HW3: Markov Clustering Algorithms

Group #: 4

Jagvir Singh (50134824), Shahid Fazal (50133053), Vidyadhar Reddy (50134358)

MCL algorithm Description:

MCL Algorithm is an unsupervised algorithm for *hard clustering* datasets that can be typically represented in graph format. It is based on probabilistic reachability of nodes from a particular node in the unidirectional-edged network. In other words, the probability that a random walk will take an edge at node *p* depends on *p* and the given edge and is independent of edges previously traversed.

It is considered to be a form of **hard clustering** because each data point can be categorized in only one of the clusters, unlike in soft clustering where a point can belong to multiple clusters.

When it works:

MCL is best suited for clustering of datasets where the data can be represented in the form of a network or graph with edges between set of vertices. The edges may be unidirectional or bidirectional (unidirectional). The edges may further be weighted or un-weighted. The algorithm feeds on the dissimilarity of clusters and amplifies the difference to a significance value.

Advantages:

a. We do not have to know how many clusters to categorize the data into.
b. MCL scales well for large datasets – in linear proportion with number of vertices in graph
c. Pruning quickens the convergence of matrix within acceptable levels of deviation
d. Not sensitive to outliers

Disadvantages:

a. With no pruning, it might take long to converge; sometimes, it might not even converge.
b. Hard to find optimal parameters for clustering – changing the power and inflation parameters yields different-sized clusters but no way to assure quality of clustering unless we ***visualize*** results, which needs tools like Pajek

2) Pseudo code for **implementation**.

- Load data files and find maximum number of vertices in the graph.
- Create a *vertices*vertices* adjacency matrix initialized by all 0 values.
- Populate the adjacency matrix:
  - For each line of data: “p q” in the file, matrix[p][q] = 1
  - For each line of data: “p q” in the file, matrix[q][p] = 1
  - Set all diagonal elements as 1 (adjMatrix + Identity Matrix)
- Expand the matrix (A+I) to the e^{th} power (configurable by user)
- Normalize the matrix
- Prune matrix

    - Round off values > 0.99 to 1
    - Round off values < 0.01 to 0
- Check for convergence
- If converged, identify clusters and their sizes from the matrix
- If not, repeat expansion, inflation and pruning until convergence

3) Briefly describe how you implemented the algorithms using your preferred language, e.g., pattern design, data structures.
    - Implementation:
        - Psudeo code, and working of algorithms is explained above.
        - Code repository and stepwise commits: https://github.com/jagvirsingh5/Marchov_clustering_algorithm
        - All user has to do is run the program and select the input file(choice driven), pass expansion parameter(**integer**), and the inflation parameter(**double**)
        - Output:
            - Number of clusters generated for the dataset
            - Modularity of graph for the given parameters
        - Data structures used: We have extensively used the ***Matrix*** class from the JAMA Java library for all the steps after creation of adjacency matrix – expansion, inflation, normalization.
            - **Matrix1.times(Matrix2)** function – multiplication of Matrix objects
    - JAMA java library: We have used in this project JAMA java library, which is a basic linear algebra package for Java. It provides user-level classes for constructing and manipulating real, dense matrices. It is meant to provide sufficient functionality for routine problems, packaged in a way that is understandable to non-experts.
      [Reference: http://math.nist.gov/javanumerics/jama/]

    - Cluster identification: Once the natural and matrix converges after repeated expansion and inflation, for each row in the matrix, we identify the non-zero elements in a separate list. We did not have to explicitly handle overlapping clusters.
    - CLU file generation: Once the clusters are generated, we map each vertex to its cluster and write the new file using ***printWriter.***

Results:

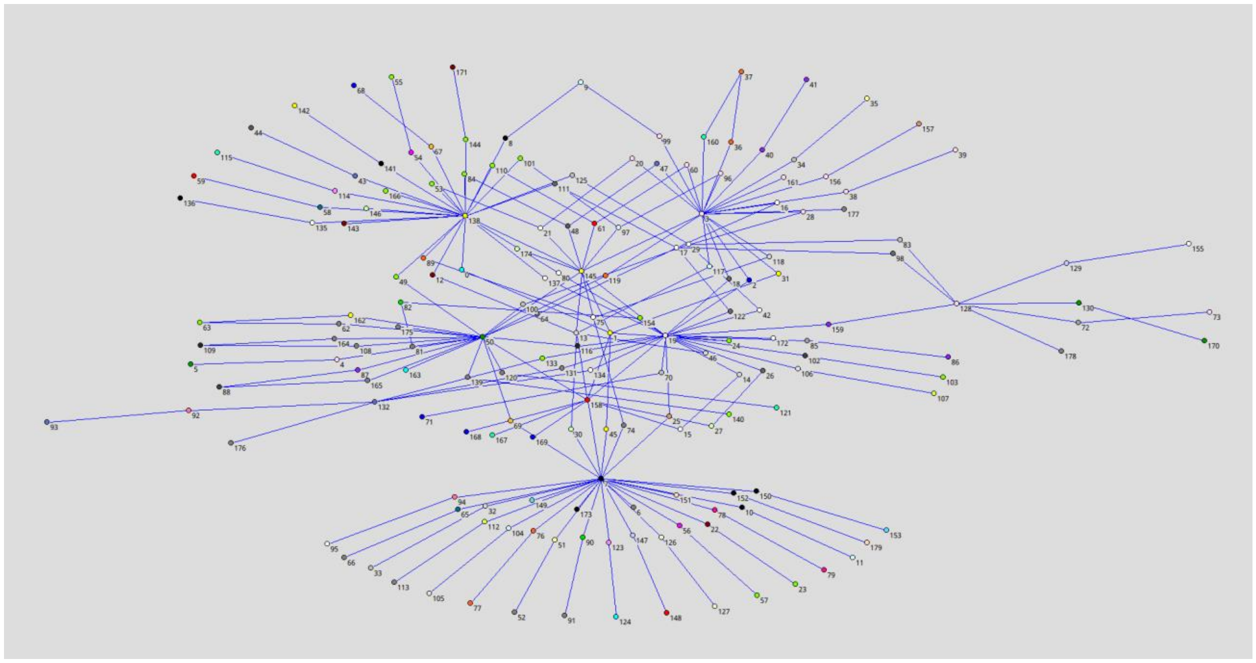
There are majorly two tuning parameters in MCL algorithms: inflation rate r, and expansion rate m, (both are default 2). Tune the two parameters, visualize clustering result using Pajek or other software.

1) Calculate modularity for each parametric configuration (inflation, expansion). The modularity is defined Equation (45) on Page 44,
   You should choose a reasonable range of parameters **based on your own experiment**. For example, $r = \{1.1, 1.3, 1.5, 1.7, 1.9, 2\}$.
   The parameters could be different for each dataset. Some parameters may not even converge.

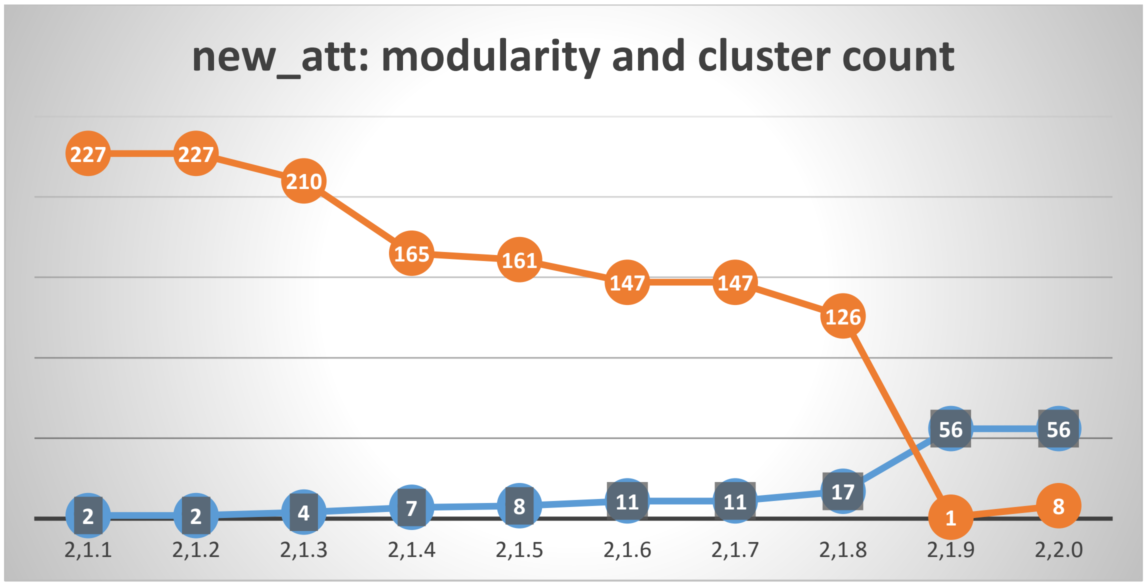
Following is the representation of variation of number of clusters against e,r parameters.

Input Parameters			**Number of Clusters**			
(e,r)	new_att	Mod_att	new_collab	Mod_collab	new_yeast	Mod_yeast
2,1.1	2	227	2	338	2	433
2,1.2	2	227	4	317	7	395
2,1.3	4	210	8	256	22	305
2,1.4	7	165	12	215	38	237
2,1.5	8	161	15	154	53	181
2,1.6	11	147	17	143	70	117
2,1.7	11	147	21	121	85	71
2,1.8	17	126	24	105	98	27
2,1.9	56	1	26	80	107	7
2,2.0	56	8	26	72	120	28

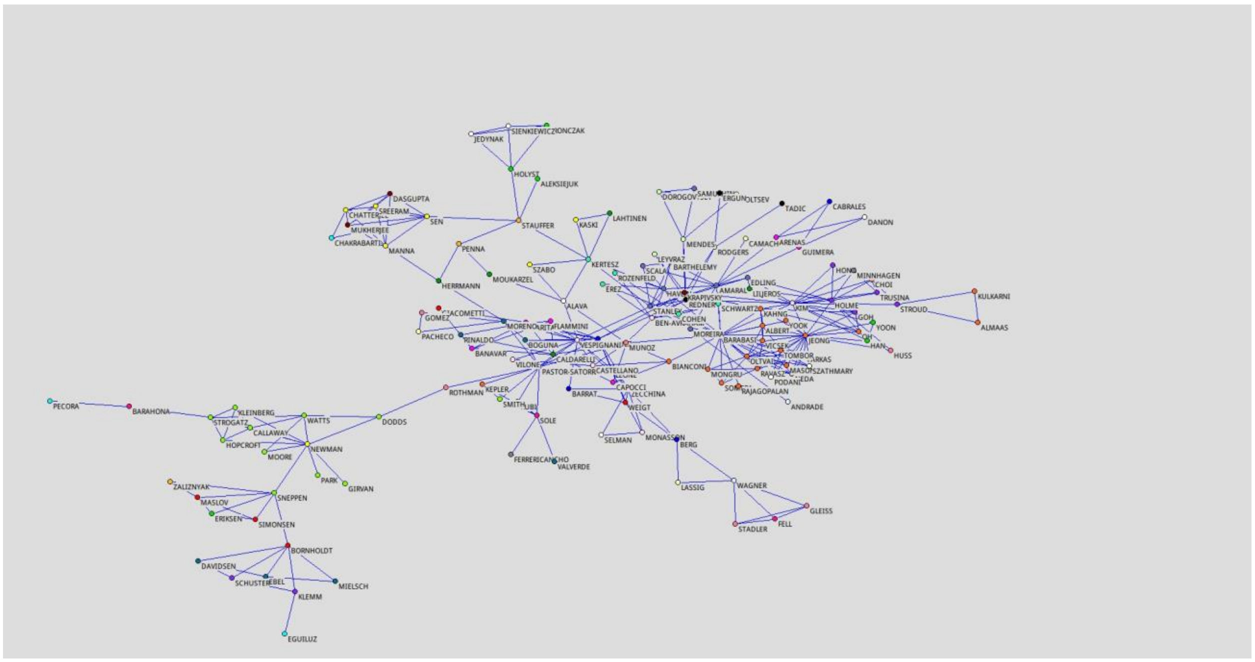
a. ATT dataset

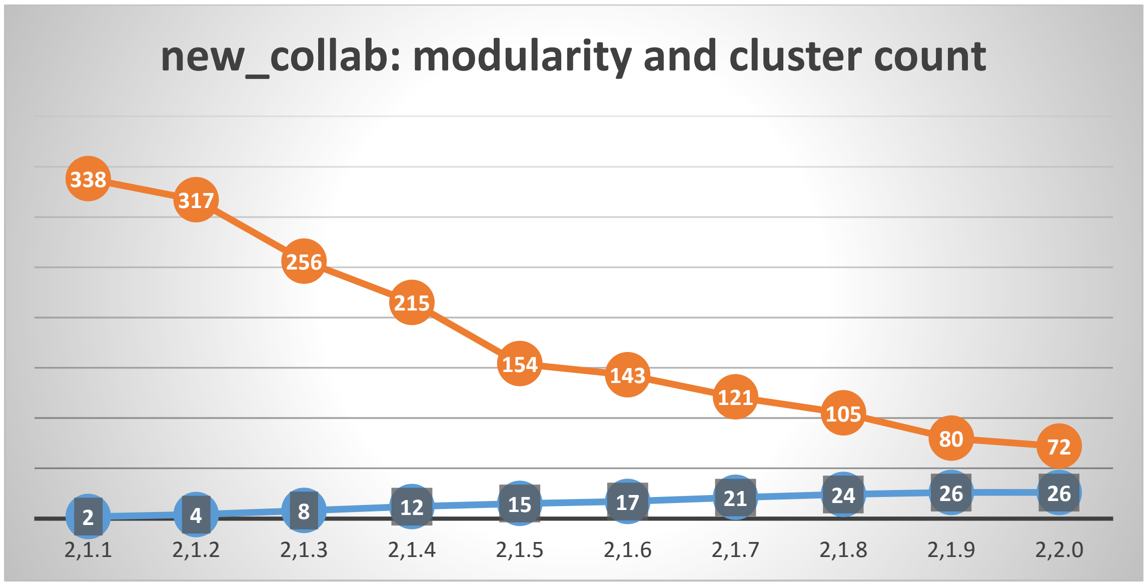


Modularity
Cluster Count

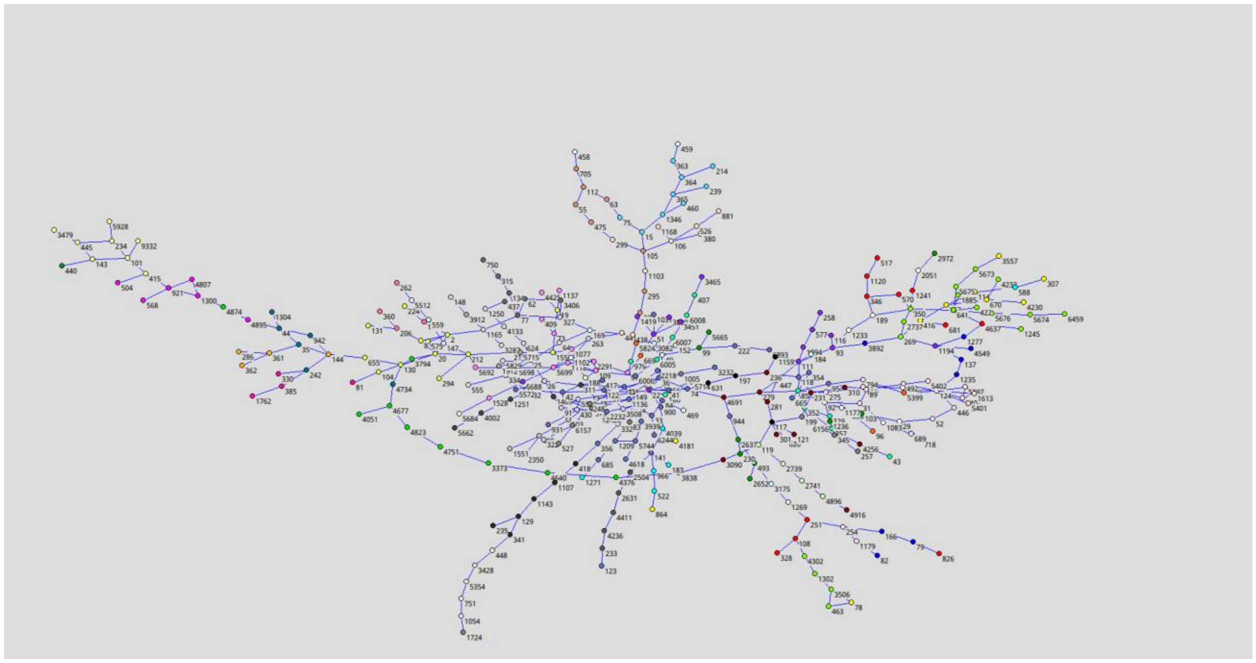


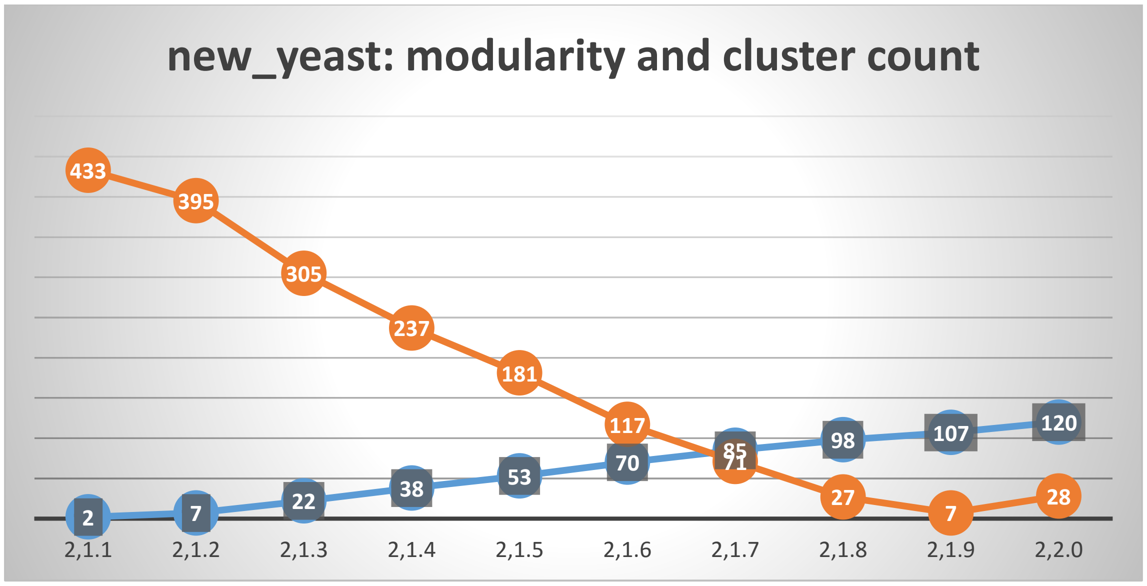
b. Collaboration dataset

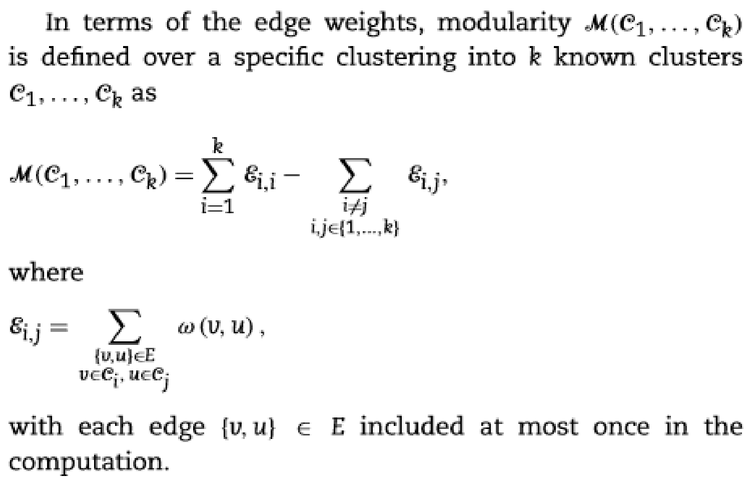




c. Yeast dataset






In terms of the edge weights, modularity $\mathcal{M}(\mathcal{C}_1, \ldots, \mathcal{C}_k)$ is defined over a specific clustering into k known clusters $\mathcal{C}_1, \ldots, \mathcal{C}_k$ as

$$\mathcal{M}(\mathcal{C}_1, \ldots, \mathcal{C}_k) = \sum_{i=1}^{k} \mathcal{E}_{i,i} - \sum_{\substack{i \neq j \\ i,j \in \{1,\ldots,k\}}} \mathcal{E}_{i,j},$$

where

$$\mathcal{E}_{i,j} = \sum_{\substack{\{v,u\} \in E \\ v \in \mathcal{C}_i,\, u \in \mathcal{C}_j}} \omega(v,u),$$

with each edge $\{v, u\} \in E$ included at most once in the computation.

We observed that modularity is higher for the graph when the elements in the converged matrix are aligned along the diagonal. Higher the modularity, higher is the inter-cluster distance (difference).
We tend to maximize the modularity index for a graph to ensure best quality of clustering.

High modularity → greater difference between clusters → better quality of clustering

We could have also used the silhouette index for the same purpose, which gives the idea of dissimilarity (and similarity) between clusters from its value ranging from -1 to 1.

The optimal parameters for the given datasets are:

a. new_att: e=2, r=2

b. new_collaboration: e=2, r=2

c. new_yeast: e=2,r=2

The optimal parameters have been determined from the visual interpretations of Pajek network partition visualizations.

2) Discuss how the two parameters, inflation rate and expansion rate, influence the clustering results, and how to choose the parameters.

<u>**Impact of inflation rate (r) on clustering:**</u>

- 'r' is responsible for both strengthening and weakening of current in a way that it strengthens strong currents, and weakens already weak currents.
- 'r' controls the extent of this strengthening / weakening. In the end, this influences the granularity of clusters.

Impact of expansion rate (e) on clustering:

- ‘e’ is responsible for allowing flow to connect different regions of the graph.
- ‘e’ also accounts for strengthening and weakening of current in the network.

Best practices:

- Code maintenance and version control in github
- User interactive input retrieval – choice driven file and expansion, inflation parameters
- Work distribution – algorithm design and implementation by Shahid Fazal, programming and visualization – Jagvir Singh, testing, data generation and reporting by Vidyadhar Reddy
-

References:

http://dollar.biz.uiowa.edu/~street/graphClustering.pdf

https://www.cs.ucsb.edu/~xyan/classes/CS595D-2009winter/MCL_Presentation2.pdf

https://piazza.com/class_profile/get_resource/idko1selpno766/iggqlrhzdjo1t1

Data Mining and Techniques (3rd Edition): http://romisatriawahono.net/lecture/dm/book/Han%20-%20Data%20Mining%20Concepts%20and%20Techniques%203rd%20Edition%20-%202012.pdf