:\users\aldoe\anaconda3\lib\site-packages (from Jinja2>=3.0->Flask->community)
(2.1.1)
Building wheels for collected packages: community
  Building wheel for community (setup.py): started
  Building wheel for community (setup.py): finished with status 'done'
  Created wheel for community: filename=community-1.0.0b1-py3-none-any.whl
size=2143
sha256=a58522018fc50373fdd53aa86c3d388e6603281137da6de3d1e834a3fa87ad90
  Stored in directory: c:\users\aldoe\appdata\local\pip\cache\wheels\b7\c9\3f\e2
22b011e31d3d1de1fd799caed228f770d56f66563014285d
Successfully built community
Installing collected packages: community
Successfully installed community-1.0.0b1
```

1 Girvan-Newman algorithm

[6]:
```
import networkx as nx
import matplotlib.pyplot as plt
from networkx.algorithms.community import girvan_newman
import itertools

# Girvan-Newman algorithm
```

```
comp = girvan_newman(G_facebook)
limited = itertools.takewhile(lambda c: len(c) <= 10, comp)
communities = next(limited)

# Assign colors to communities
color_map = {}
for i, community in enumerate(communities):
    for node in community:
        color_map[node] = i


colors = [color_map[node] for node in G_facebook.nodes()]
plt.figure(figsize=(10,7))
pos = nx.spring_layout(G_facebook, k=0.5, iterations=100)
nx.draw(G_facebook, pos, node_size = 50,  node_color=colors, with_labels=False)
plt.title('Girvan-Newman Community detection')
plt.show()
```



2 Louvain Algorithm

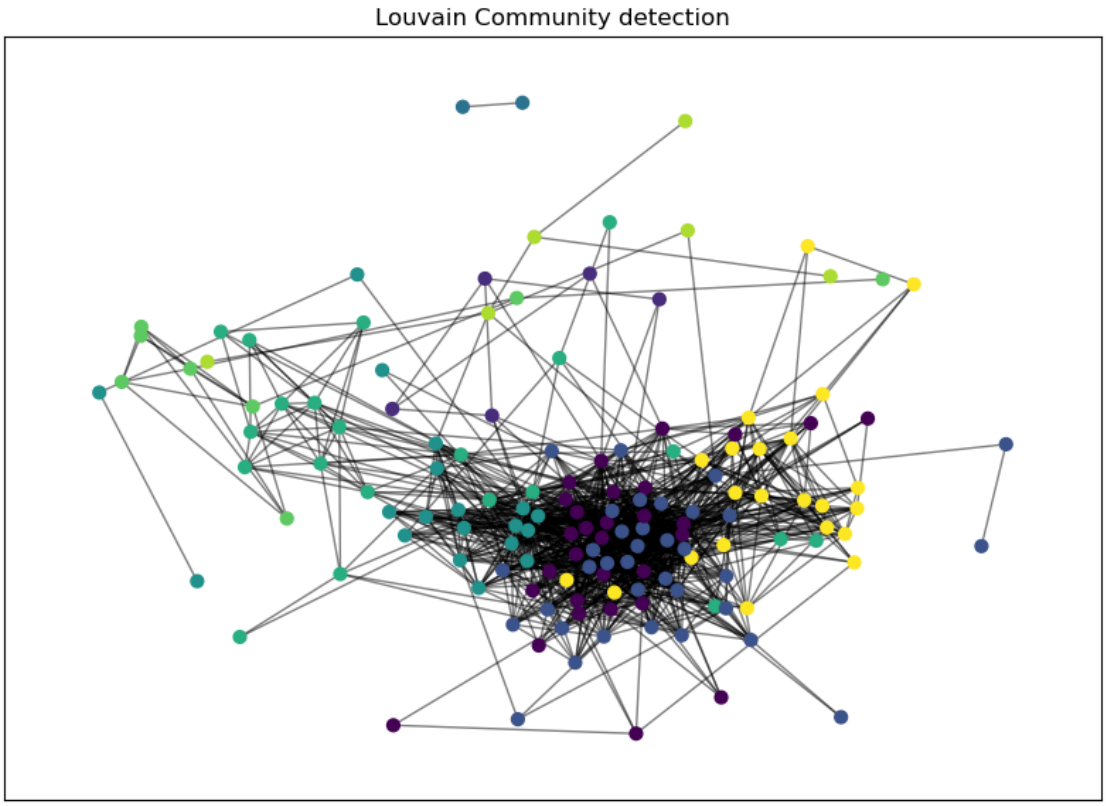
[16]:
```
!pip install networkx python-louvain matplotlib
```

```
Requirement already satisfied: networkx in c:\users\aldoe\anaconda3\lib\site-
packages (3.1)
Requirement already satisfied: python-louvain in
c:\users\aldoe\anaconda3\lib\site-packages (0.16)
Requirement already satisfied: matplotlib in c:\users\aldoe\anaconda3\lib\site-
packages (3.7.1)
Requirement already satisfied: numpy in c:\users\aldoe\anaconda3\lib\site-
packages (from python-louvain) (1.24.3)
Requirement already satisfied: contourpy>=1.0.1 in
c:\users\aldoe\anaconda3\lib\site-packages (from matplotlib) (1.0.5)
Requirement already satisfied: cycler>=0.10 in
c:\users\aldoe\anaconda3\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in
c:\users\aldoe\anaconda3\lib\site-packages (from matplotlib) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
c:\users\aldoe\anaconda3\lib\site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: packaging>=20.0 in
c:\users\aldoe\anaconda3\lib\site-packages (from matplotlib) (23.0)
Requirement already satisfied: pillow>=6.2.0 in
c:\users\aldoe\anaconda3\lib\site-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
c:\users\aldoe\anaconda3\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in
c:\users\aldoe\anaconda3\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\aldoe\anaconda3\lib\site-
packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
```

[9]:
```
import matplotlib.cm as cm
#!pip install python-louvain

from community import community_louvain

partition = community_louvain.best_partition(G_facebook)
plt.figure(figsize=(10,7))
pos = nx.spring_layout(G_facebook, k=0.9, iterations=100)
cmap = cm.get_cmap('viridis', max(partition.values()) + 1)
nx.draw_networkx_nodes(G_facebook, pos, partition.keys(), node_size=40,
                       cmap=cmap, node_color=list(partition.values()))
nx.draw_networkx_edges(G_facebook, pos, alpha=0.5)
plt.title('Louvain Community detection')
plt.show()
```

```
C:\Users\aldoe\AppData\Local\Temp\ipykernel_37772\2707732404.py:9:
MatplotlibDeprecationWarning: The get_cmap function was deprecated in Matplotlib
```

```
3.7 and will be removed two minor releases later. Use
``matplotlib.colormaps[name]`` or ``matplotlib.colormaps.get_cmap(obj)``
instead.
  cmap = cm.get_cmap('viridis', max(partition.values()) + 1)
```



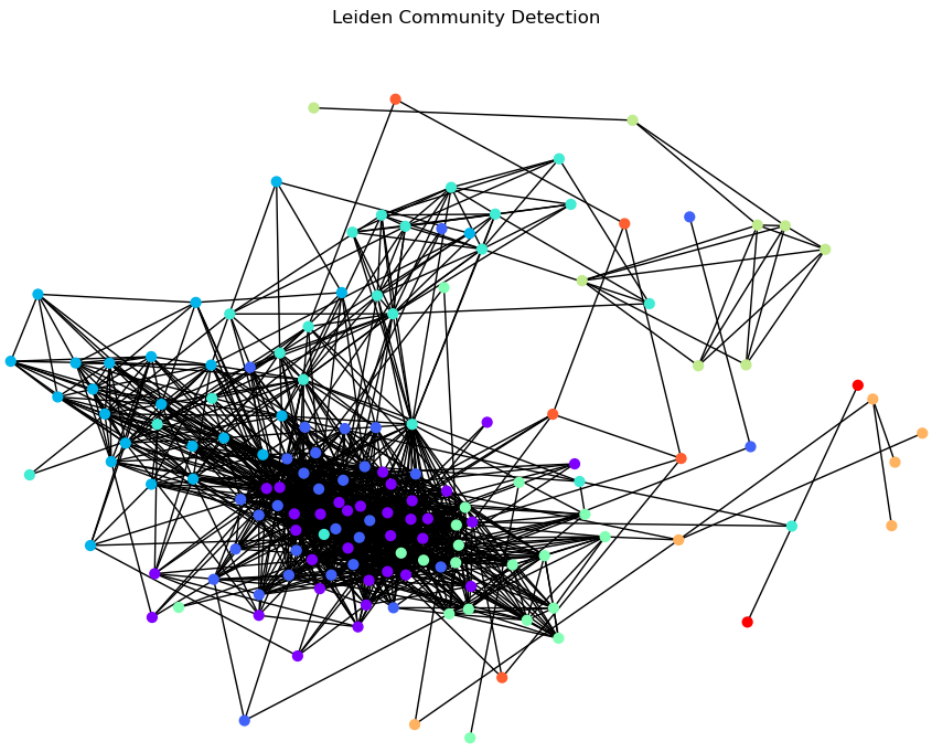
Leiden algorithm

[18]:
```
#!pip install cdlib
!pip install igraph leidenalg
from cdlib import algorithms


plt.figure(figsize=(10,7))
pos = nx.spring_layout(G_facebook, k=0.9, iterations=100)

coms = algorithms.leiden(G_facebook)
partition = coms.communities

color_map = {}
for i, community in enumerate(partition):
    for node in community:
        color_map[node] = i
```

```
# Apply colors to nodes based on their communities
colors = [color_map[node] for node in G_facebook.nodes()]
plt.figure(figsize=(10,7))
nx.draw(G_facebook,pos=pos, node_color=colors, with_labels=False, cmap=plt.cm.
 ↪rainbow, node_size = 40)
plt.title('Leiden Community Detection')
plt.show()
```

```
Requirement already satisfied: igraph in c:\users\aldoe\anaconda3\lib\site-
packages (0.11.6)
Collecting leidenalg
  Obtaining dependency information for leidenalg from https://files.pythonhosted
.org/packages/29/2f/f315fca13523f6b7520b810cce942348e37e8e5ea4c39e4eaaeb3541fee0
/leidenalg-0.10.2-cp38-abi3-win_amd64.whl.metadata
  Downloading leidenalg-0.10.2-cp38-abi3-win_amd64.whl.metadata (10 kB)
Requirement already satisfied: texttable>=1.6.2 in
c:\users\aldoe\anaconda3\lib\site-packages (from igraph) (1.7.0)
Downloading leidenalg-0.10.2-cp38-abi3-win_amd64.whl (1.6 MB)
   ---------------------------------------- 0.0/1.6 MB ? eta -:--:--
   ---------- ----------------------------- 0.6/1.6 MB 13.3 MB/s eta 0:00:01
   ---------------------------------------  1.6/1.6 MB 20.0 MB/s eta 0:00:01
   ---------------------------------------  1.6/1.6 MB 20.0 MB/s eta 0:00:01
   ---------------------------------------- 1.6/1.6 MB 10.0 MB/s eta 0:00:00
Installing collected packages: leidenalg
Successfully installed leidenalg-0.10.2
```

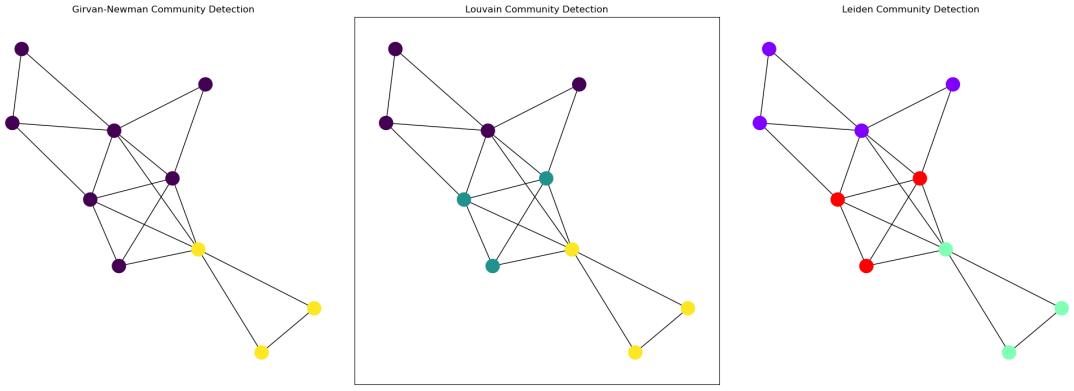
```
<Figure size 1000x700 with 0 Axes>
```



2.1 Algorithms applied on the subgraph

[19]:
```
## Layout
pos = nx.spring_layout(subgraph_facebook, k=0.5, iterations=100)
fig, ax = plt.subplots(1, 3, figsize=(20, 7))

# Girvan-Newman Algorithm
comp = girvan_newman(subgraph_facebook)
limited = itertools.takewhile(lambda c: len(c) <= 100, comp)
communities = next(limited)

color_map = {}
for i, community in enumerate(communities):
    for node in community:
        color_map[node] = i

colors = [color_map[node] for node in subgraph_facebook.nodes()]
ax[0].set_title('Girvan-Newman Community Detection')
nx.draw(subgraph_facebook, pos, ax=ax[0], node_color=colors, with_labels=False)
```

```
# Louvain Method
partition = community_louvain.best_partition(subgraph_facebook)

cmap = cm.get_cmap('viridis', max(partition.values()) + 1)
ax[1].set_title('Louvain Community Detection')
nx.draw_networkx_nodes(subgraph_facebook, pos, ax=ax[1], cmap=cmap,␣
 ↪node_color=list(partition.values()))
nx.draw_networkx_edges(subgraph_facebook, pos, ax=ax[1])


# Leiden Algorithm
coms = algorithms.leiden(subgraph_facebook)
partition = coms.communities


color_map = {}
for i, community in enumerate(partition):
    for node in community:
        color_map[node] = i

colors = [color_map[node] for node in subgraph_facebook.nodes()]
ax[2].set_title('Leiden Community Detection')
nx.draw(subgraph_facebook, pos=pos, ax=ax[2], node_color=colors,␣
 ↪with_labels=False, cmap=plt.cm.rainbow)

## Show plot
plt.tight_layout()
plt.show()
```



(d) What are the differences between the community detection results shown in (c)? What are the possible reasons?

On the subgraph it was easier to see the number of communities detected by each of the algorithms,

we can detect that **Girvan-Newman** was detecting just two communities, while on the other hand the other two algorithms (**Louvain and Leiden**) are detecting one more community, this can be because Girvan-Newman algorithms identifies smaller communities, while **leiden** is related to **louvain** algorithm which optimizes the modularity of the graph, which is a measure of the density of links inside communities.