GRAPH ANALYTICS AND APPLICATIONS
ÉCOLE CENTRALE CASABLANCA

Lab 2: Community Detection

Instructor: Fragkiskos Malliaros

January 7, 2026

Description

The lab mainly introduces various well-known community detection methods, aiming to understand the mechanisms behind the approaches. During the lab we will be mainly working on toy example datasets (mainly the *Karate* network). Of course, this is only for demonstration purposes. The same algorithms could be applied in the *NetSci* collaboration network used in the first lab, or any other network.

Part I: Girvan–Newman Algorithm

Centrality measures can play a significant role towards detecting important nodes in the network; we have already studied this topic during the previous lab. *Betweenness* centrality is a measure relying on the shortest paths; it is basically defined for an edge as the number of shortest paths that pass through the edge [2].

If a network consists of communities of different sizes in which the nodes are more likely to be connected to each other within the same community, then the edges connecting different communities might have higher edge betweenness values because of the fact that any shortest path between nodes belonging to different communities has to go along one of these edges.

Girvan–Newman's algorithm extracts communities by recursively removing edges from the network based on betweenness centrality; at every step, the algorithm chooses the edge having the highest betweenness value. The basic steps of the algorithm [1] are summarized as follows:

1. Calculate the betweenness centrality for all edges in the network.
2. Remove the edge with the highest betweenness.
3. Recalculate betweennesses for all edges affected by the removal.
4. Repeat from 2 until no edges remain in the graph.

Exercise 1: Implementation of Girvan-Newman Algorithm

1. In the first exercise, we will implement Girvan-Newman's algorithm by completing the following function.

```
def girvan_newman_algorithm(G):
    '''
```

```
    :param G: input graph
    :return partitions: a list tuples of sets of nodes in the graph.
    '''
    g = G.copy()

    ...
    ...

    return partitions
```

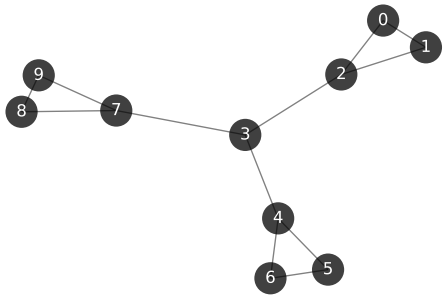

Figure 1: A toy example graph.

- The built-in `edge_betweenness_centrality()` method of NetworkX package can be used to compute the betweenness centrality values of edges.
- **Example:** You can use `3net.gml` file to load the toy network shown in Figure 1. `girman_newman_algorithm(G)` method should similarly return the following list of tuples:

```
partitions = girvan_newman_algorithm(G)
for partition in partitions:
    print(partition)

({0, 1, 2}, {3, 4, 5, 6, 7, 8, 9})
({0, 1, 2}, {8, 9, 3, 7}, {4, 5, 6})
({0, 1, 2}, {3}, {4, 5, 6}, {8, 9, 7})
({0}, {1, 2}, {3}, {4, 5, 6}, {8, 9, 7})
({0}, {1}, {2}, {3}, {4, 5, 6}, {8, 9, 7})
({0}, {1}, {2}, {3}, {4}, {5, 6}, {8, 9, 7})
({0}, {1}, {2}, {3}, {4}, {5}, {6}, {8, 9, 7})
({0}, {1}, {2}, {3}, {4}, {5}, {6}, {7}, {8, 9})
({0}, {1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}, {9})
```

2. **Dendrogram.** `plot_dendrogram(G, partitions)` method has been implemented for you to depict the extracted community structure as a dendrogram. Plot the dendrogram of the network using the output provided by Girvan-Newman algorithm.

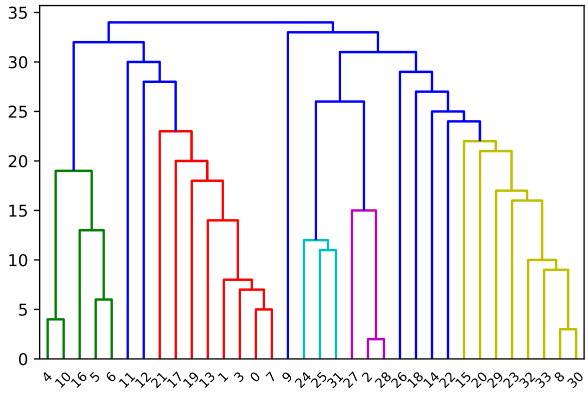

Figure 2: An example dendrogram for Karate network.

Exercise 2: Implementation of Modularity

Modularity is a measure mostly used to evaluate the community structure of a network. Modularity can provide us an intuition about the structure of the network — with higher values indicating better community structure. More formally, for a division of the graph into two groups, modularity can be defined in the following way [3]:

$$Q := \frac{1}{2m} \sum_{v,u \in \mathcal{V} \times \mathcal{V}} \left[A_{v,u} - \frac{k_v k_u}{2m} \right] \frac{s_v s_u - 1}{2m} \tag{1}$$

where m is the number of edges, $\mathcal{V}$ is the vertex set of the graph, A is the adjacency matrix, k_v is degree of node v and s_v is equal to 1 if v is a member of the first group otherwise -1.

1. Implement the modularity function by completing the `compute_modularity(G, partition)` method, which takes as input a graph object and a tuple of sets as parameters.

```
def compute_modularity(G, partition):
    '''
    :param G: given graph
    :param partition: a tuple of node sets.
    :return result: modularity value for a partition of the network.
    '''
    ...
    ...

    return result
```

   Example: For the partition `({0, 1, 2}, {8, 9, 3, 7}, {4, 5, 6})` of the toy network in Figure 1, you should obtain a modularity value, Q, close to 0.4896.

2. Find the communities of *Karate* network in which the modularity value is maximized.

Part II: Spectral Clustering

Spectral Clustering is very well-known approach to detect clusters in the network. For more details, please look at the paper [4]. In this last part, we will implement spectral clustering to extract communities and we will compare the results obtained to those returned by the Girvan-Newman algorithm.

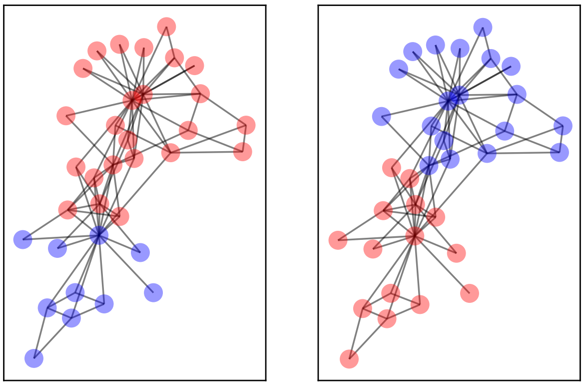

Figure 3: Two different partitionings of Karate network.

Exercise 1: Implementation of Spectral Clustering

1. Implement the spectral clustering algorithm by filling the following method:

```
def spectral_clustering(G, k):
'''
:param G: given graph
:param k: the number of clusters
:return partitions: a list of node sets where each node set
:indicates a cluster
'''
...
...
return partition
```

2. Split the *Karate* network into two communities using spectral clustering and Girvan–Newman algorithms. Which one gives the best clustering result?

3. You can use the `visualize(G, partition1, partition2)` function to visualize two different partitioning of the network side by side. You can see an example in Fig. 3.

References

[1] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[2] LongJason Lu and Minlu Zhang. *Edge Betweenness Centrality*, pages 647–648. Springer New York, New York, NY, 2013.

[3] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 103(23):8577–8582, Jun 2006. 16723398[pmid].

[4] Ulrike von Luxburg. A tutorial on spectral clustering, 2007.