

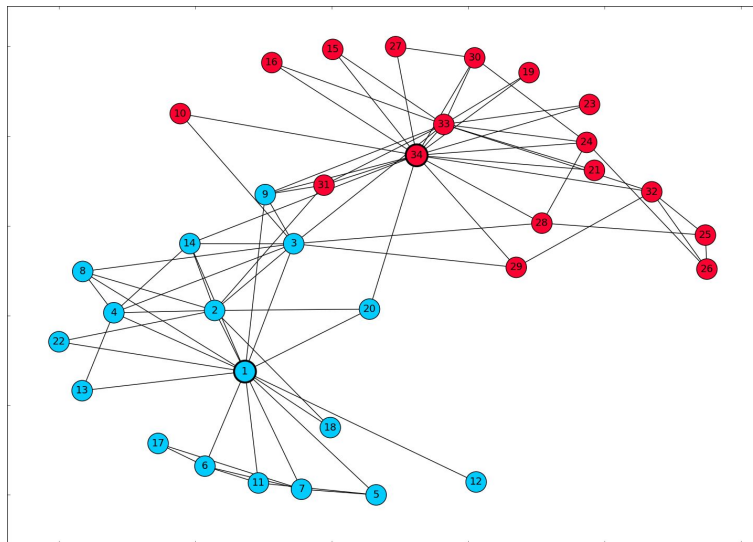
Network Exploration with Linear Algebra

Project Scope and Goals

- The theory side of project:
 - Investigates different spectral partitioning and clustering methods based on the spectral decomposition of the Laplacian
 - Examines the performance of these methods in a few random graph models, such as the Stochastic Block Model and the Planted Partition Model
- The application side of the project:
 - Implements these methods in Python on top of the NetworkX package
 - Upload this implementation as a package to a repository such as PyPi

Motivating Question

- One question of interest is to find an effective partition of a graph
 - “Effective” means that nodes in one part of the graph are more similar to one another than to nodes in other parts of the graph

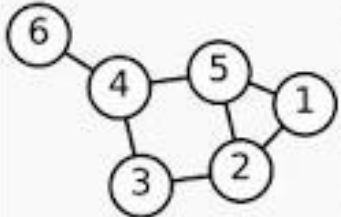


The Laplacian

- The Combinatorial Laplacian L is defined as:

$$L = D - W$$

where D is the $n \times n$ degree matrix and W is the adjacency matrix.

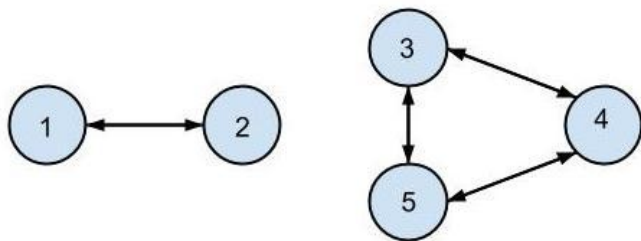
Labeled graph	Laplacian matrix
	$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$

Motivation for Spectral Methods

- For a graph with K connected components, the null space of L is

$$\text{null}(L) = \text{span}(\mathbf{1}_k) \quad \text{for } k = 1, \dots, K$$

- Thus, the null space of L tells us which nodes are in which connected components.



$$\begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Criterion to Bi-Partition

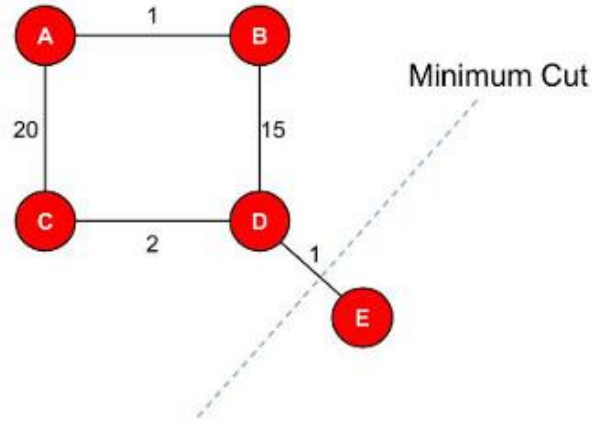
- For any subset A of the vertex set V , the **cut of A** is defined as:

$$\text{cut}(A, A^c) = \sum_{u \in A, v \in A^c} w(u, v)$$

- If W is the adjacency matrix, then the cut of A is the number of edges between A and A^c that need to be removed in order to form the partition $\{A, A^c\}$.

Why not use minimum cut?

- A minimum cut criterion tends to cut isolated nodes in the periphery of the graph.



What to use instead?

Shi and Malik proposed the minimum Normalized Cut,

$$NCut(A, A^c) = \frac{cut(A, A^c)}{assoc(A, V)} + \frac{cut(A, A^c)}{assoc(A^c, V)}$$

Where

$$assoc(A, V) = \sum_{u \in A, t \in V} w(u, t)$$

is the total number of connections from nodes in set A to all nodes in the graph.

Where does spectral clustering come in?

Shi and Malik showed that

$$\min_A NCut(A, A^c) = \min_y \frac{y^T L y}{y^T D y}$$

Under the constraints that (1) $y(i) \in \{1, -b\}$ and (2) $y^T D 1 = 0$

Solution to the Bi-Partitioning Problem

Because RHS of equation on the last page is the Rayleigh quotient, the real valued solution to our discrete problem is the eigenvector with second smallest eigenvalue of

$$Ly = \lambda Dy$$

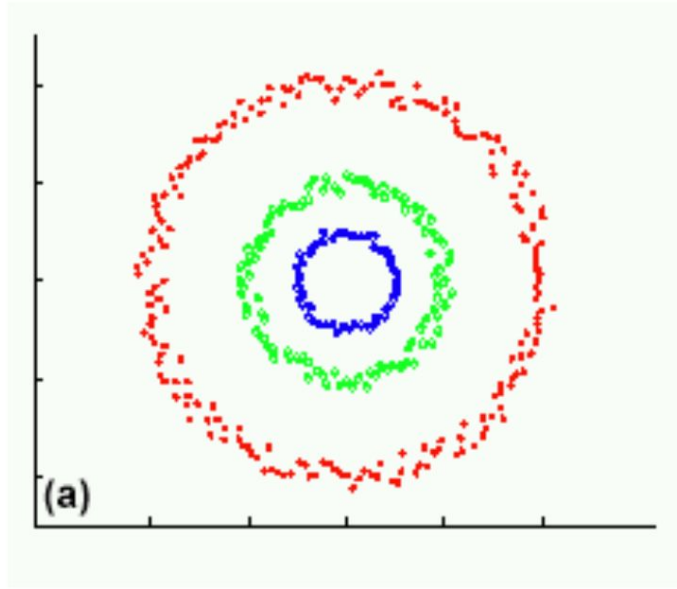
Examining the small eigenvectors of the Laplacian L can help find a “good” clustering of the original vertices.

What about splitting a graph K-ways?

1. Define a graph and calculate L
2. Find K smallest eigenvectors x_1, \dots, x_K of L
3. Form the matrix $X = [x_1, \dots, x_K]$
4. Form the matrix Y by normalizing each row of X to have length 1
5. Treat each row in Y as a point in \mathbf{R}^K and cluster using K-means
6. Assign the point S_i to cluster j if and only if row i of Y was assigned to the cluster j

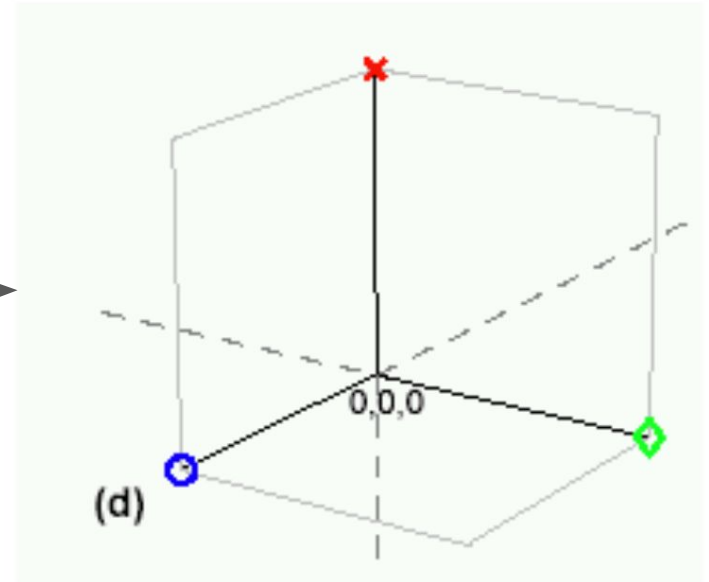
Visual Motivation (K = 3, Ideal Case)

Original data



n-Dimensional Similarity Space

Projected data



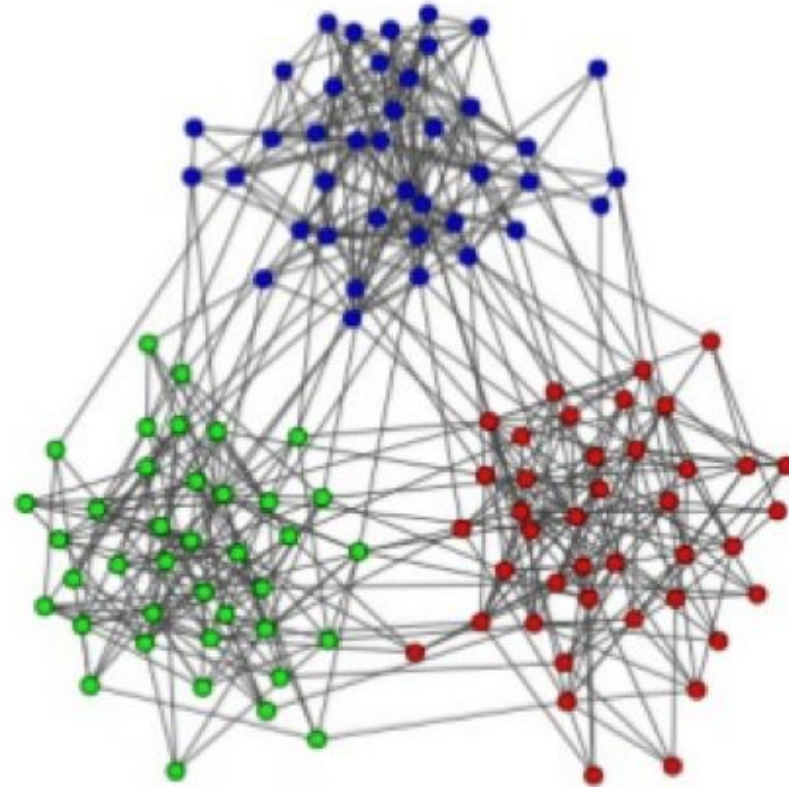
K-Dimensional Similarity Space

K-Way Spectral Clustering

- When there is only 1 connected component, each row is mapped to a unique point in \mathbf{R}^k
 - Grouping these points using a distortion minimizing algorithm such as K-means empirically finds good clusters
- Therefore, by looking at the small eigenvectors of L , we can determine a “good” partitioning of the data

Stochastic Block Model

- Detect smaller communities/clusters within a large network
- This may reveal hidden communities or reveal groups of nodes that frequently communicate
- The probability of nodes being connected within each cluster is the same, but the probability of internal connections varies for each community
- Nodes within a community are densely connected, while nodes between communities are loosely connected
- In other words, the probability of connection within communities is large compared to a small probability of connectivity between communities



Planted Partition Model

- The Planted Partition Model is a special case of the Stochastic Block Model
- The probability of connections within communities is still larger than the probability of connectivity between communities
- However, the probability of connectivity within all of the communities is constant
- Similarly, the probability of connectivity between communities is the same
- In other words, we are only concerned with 2 probabilities: the probability of a connection within a group and between groups
- Compared to the probability of connections within each community varying in the Stochastic Block Model

Hierarchical Clustering

Topmost cluster contains every point in the data set

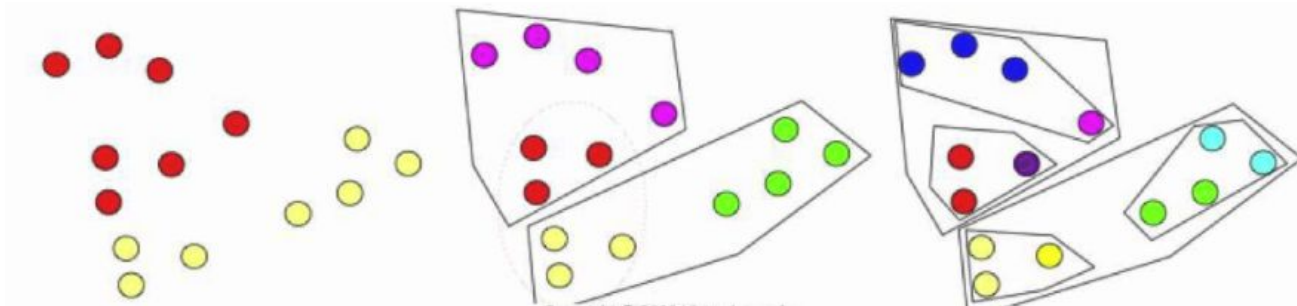
Bottom level contains set of n singleton clusters, one for each data point

Start with all the items in one cluster, then split recursively

Within each block, there is a planted partition model that is also hierarchical

Run the planted partition model on the the original data set, then recursively run the planted partition model on each of the clusters

This algorithm can tolerate noise that grows with the increasing number of data points



Application

- Concepts: Graph Theory, Python Classes & Objects, Partitioning Methods
- Goal: Create a package (Graph-Operator) to allow the users to generate graphs using different matrices, plot eigenvalues & eigenvectors and do graph partitioning
- Packages Used: Networkx, Scipy, Matplotlib and Scikit-Learn

NetworkX



matplotlib



Shi & Malik's method - Example

Networkx graph:

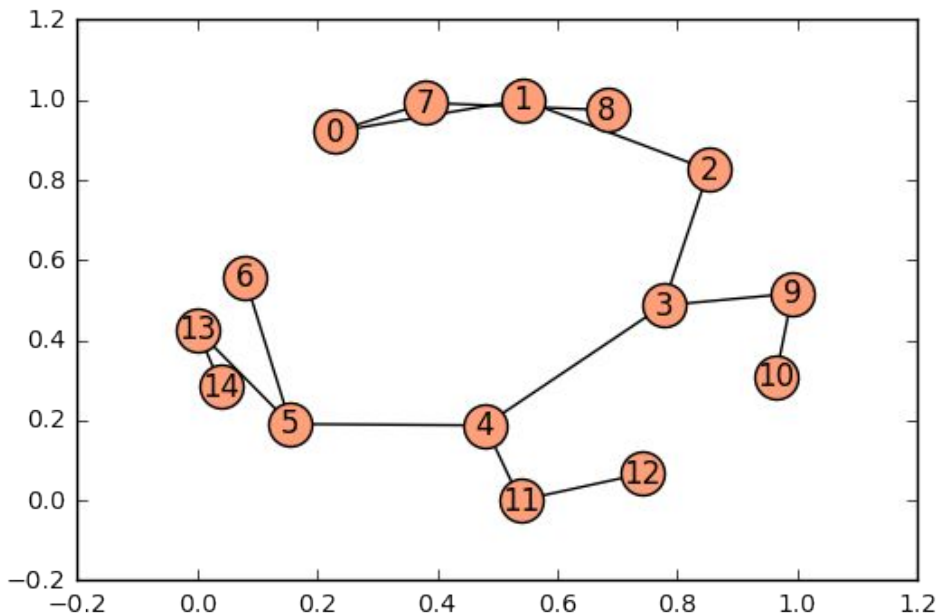
```
lob = nx.random_lobster(5, 0.6, 0.9, seed = 37)
```

Create a GraphOperator:

```
lobGraph = GraphOperator(lob, "combinatorial  
laplacian")
```

- Calculates Laplacian Matrix and its eigenvectors

Plot the graph:

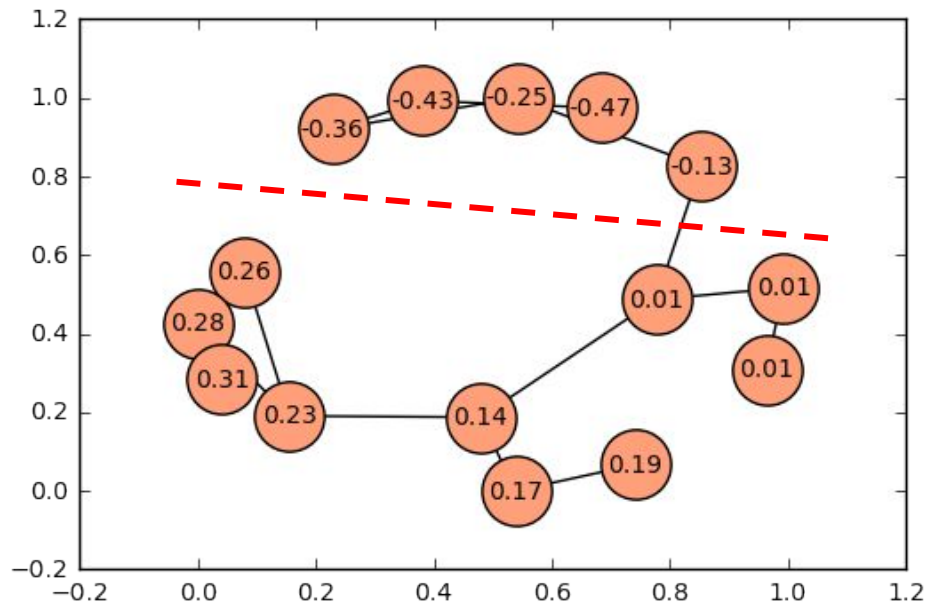


Second smallest eigenvector of Laplacian Matrix

The second smallest eigenvector:

```
lobGraph.eigvecs[:,1].round(5)  
>>> array([-0.35625, -0.25125, -0.12528,  
0.01114, 0.14427, 0.23488, 0.25626, -0.43153,  
-0.47081, 0.01349, 0.01472, 0.17476,  
0.19067, 0.28452, 0.31042])
```

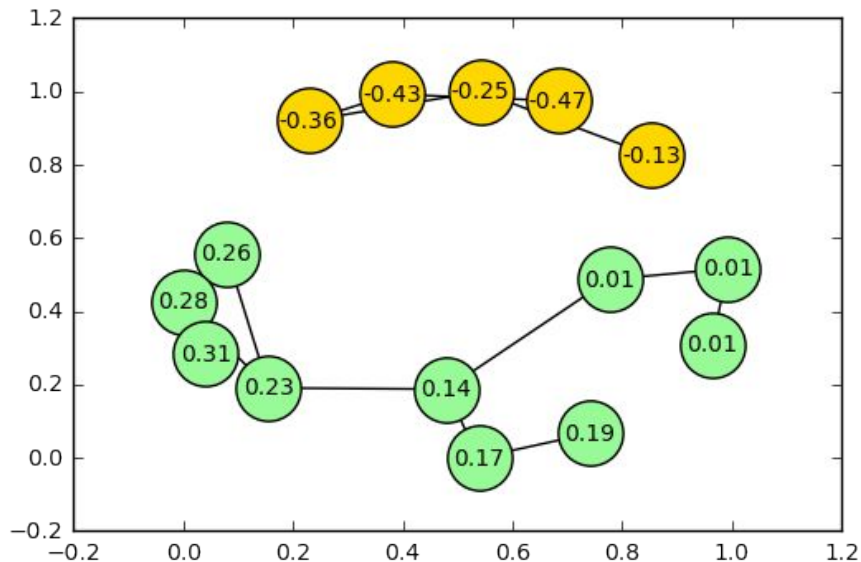
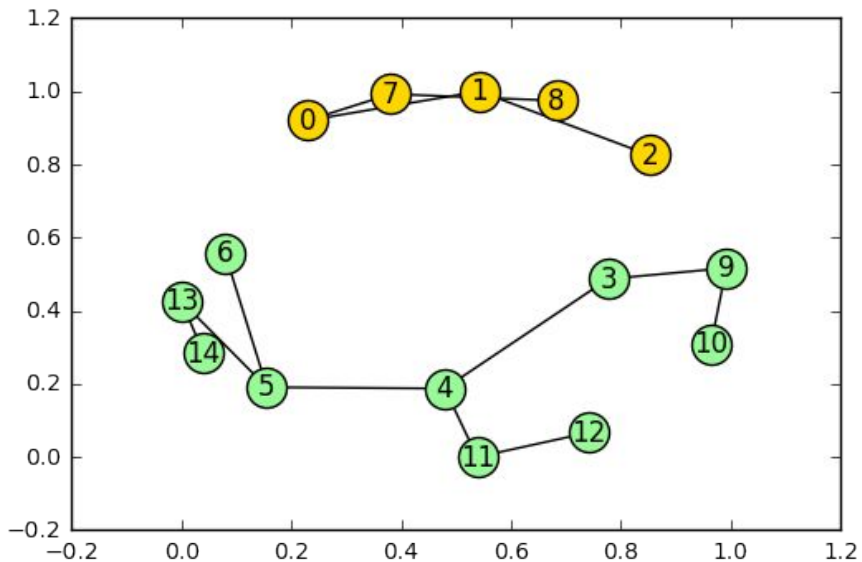
- Corresponds to vertex 0, 1, 2, 3, 4, ..., 12, 13, 14, respectively



Partition & Results

```
list_nodes = lobGraph._split_once(standardVal = 0)
```

```
lobGraph.draw_partitionGraph(list_nodes, removeEdges = True, list_color = ['palegreen', 'gold'],  
node_size = 300, alpha = 1)
```



Hierarchical Clustering Example

- Facebook Network Graph

```
g = nx.Graph()
```

```
g.number_of_nodes()
```

```
>>> 3959
```

```
g.number_of_edges()
```

```
>>> 84243
```

- Preprocess

```
graphs = list(nx.connected_component_subgraphs(g))
```

```
graphs.sort(key = lambda x : x.number_of_nodes(),  
reverse = True)
```

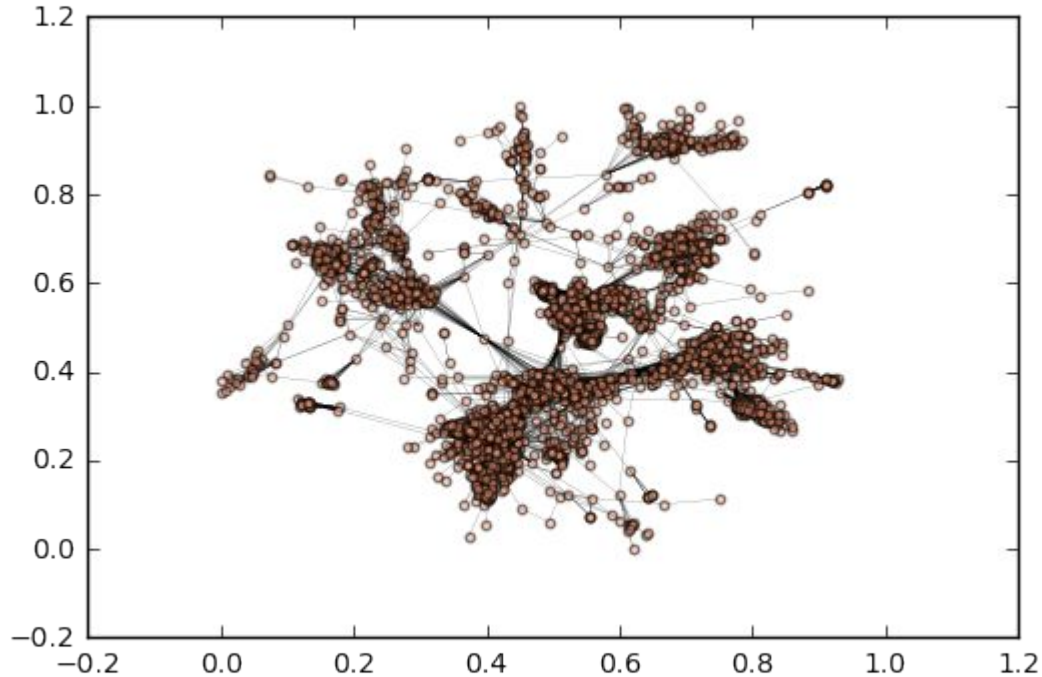
```
[x.number_of_nodes() for x in graphs]
```

```
>>> [3927, 6, 4, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2]
```

Visualization

```
g1 = GraphOperator(graphs[0], "normalized laplacian")
```

```
g1.draw_graph(labels = False, node_size = 10, width = 0.1)
```

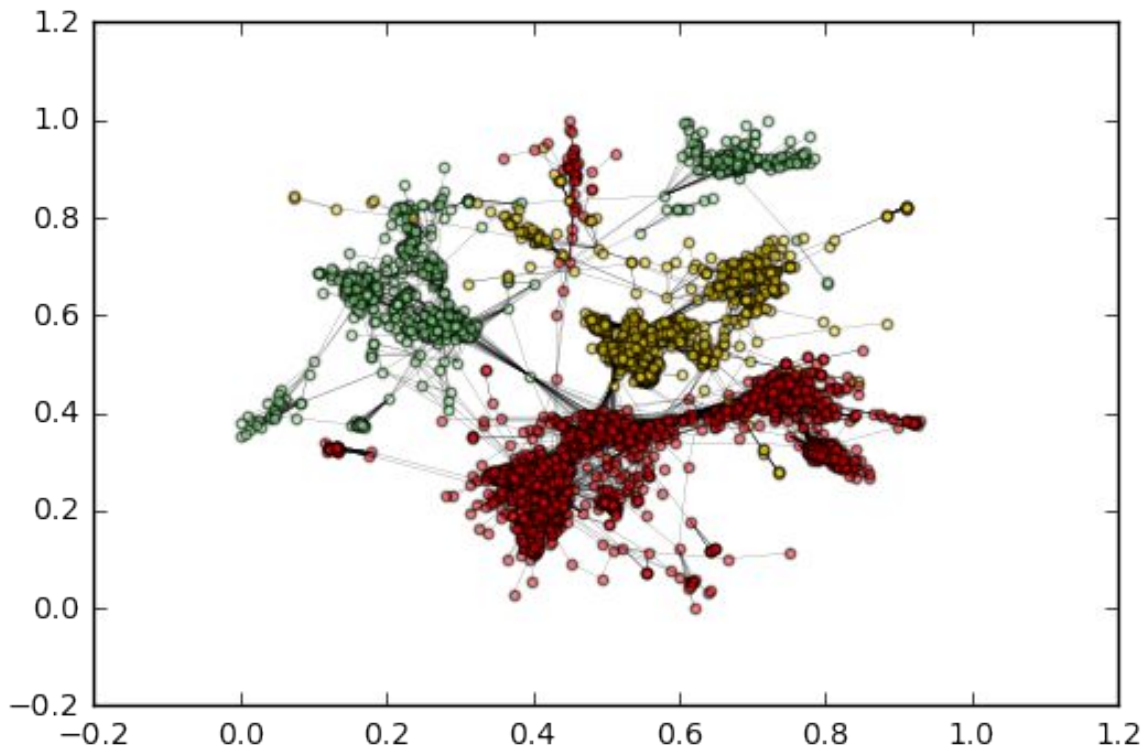
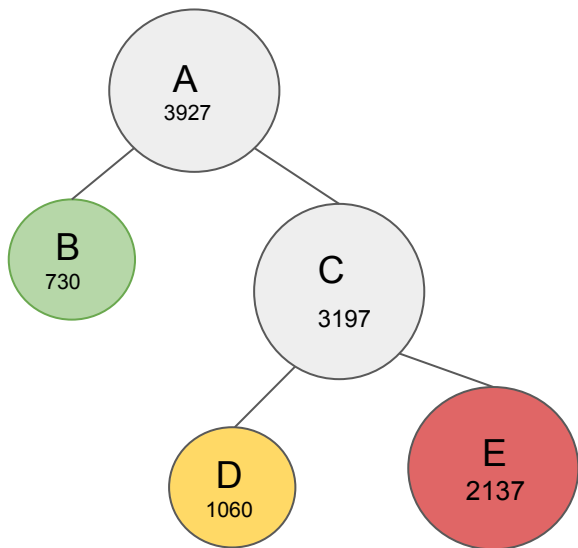


Hierarchical Clustering

```
list_subgraph_nodes = g1.partition(3)
```

```
[len(i) for i in list_subgraph_nodes]
```

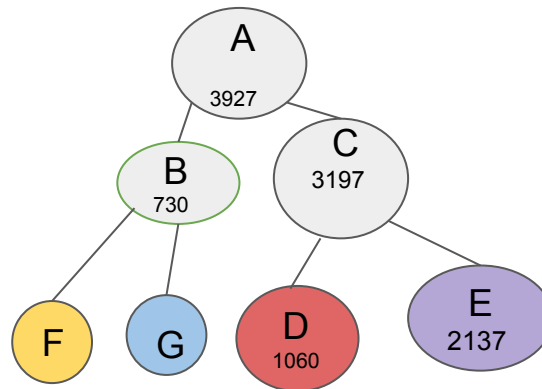
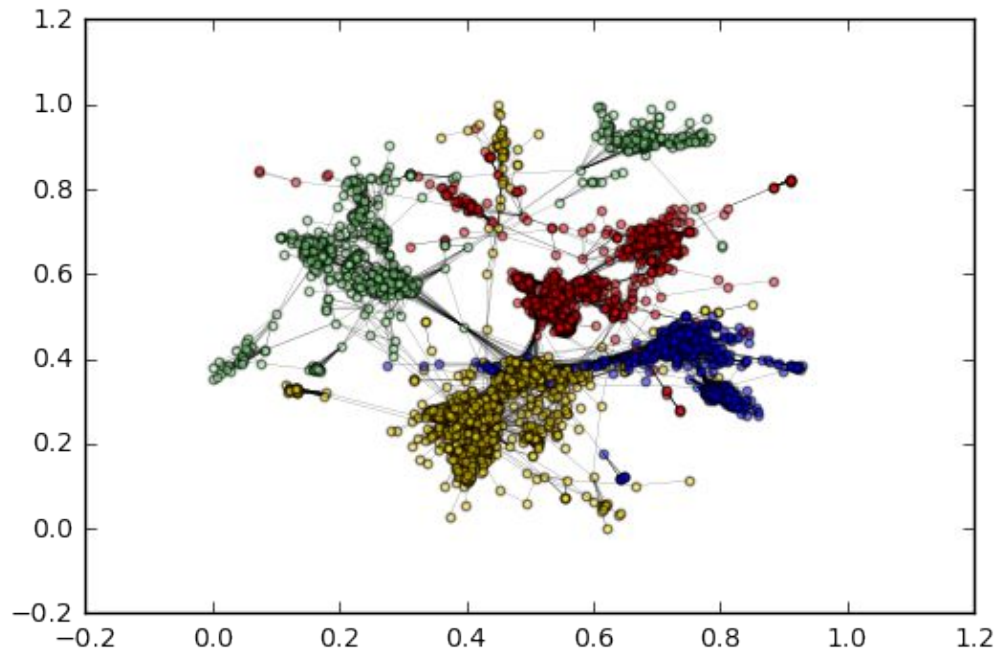
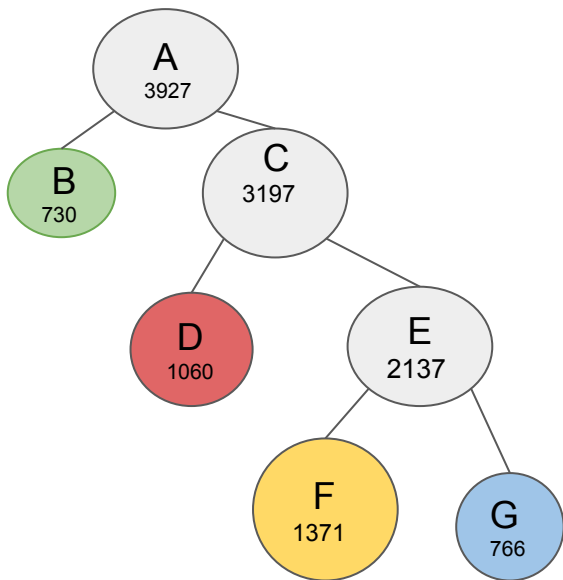
```
>>> [730, 1060, 2137]
```



Hierarchical Clustering

```
list_subgraph_nodes = g1.partition(4)  
[len(i) for i in list_subgraph_nodes]
```

```
>>>[1060, 730, 1371, 766]
```



K-Means Clustering

- Steps:
 - KMeans Function - Simultaneously separate the nodes into k groups using the k smallest eigenvectors of the Laplacian Matrix
- `KMeans(self, n_clusters, **kwargs)`
 - `KMeans_partition` Function - Example to the left
- `KMeans_partition(self, n_clusters)`

