Note: I did this lab assignment by working with Kiante. Therefore, the results and the codes are near.

Code explanation

The first thing that was obstacle in this lab was the binary files. Binary files were not running on macOS, so we had to re-compile them to get their outputs. This was done by cloning GKlib, METIS, snap, and mlrmcl1.2 and compiling them using Terminal.

To get the outputs from the CNM algorithm for the YouTube dataset, I tried deleting nodes less than a threshold, which I tried deleting nodes with a degree less than 2 to 15, but the problem would remain, and the runtime was not changing much.

On the other hand, we could get the outputs for the other algorithms, and we changed the outputs' formats so we could use them for NetworkX pre-defined functions. These functions are in the Code Listing 1.

Code Listing 1: Functions for output formats

```
#A list of frozensets of nodes, one for each community. Sorted by length with largest communities first.
def convert_communities(G, communities):
    communities_temp = communities.copy()
    cres = []
    temp = communities.value_counts()

    communities_temp['Node_id'] = G.nodes #for the output fromt the gpmetis | mlrmcl
    # if method == 1:
    #     communities_temp['Node_id'] = G.nodes #for the output fromt the gpmetis | mlrmcl
    # else:
    #     communities_temp['Node_id'] = communities.index.values #for the CNM

    for com in np.concatenate(np.array(temp.index.values)):
        com_temp = [communities_temp[communities_temp['Community'] == com ]['Node_id'].values]
        fnum = frozenset(set(com_temp[0]))
        cres.append(fnum)
    return cres
```

Results and comparisons

After getting the desired format for the outputs, we used the networkx.algorithm.community library to compute the modularity of each output. In figure 1, we can see the bar plot for each dataset, except the YouTube dataset, and in figure 2, we can see two plots from the first lab assignment.

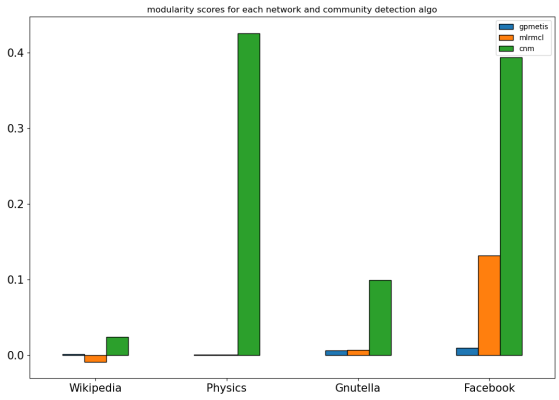

Figure 1: Modularity scores computed by different algorithms for each dataset.

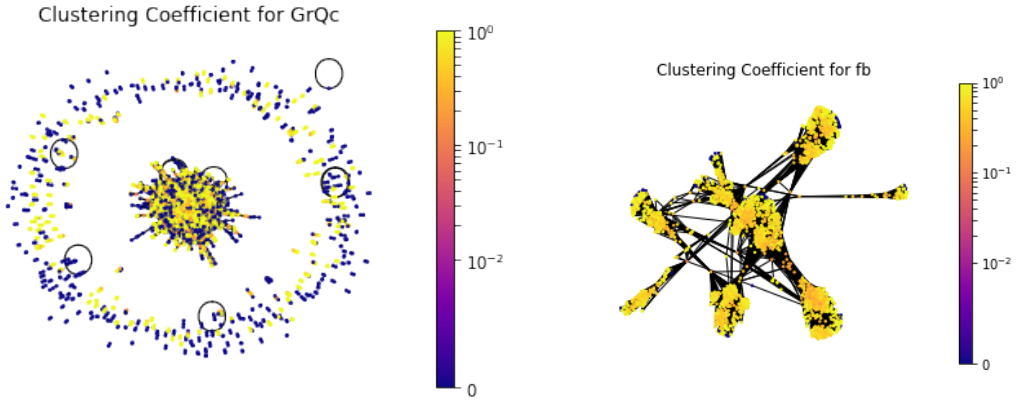

Figure 2: Heatmap for degree centrality and an approximate visualization of the network for Physics (left) and facebook (right).

A high modularity score in a network means that the network can be divided into distinct modules or communities, with nodes within each module being more densely connected than nodes in other modules. In figure 2, we expect our algorithms to have high modularity scores in facebook and physics datasets. This is the case for the CNM algorithm, but we can't say they are successful for MLR-MCL and gpmetis.

Furthermore, by changing the -c 500 for MLR-MCL to -c 1000 and -c 10000 we got different results that are in figure 3.

As we can see, the score for facebook dataset grows but almost stays the same for

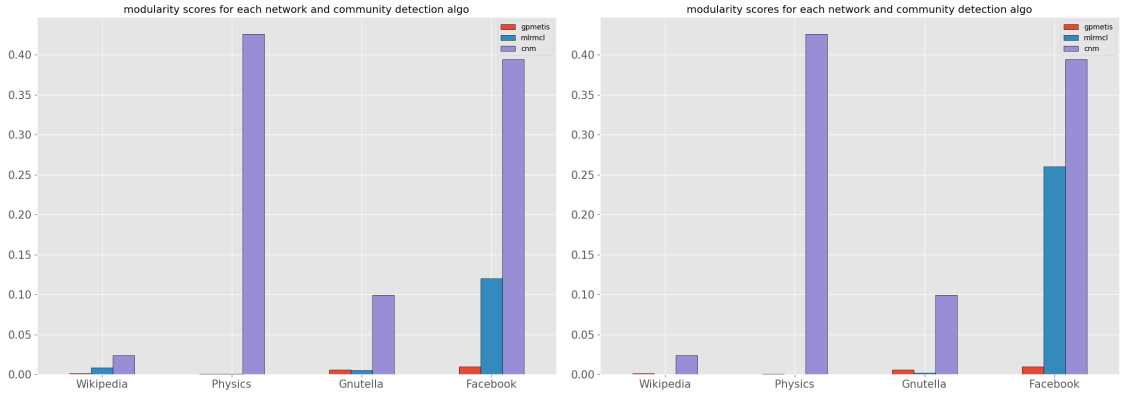

Figure 3: Changing parameters on MLR-MCL to -c 1000 (left) and -c 10000 (right).

other datasets.

References

[1] Easley, David; Kleinberg Jon. *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*. Cambridge University Press, 2010. Print.

[2] "Stanford Network Analysis Project". SNAP, Stanford. Online. `http://snap.stanford.edu/index.html`

[3] "METIS". Online. `https://github.com/KarypisLab/METIS`

[4] "GKlib". Online. `https://github.com/KarypisLab/GKlib`

[5] "Network Analysis in Python". NetworkX. Online. `https://networkx.org/documentation/stable/reference/introduction.html`