In-Between: Communities

In [ ]:
```python
# Configure plotting in Jupyter
from matplotlib import pyplot as plt
%matplotlib inline
plt.rcParams.update({
    'figure.figsize': (7.5, 7.5),
    'axes.spines.right': False,
    'axes.spines.left': False,
    'axes.spines.top': False,
    'axes.spines.bottom': False})
# Seed random number generator
import random
from numpy import random as nprand
seed = hash("Network Science in Python") % 2**32
nprand.seed(seed)
random.seed(seed)
```

In [ ]:
```python
import networkx as nx
```

Community Detection in NetworkX

In [ ]:
```python
import networkx.algorithms.community as nxcom
# Generate the network
G_karate = nx.karate_club_graph()
# Find the communities
communities = sorted(nxcom.greedy_modularity_communities(G_karate), key=len, reverse=True)
# Count the communities
len(communities)
```

Out[ ]:
```
3
```

In [ ]:
```python
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
```

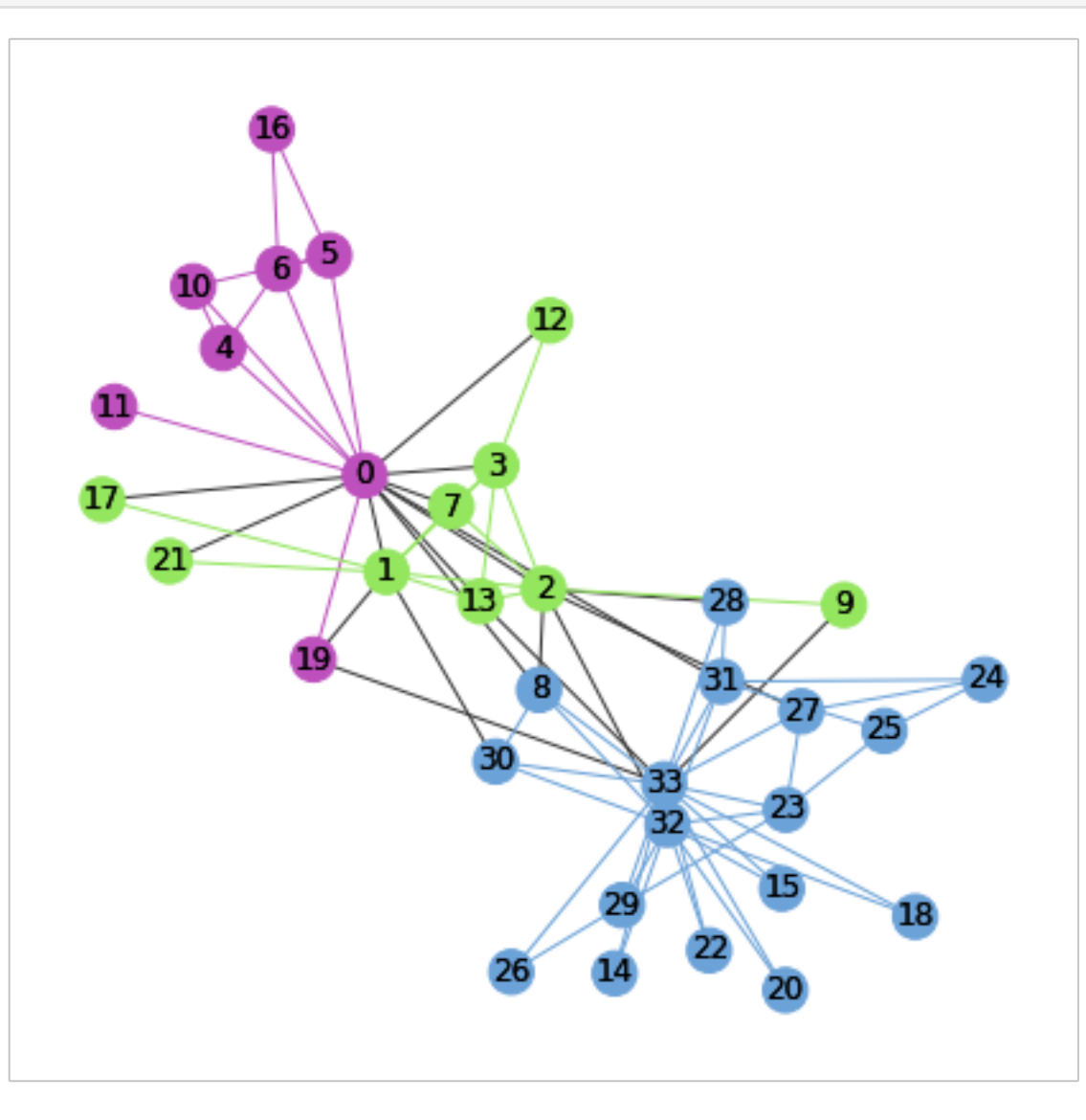
In [ ]:
```python
def get_color(i, r_off=1, g_off=1, b_off=1):
    r0, g0, b0 = 0, 0, 0
    n = 16
    low, high = 0.1, 0.9
    span = high - low
    r = low + span * (((i + r_off) * 3) % n) / (n - 1)
    g = low + span * (((i + g_off) * 5) % n) / (n - 1)
    b = low + span * (((i + b_off) * 7) % n) / (n - 1)
    return (r, g, b)
```

In [ ]:
```python
# Set node and edge communities
set_node_community(G_karate, communities)
set_edge_community(G_karate)

# Set community color for nodes
node_color = [get_color(G_karate.nodes[v]['community']) for v in G_karate.nodes]

# Set community color for internal edges
external = [(v, w) for v, w in G_karate.edges if G_karate.edges[v, w]['community'] == 0]
internal = [(v, w) for v, w in G_karate.edges if G_karate.edges[v, w]['community'] > 0]
internal_color = [get_color(G_karate.edges[e]['community']) for e in internal]
```

In [ ]:
```python
karate_pos = nx.spring_layout(G_karate)
# Draw external edges
nx.draw_networkx(
    G_karate,
    pos=karate_pos,
    node_size=0,
    edgelist=external,
    edge_color="#333333")
# Draw nodes and internal edges
nx.draw_networkx(
    G_karate,
    pos=karate_pos,
    node_color=node_color,
    edgelist=internal,
    edge_color=internal_color)
```



Online social nework

In [ ]:
```python
# Load data file into network
from pathlib import Path
data_dir = Path('.') / 'data'
G_social = nx.read_edgelist(data_dir / 'mcauley2012' / 'facebook_combined.txt')
```

In [ ]:
```python
# Caluclate layout and draw
pos = nx.spring_layout(G_social, k=0.1)
nx.draw_networkx(
    G_social, pos=pos, node_size=0, edge_color="#333333", alpha=0.05, with_labels=False)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/Users/NathanBick/Documents/Graduate School/MATH517 - Social Network Analysis/Network-Science-with-Python-and-NetworkX-Quick-Start-Guide-master/Chapter07/Chapter_07.ipynb Cell 12' in <cell line: 2>()
      <a href='vscode-notebook-cell:/Users/NathanBick/Documents/Graduate%20School/MATH517%20-%20Social%20Network%20Analysis/Network-Science-with-Python-and-NetworkX-Quick-Start-Guide-master/Chapter07/Chapter_07.ipynb#ch0000011?line=0'>1</a> # Caluclate layout and draw
----> <a href='vscode-notebook-cell:/Users/NathanBick/Documents/Graduate%20School/MATH517%20-%20Social%20Network%20Analysis/Network-Science-with-Python-and-NetworkX-Quick-Start-Guide-master/Chapter07/Chapter_07.ipynb#ch0000011?line=1'>2</a> pos = nx.spring_layout(G_social, k=0.1)
      <a href='vscode-notebook-cell:/Users/NathanBick/Documents/Graduate%20School/MATH517%20-%20Social%20Network%20Analysis/Network-Science-with-Python-and-NetworkX-Quick-Start-Guide-master/Chapter07/Chapter_07.ipynb#ch0000011?line=2'>3</a> nx.draw_networkx(
      <a href='vscode-notebook-cell:/Users/NathanBick/Documents/Graduate%20School/MATH517%20-%20Social%20Network%20Analysis/Network-Science-with-Python-and-NetworkX-Quick-Start-Guide-master/Chapter07/Chapter_07.ipynb#ch0000011?line=3'>4</a>     G_social, pos=pos, node_size=0, edge_color="#333333", alpha=0.05, with_labels=False)

File <class 'networkx.utils.decorators.argmap'> compilation 4:4, in argmap_spring_layout_1(G, k, pos, fixed, iterations, threshold, weight, scale, center, dim, seed)
      2 import collections
      3 import gzip
----> 4 import inspect
      5 import itertools
      6 import re

File ~/miniconda3/envs/network/lib/python3.10/site-packages/networkx/drawing/layout.py:476, in spring_layout(G, k, pos, fixed, iterations, threshold, weight, scale, center, dim, seed)
    474 if len(G) < 500:  # sparse solver for large graphs
    475     raise ValueError
--> 476 A = nx.to_scipy_sparse_array(G, weight=weight, dtype="f")
    477 if k is None and fixed is not None:
    478     # We must adjust k by domain size for layouts not near 1x1
    479     nnodes, _ = A.shape

File ~/miniconda3/envs/network/lib/python3.10/site-packages/networkx/convert_matrix.py:923, in to_scipy_sparse_array(G, nodelist, dtype, weight, format)
    921         r += diag_index
    922         c += diag_index
--> 923     A = sp.sparse.coo_array((d, (r, c)), shape=(nlen, nlen), dtype=dtype)
    924 try:
    925     return A.asformat(format)

AttributeError: module 'scipy.sparse' has no attribute 'coo_array'
```

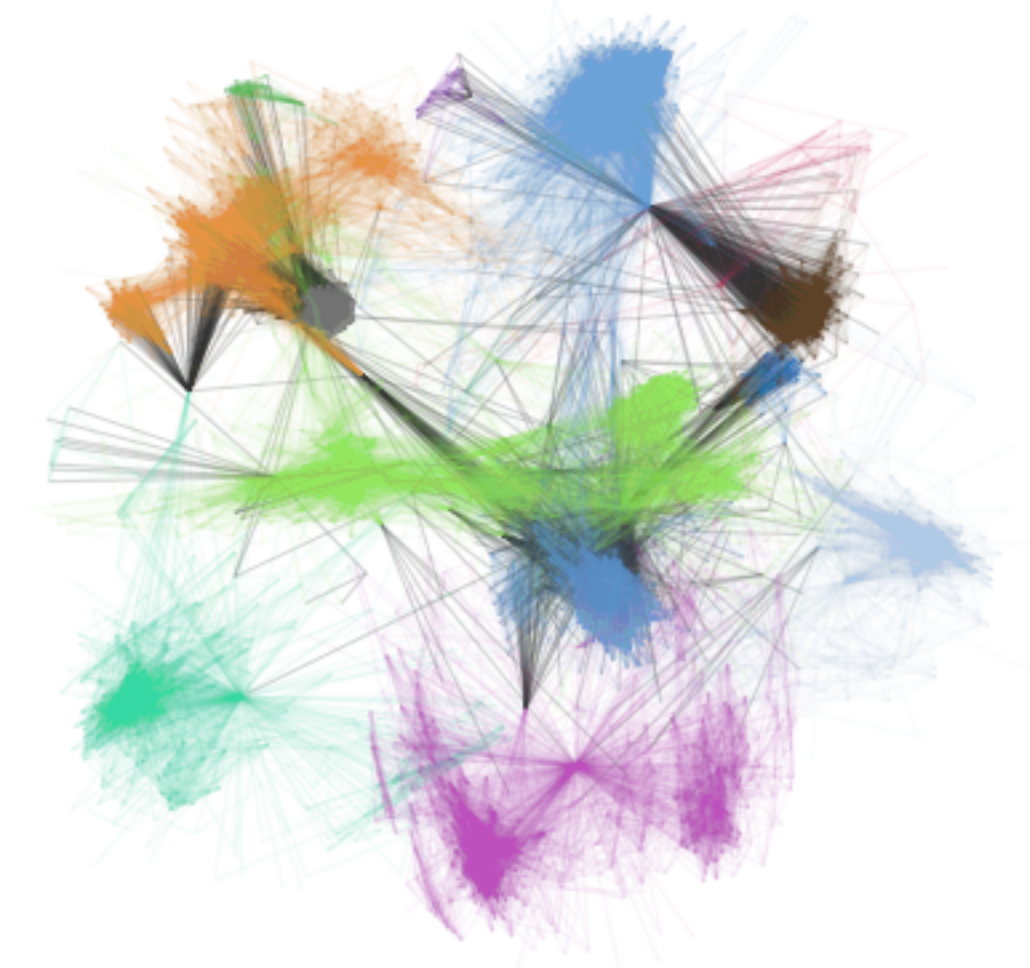
In [ ]:
```python
communities = sorted(nxcom.greedy_modularity_communities(G_social), key=len, reverse=True)
len(communities)
```

Out[ ]:
```
13
```

In [ ]:
```python
# Set node and edge communities
set_node_community(G_social, communities)
set_edge_community(G_social)
```

In [ ]:
```python
# Set community color for internal edges
external = [(v, w) for v, w in G_social.edges if G_social.edges[v, w]['community'] == 0]
internal = [(v, w) for v, w in G_social.edges if G_social.edges[v, w]['community'] > 0]
internal_color = [get_color(G_social.edges[e]['community']) for e in internal]
```

In [ ]:
```python
# Draw external edges
nx.draw_networkx(
    G_social, pos=pos, node_size=0, edgelist=external, edge_color="#333333",
    alpha=0.2, with_labels=False)
# Draw internal edges
nx.draw_networkx(
    G_social, pos=pos, node_size=0, edgelist=internal, edge_color=internal_color,
    alpha=0.05, with_labels=False)
```



Girvan-Newman Community Detection

In [ ]:
```python
result = nxcom.girvan_newman(G_karate)
communities = next(result)
len(communities)
```

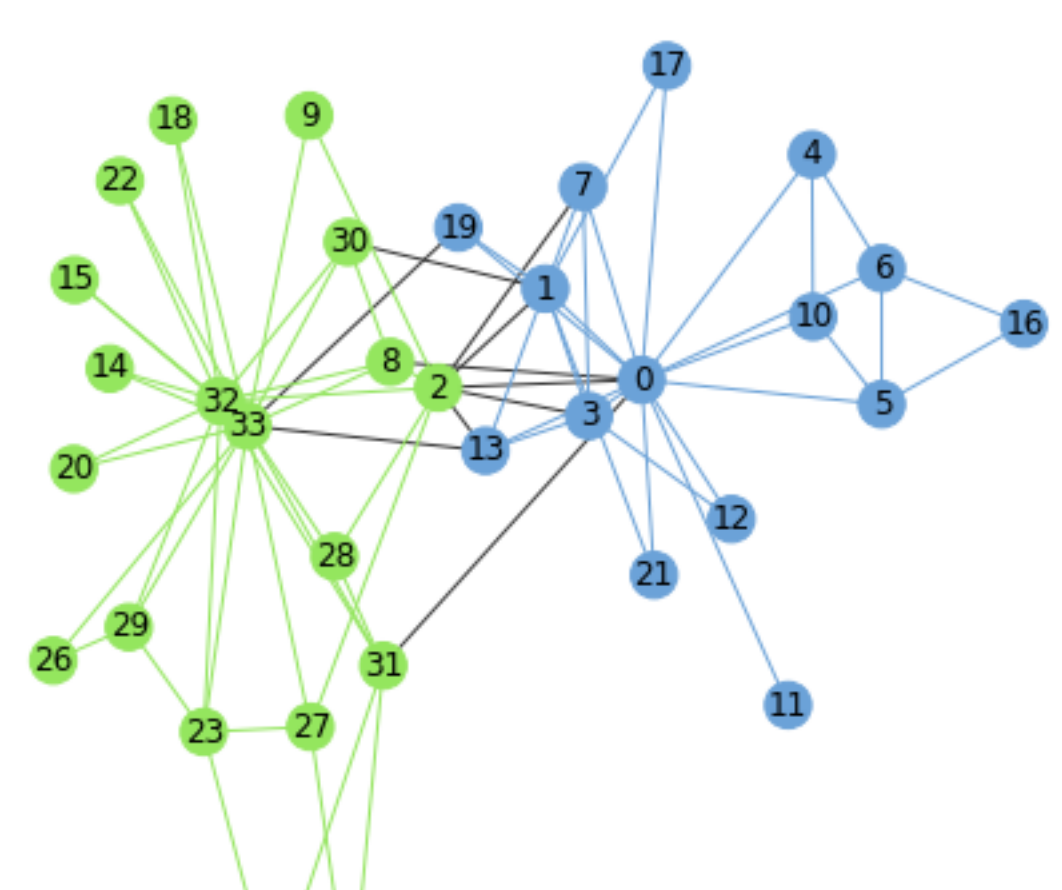
Out[ ]:
```
2
```

In [ ]:
```python
# Set node and edge communities
set_node_community(G_karate, communities)
set_edge_community(G_karate)
```

In [ ]:
```python
# Set community color for nodes
node_color = [get_color(G_karate.nodes[v]['community']) for v in G_karate.nodes]

# Set community color for internal edges
external = [(v, w) for v, w in G_karate.edges if G_karate.edges[v, w]['community'] == 0]
internal = [(v, w) for v, w in G_karate.edges if G_karate.edges[v, w]['community'] > 0]
internal_color = [get_color(G_karate.edges[e]['community']) for e in internal]
```

In [ ]:
```python
# Draw external edges
nx.draw_networkx(
    G_karate, pos=karate_pos, node_size=0,
    edgelist=external, edge_color="#333333", with_labels=False)
# Draw nodes and internal edges
nx.draw_networkx(
    G_karate, pos=karate_pos, node_color=node_color,
    edgelist=internal, edge_color=internal_color)
```



In [ ]:
```python
import itertools
result = nxcom.girvan_newman(G_karate)
communities = next(itertools.islice(result, 2, 3))
```

In [ ]:
```python
# Set node and edge communities
set_node_community(G_karate, communities)
set_edge_community(G_karate)
```

In [ ]:
```python
# Set community color for nodes
node_color = [get_color(G_karate.nodes[v]['community']) for v in G_karate.nodes]

# Set community color for internal edges
external = [(v, w) for v, w in G_karate.edges if G_karate.edges[v, w]['community'] == 0]
internal = [(v, w) for v, w in G_karate.edges if G_karate.edges[v, w]['community'] > 0]
internal_color = [get_color(G_karate.edges[e]['community']) for e in internal]
```

In [ ]:
```python
# Draw external edges
nx.draw_networkx(
    G_karate, pos=karate_pos, node_size=0,
    edgelist=external, edge_color="#333333", with_labels=False)
# Draw nodes and internal edges
nx.draw_networkx(
    G_karate, pos=karate_pos, node_color=node_color,
    edgelist=internal, edge_color=internal_color)
```

Cliques

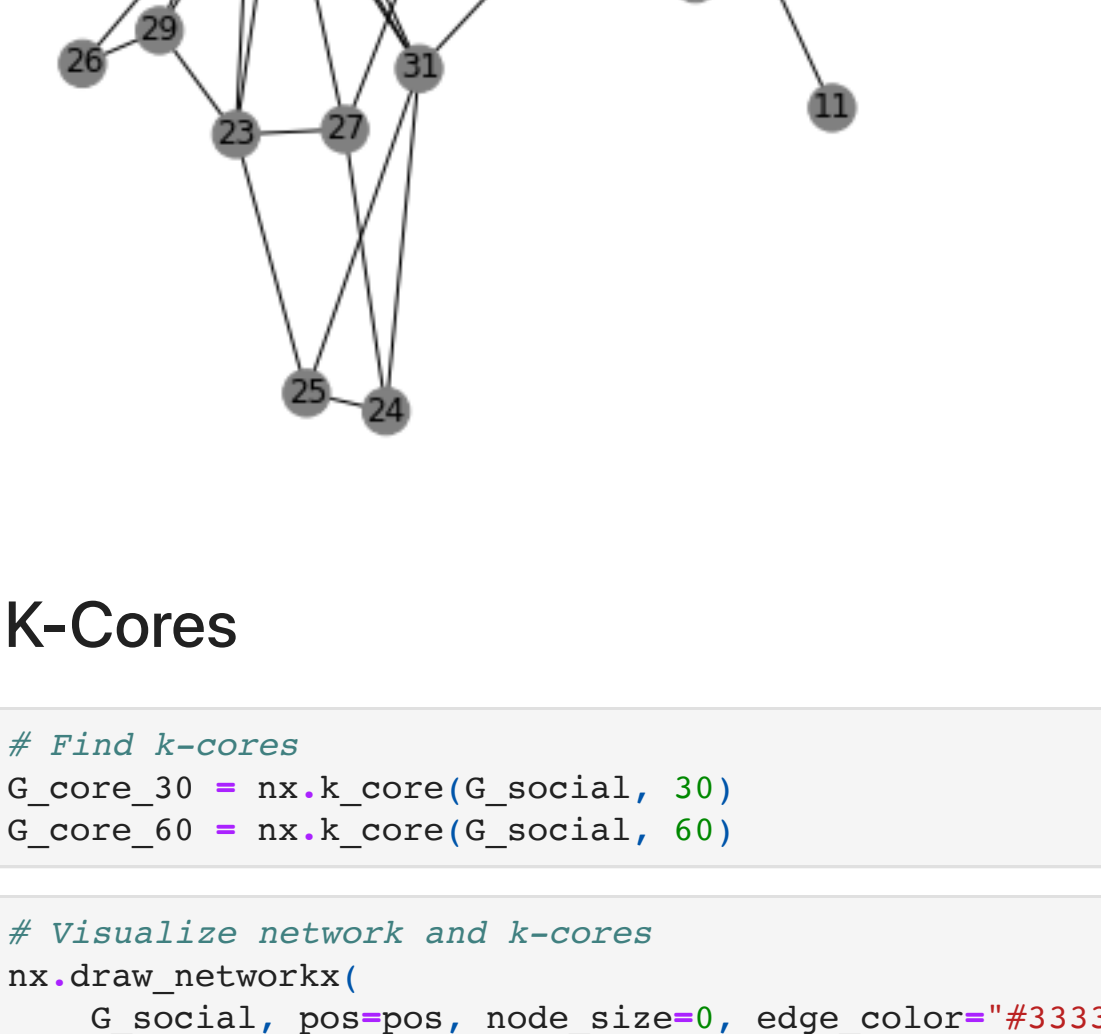
In [ ]:
```python
cliques = list(nx.find_cliques(G_karate))
cliques
```

Out[ ]:
```
[[0, 1, 17],
 [0, 1, 2, 3, 13],
 [0, 1, 2, 3, 7],
 [0, 1, 19],
 [0, 1, 21],
 [0, 4, 10],
 [0, 4, 6],
 [0, 5, 10],
 [0, 5, 6],
 [0, 8, 2],
 [0, 11],
 [0, 12, 3],
 [0, 31],
 [1, 30],
 [2, 32, 8],
 [2, 9],
 [2, 27],
 [2, 28],
 [5, 16, 6],
 [33, 32, 8, 30],
 [33, 32, 14],
 [33, 32, 15],
 [33, 32, 18],
 [33, 32, 20],
 [33, 32, 22],
 [33, 32, 23, 29],
 [33, 32, 31],
 [33, 9],
 [33, 13],
 [33, 19],
 [33, 26, 29],
 [33, 27, 23],
 [33, 28, 31],
 [24, 25, 31],
 [24, 27],
 [25, 23]]
```

In [ ]:
```python
# Find maximum clique
max_clique = max(cliques, key=len)
max_clique
```

Out[ ]:
```
[0, 1, 2, 3, 13]
```

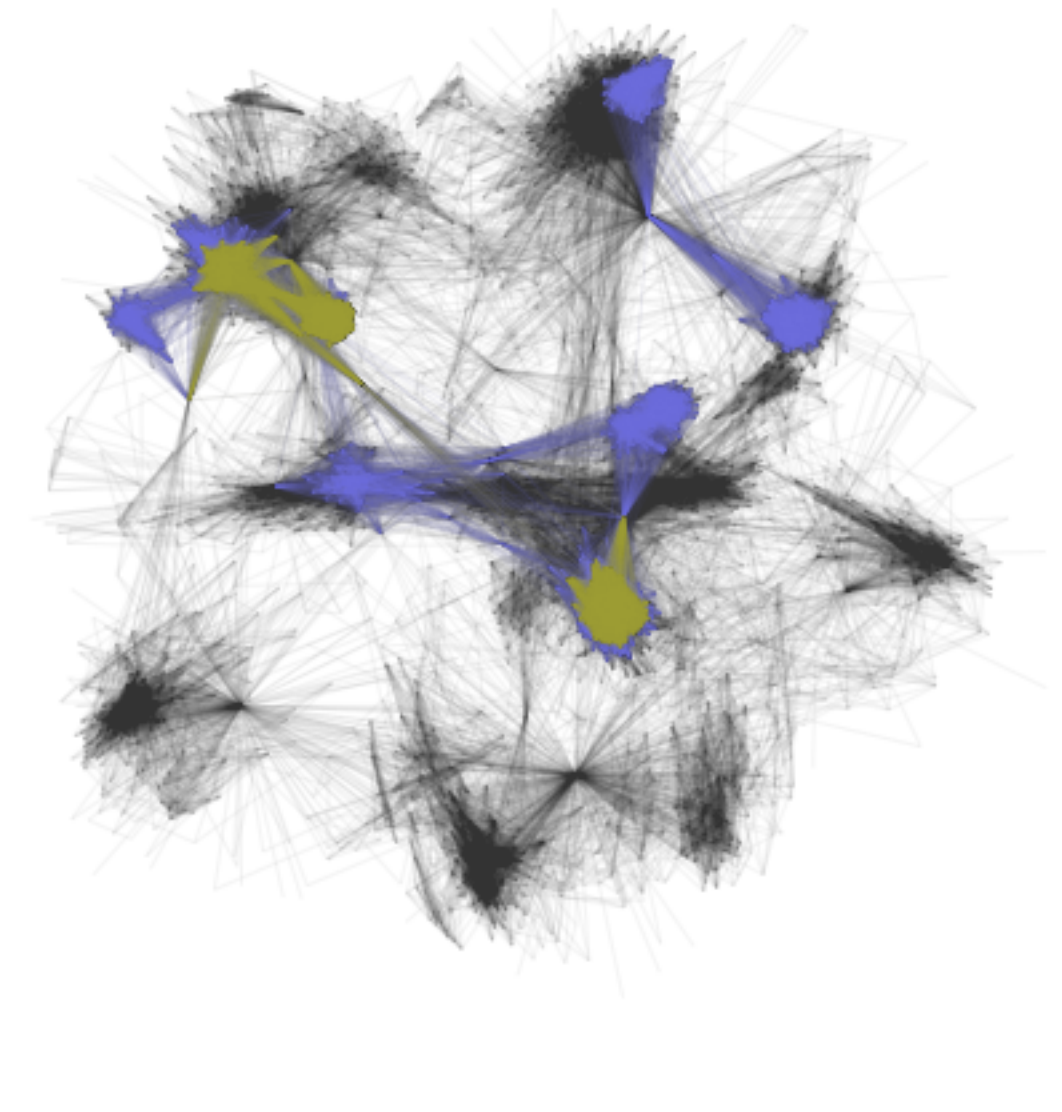
In [ ]:
```python
# Visualize maximum clique
node_color = [(0.5, 0.5, 0.5) for v in G_karate.nodes()]
for i, v in enumerate(G_karate.nodes()):
    if v in max_clique:
        node_color[i] = (0.5, 0.5, 0.9)
nx.draw_networkx(G_karate, node_color=node_color, pos=karate_pos)
```



K-Cores

In [ ]:
```python
# Find k-cores
G_core_30 = nx.k_core(G_social, 30)
G_core_60 = nx.k_core(G_social, 60)
```

In [ ]:
```python
# Visualize network and k-cores
nx.draw_networkx(
    G_social, pos=pos, node_size=0, edge_color="#333333", alpha=0.05, with_labels=False)
nx.draw_networkx(
    G_core_30, pos=pos, node_size=0, edge_color="#7F7FEF", alpha=0.05, with_labels=False)
nx.draw_networkx(
    G_core_60, pos=pos, node_size=0, edge_color="#AFAF33", alpha=0.05, with_labels=False)
```



In [ ]: