Homework 3

Markov Clustering Algorithm

Sagar Dhamija (sagardha: 50169364)

Varun Khandelwal (varunkha: 50168936)

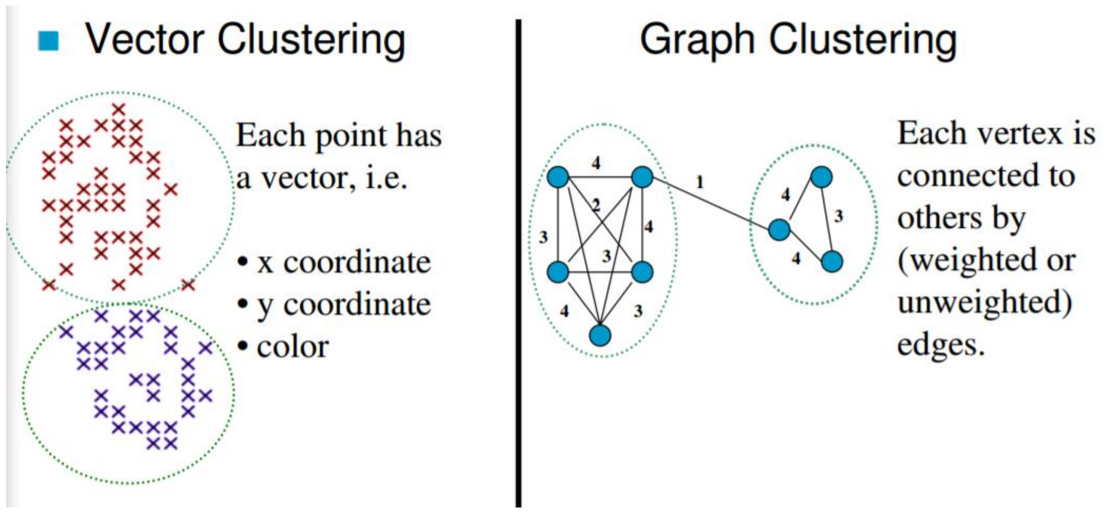
Contents

Introduction

Graph Clustering:

Clustering is of two types:

- Vector Clustering
- Graph Clustering


Vector Clustering
Each point has a vector, i.e.
• x coordinate
• y coordinate
• color
Graph Clustering
Each vertex is connected to others by (weighted or unweighted) edges.

For this report, we will just discuss graph clustering and in particular, Markov Clustering Algorithm.

The structures formed by a set of vertices called nodes and a set of edges that are connections between pairs of vertices are called **Graphs**.

The task of grouping the vertices together is called **Graph Clustering.**

Markov Clustering

Markov Clustering Algorithm, also known as **MCL** algorithm, is a fast and scalable unsupervised clustering algorithm for graphs/networks. It is based on the simulation of flow in graphs. The algorithm was developed in Netherlands by Stijn van Dongen at the Centre for Mathematics and Computer Science (also known as CWI) in the Netherlands.

The basic idea behind MCL algorithm is to find cluster structure in graphs. The probabilities of random walks through the graph are deterministically computed utilizing two operators transforming one set of probabilities into another. The random walk is performed by MCL algorithm by way of two operations, **Expansion and Inflation.**

Expansion can be explained as a process of computing random walks of higher lengths, which simply implies random walks with many steps. In this operation the new probabilities are associated with all pairs of nodes, where one node is the point of departure and other is destination. With higher lengths paths being more common within clusters than between different clusters, the probabilities associated with node pairs lying in the same cluster are relatively larger as there are many ways of going from one node to the other.

Inflation will then serve the purpose of boosting the probabilities of intra-cluster walks and will demote inter-cluster walks. This is achieved without any a priori knowledge of cluster structure and is simply the result of cluster structure being present.

Thus in order to separate the graph into different segments we can iterate over different results obtained by using various expansion and inflation coefficients. Once done iterating over expansion and inflation sufficient number of times there will no longer be any paths between these segments and the collection of resulting segments is simply interpreted as clustering. It is being observed that while keeping the expansion operator constant, increasing the inflation operators increases the granularity or tightness of clusters.

The PhD thesis *Graph clustering by flow simulation* is fueled by this algorithm, the main topics being the mathematical theory behind it, its position in cluster analysis and graph clustering, issues concerning scalability, implementation, and benchmarking, and performance criteria for graph clustering in general.

Advantages

1. It is not misled by edges leading to other clusters.
2. It is very fast and scalable.
3. It has a natural parameter(inflation) for influencing cluster granularity.

Algorithm

1. Input is an un-directed graph, power parameter *e*, and inflation parameter *r*.
2. Create the associated matrix
3. Add self-loops to each node (optional but recommended)
4. Normalize the matrix
5. Expand by taking the *e*th power of the matrix
6. Inflate by taking inflation of the resulting matrix with parameter *r*
7. Repeat steps 5 and 6 until a steady state is reached(convergence)
8. Interpret resulting matrix to discover clusters

Our Implementation

Main code snippet:

```
#Iteration counter
i=0
#Keep looping till convergence or till 1000 iterations whichever is earlier
while(not (old==new).all() and i<=1000):
    #backup present matrix
    old = new
    #Expansion
    new = np.linalg.matrix_power(new, n_power)
    #Inflation and normalization
    new = np.power(new, n_inflation)
    new = np.divide(new,np.sum(new, axis=0))
    #Pruning - Optional
    #new = new.round(3)
    #Increment loop counter
    i = i + 1
```

1. We are using python and its numpy and pandas libraries.
2. n_file, n_file_net, n_power and n_inflation are the parameters that can be changed.
3. Next we read the file using pandas and create an adjacency matrix.
4. Next we introduce self-loops and then normalize the matrix.
5. Next is the piece of code pasted above which performs alternative expansion and inflation till convergence or 1000 iteration whichever happens earlier.
6. Finally, we find rows that have non-zero elements and every non-zero element in that row belongs to the same cluster.
7. We then output the clu file for Pajek and check the visualization in Pajek.

Deciding on parameters e and r

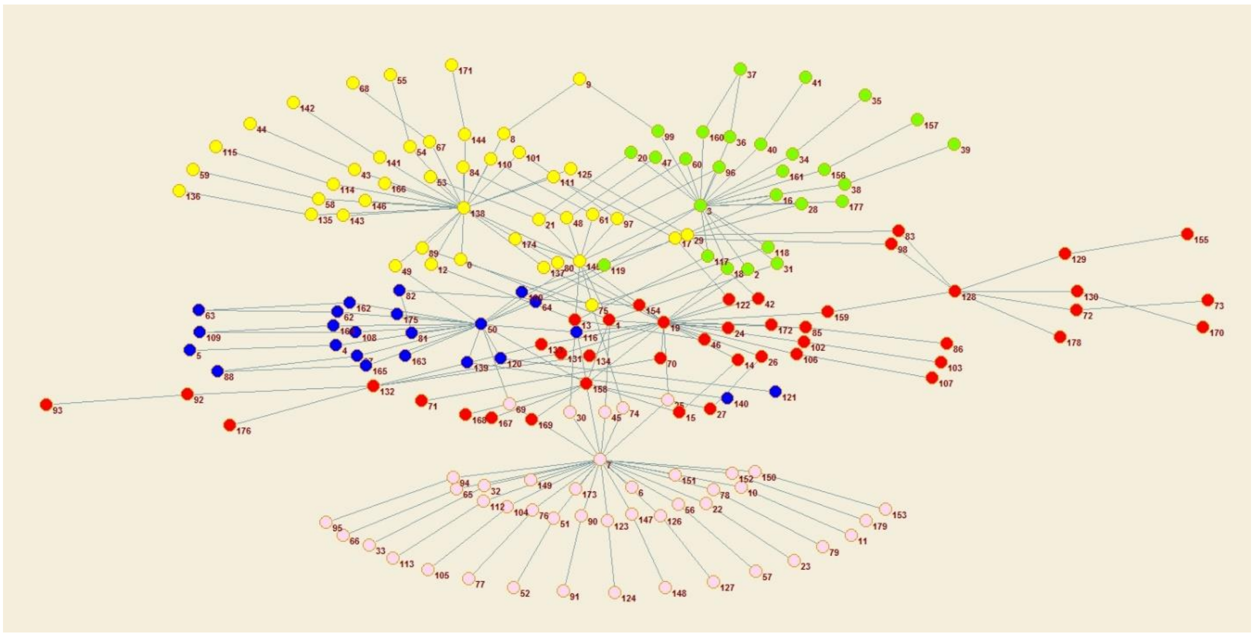
We fixed e at 2 as per the paper Van Dongen, Stijn Marinus. "Graph clustering by flow simulation." (2001) and varied r between 1 and 2. We then visualized the results using Pajek and selected r based on the best visualization.

Results

ATT

e = 2

r = 1.35

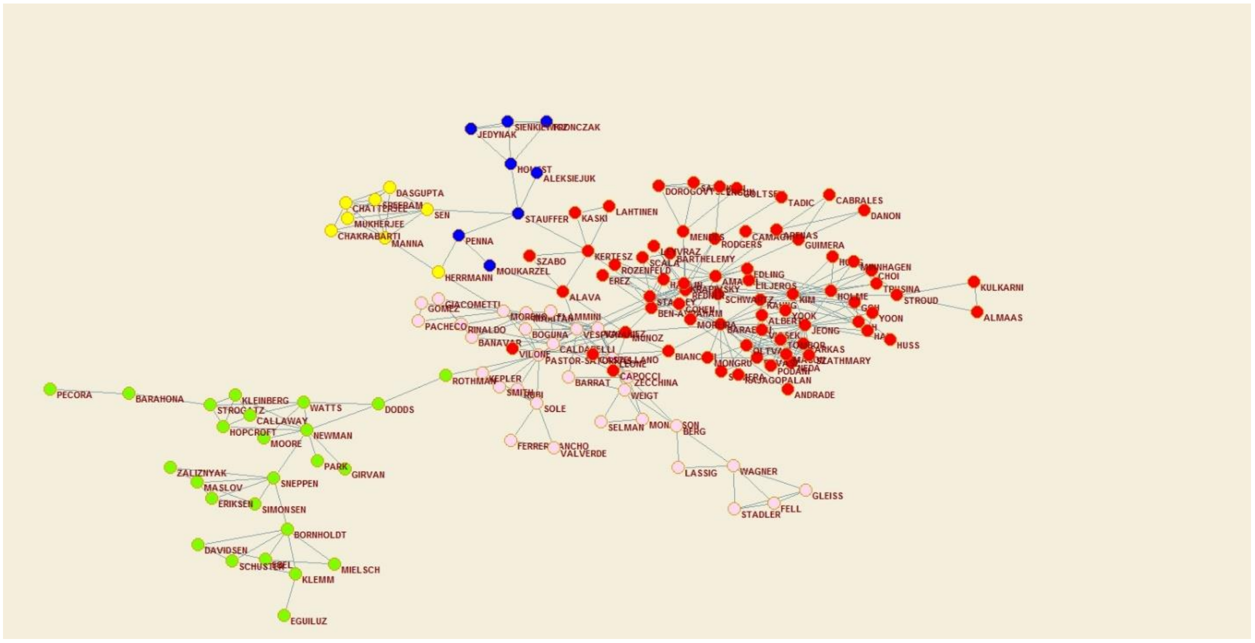


Physics

As the algorithm is unsupervised, we can get many results. We tried matching with the ground truth result shown by the corresponding image and parameters but we felt by changing the parameters ‘e’ and ‘r’ another result looked much better than the ground truth match.

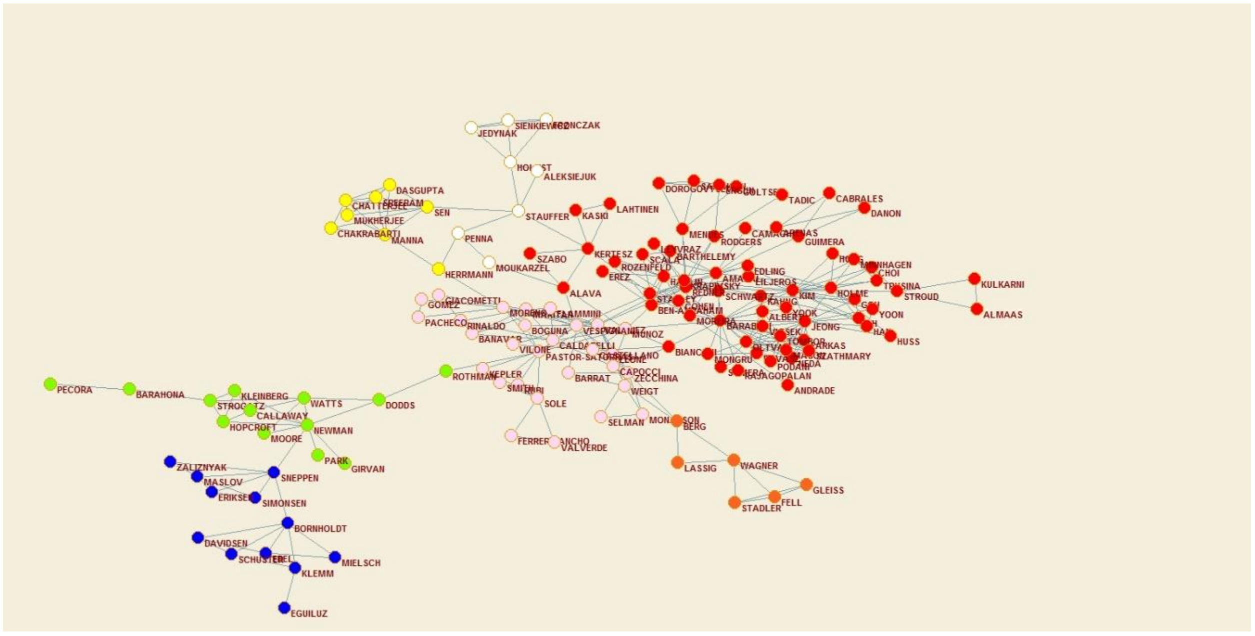
Ground truth matchup:

e = 2 r = 1.27



However, for the following parameters we felt the results obtained were somewhat better:

e = 2 r = 1.3

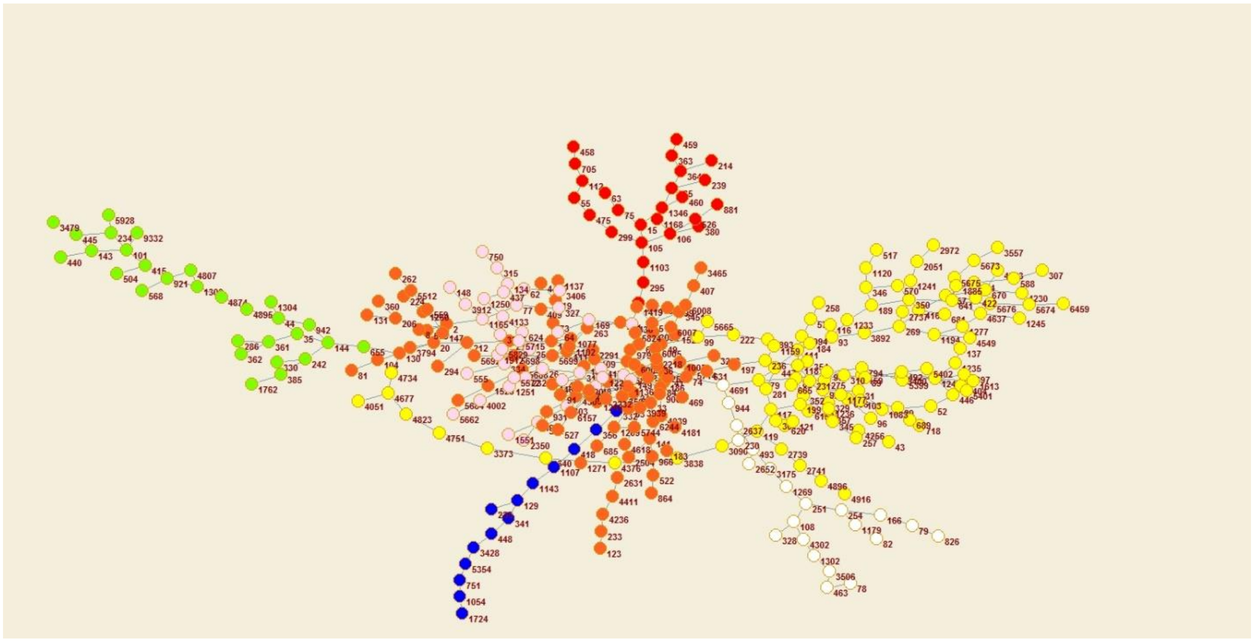


Yeast

Similar to the physics dataset, we found 2 results for this dataset: one that matches the ground truth and the other that we felt looked better visually.

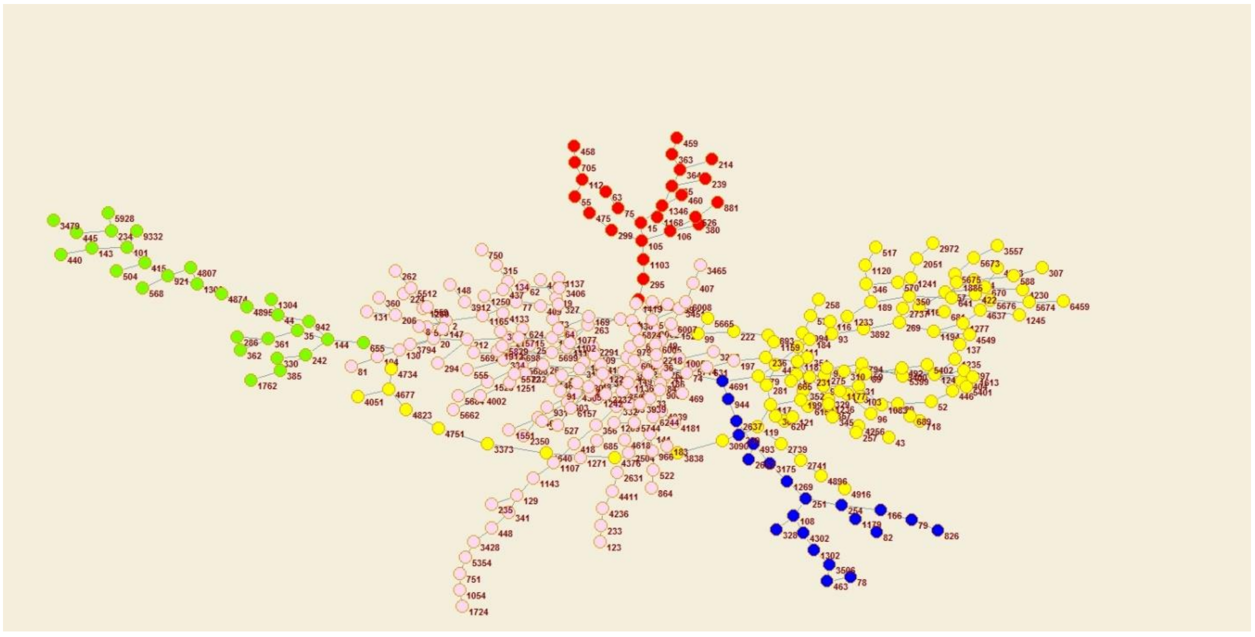
Ground truth matchup:

e = 2 r = 1.205



However, for the following parameters we felt the results obtained were somewhat better:

e = 2 r = 1.2



References

1. Van Dongen, Stijn Marinus. "Graph clustering by flow simulation." (2001).
2. http://micans.org/mcl/
3. Lecture Slides (https://www.cs.ucsb.edu/~xyan/classes/CS595D-2009winter/MCL_Presentation2.pdf)