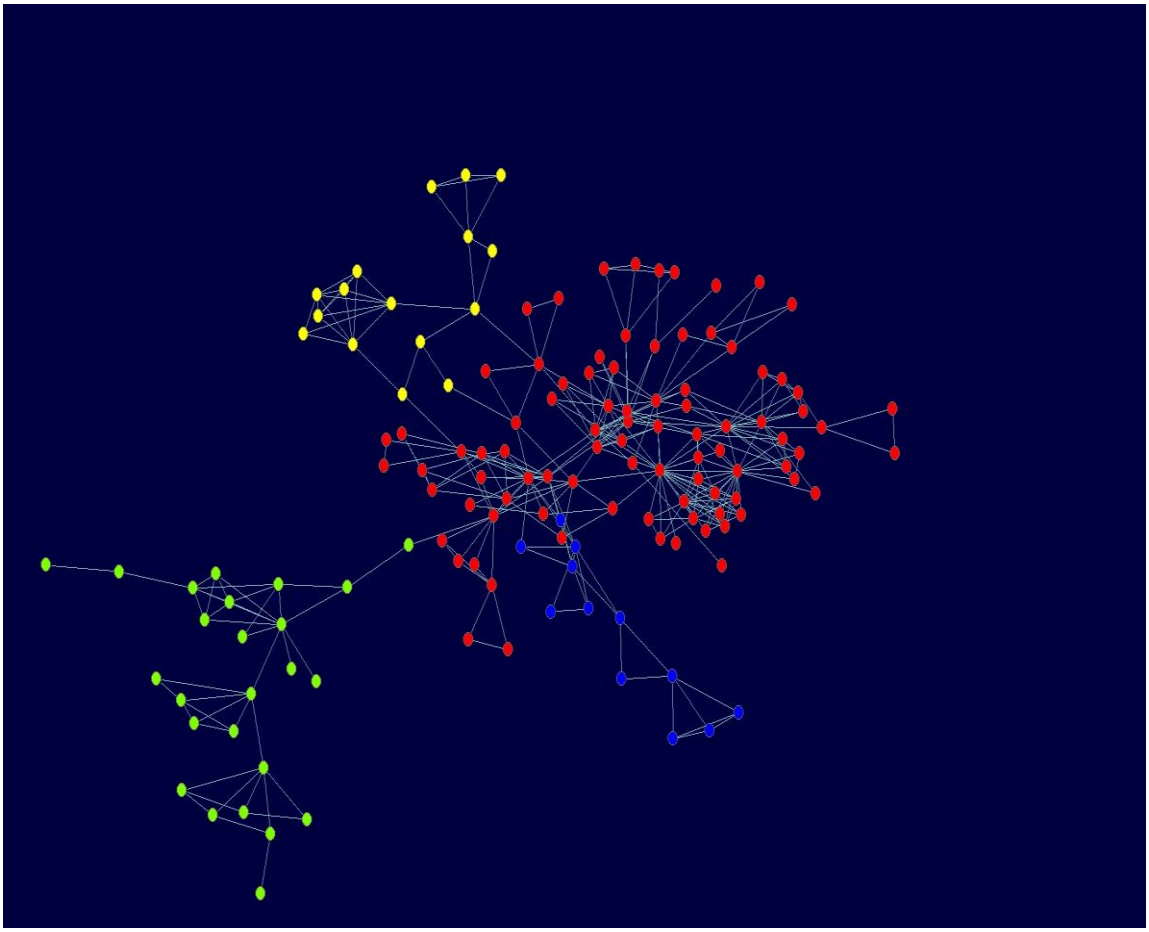


CSE_601: HW3

CLUSTERING ANALYSIS FOR COMPLEX NETWORKS

Harsha Sudarshan | 50169593 | harshasu@buffalo.edu
Akash Mandole | 50168447 | akashman@buffalo.edu
Bhavani Sundara Raman | 50169253 | bhavanis@buffalo.edu

Introduction:

One of the main challenges of analyzing complex networks is clustering of closely linked nodes. Another challenge is doing so in unsupervised manner. Markov Clustering Algorithm (MCL) is an algorithm that elegantly handles these challenges.

For this assignment, we were provided with data of three different networks and were required to analyze them by implementing MCL and visualizing the results using Pajek [1].

MCL algorithm:

The **MCL algorithm** is short for the **Markov Cluster Algorithm**, a fast and scalable unsupervised cluster algorithm for graphs (also known as networks) based on simulation of (stochastic) flow in graphs. The algorithm was invented/discovered by Stijn van Dongen at the Centre for Mathematics and Computer Science (also known as CWI) in the Netherlands[2].

The MCL algorithm finds cluster structure in graphs by a mathematical bootstrapping procedure. The process deterministically computes (the probabilities of) random walks through the graph, and uses two operators transforming one set of probabilities into another. It does so using the language of stochastic matrices (also called Markov matrices) which capture the mathematical concept of random walks on a graph.

The MCL algorithm simulates random walks within a graph by alternation of two operators called **expansion** and **inflation**.
Expansion coincides with taking the power of a stochastic matrix using the normal matrix product (i.e. matrix squaring).
Inflation corresponds with taking the Hadamard power of a matrix (taking powers entrywise), followed by a scaling step, such that the resulting matrix is stochastic again, i.e. the matrix elements (on each column) correspond to probability values.

A column stochastic matrix is a non-negative matrix with the property that each of its columns sums to 1. Given such a matrix M and a real number r > 1, the column stochastic matrix resulting from inflating each of the columns of M with power coefficient r is written Γr(M), and Γr is called the inflation operator with power coefficient r. Write Σr,j(M) for the summation of all the entries in column j of M raised to the power r (sum after taking powers). Then Γr(M) is defined in an entrywise manner by setting –

$$\Gamma_r(M_{ij}) = M_{ij}^r \,/\, \Sigma_{r,j}(M)$$

[2]

Implementation of MCL:

The algorithm was implemented in python 2.7 and using numpy, pandas libraries

1. Processing the input network data
    a. The given data was provided as .txt files with two vertices per line separated by space or tab
    b. The txt file was read into a pandas dataframe
    c. If the names of the vertices in the input file was not serialized, then
        i. a dictionary of vertex names against an int value was created from the input dataframe
        ii. Using the dictionary, a new input dataframe was constructed by mapping the original vertex names to dictionary values
    d. Using the input dataframe, a numpy matrix of NxN (N- number of distinct vertices in the dataframe) was initialized
        i. Considering the network as un-directed, for a edge between two vertices A,B was marked as edges A→B and B→A
        ii. Self-loop was created by initializing the matrix during creation
    e. Finally, normalization was applied to the matrix to create a stochastic matrix
2. The parameters for running the MCL, Expansion and Inflation were set
    a. The program was designed so that initial , step and max values for the two parameters were set to run the algorithm for multiple combinations of the parameters
    b. For edge cases, a max_iter (number of maximum iterations for the stochastic matrix to reach stability) was also set
3. Expansion was carried out by a numpy's linalg.matrix_power

    a. The expansion parameter needs to be a non-negative integer
4. Inflation was carried by raising the power of all the elements of the matrix using numpy's power function
    a. Immediately after inflation, the columns were normalized using scikit learn's preprocessing.normalize
5. After each iteration of expansion and inflation, the new matrix was checked for stability. Specifically, this involved the two steps:
    a. Check if the previous iteration of the matrix is the same as the current one
    b. Check if all the non-zero elements of a column are equal
        i. This is referred to Doubly idempotent stability
6. The expansion and inflation operation were placed in a single function expand_inflate_normalize
7. Two while loops were designed
    a. Outer-while loop was to iterate for various combinations of the two parameters as set above
    b. Inner-while loop was to handle the multiple iterations of the matrix for a single combination of parameters
        i. The inner-while loop checked for both number of iterations for the combination and stability of the matrix
        ii. Exit happened when either number of iterations exceeded the set limit or matrix reached stability
8. Once the inner-while executed (matrix reached stability), then vertices were grouped together by scanning the matrix row-wise
    a. All non-zero elements were grouped together and assigned a cluster number
    b. This cluster number was updated against the constituents in a result dataframe
9. The result dataframe has parameter combination as the column values.
    a. Ex: For parameter combination Expansion 2 and Inflation 1.6, the column name as 'exp_2_infl_1.6'
    b. This enabled storing the clustering results for various combinations in a single dataframe
10. The results for each parameter combination were written to file by
    a. Creating a folder having the same name as the input file(without file extension)
        i. Any existing folder of the same name was deleted
    b. The dataframe was iterated column-wise
        i. A txt file with name as the selected column was opened for writing

ii. The values in the selected column was written ; each value in a separate line
iii. Since the values in result dataframe are stored with indices starting from 0 and ascending, the cluster number corresponds to the expected format

11. Visualization was done using Pajek software
    a. Initially, a sample of output files was selected for visualization
        i. Sampling was done such that for each expansion value, the files with equally spaced inflation values were selected.
            1. Ex: for each expansion value, inflation values of 1.2, 1.4,1.6 – max(inflation_value)
        ii. The search was narrowed by selecting a smaller range between parameter combinations that provided the closest match to the given reference visualizations
        iii. Finally, the best perceived combination for a input file was selected

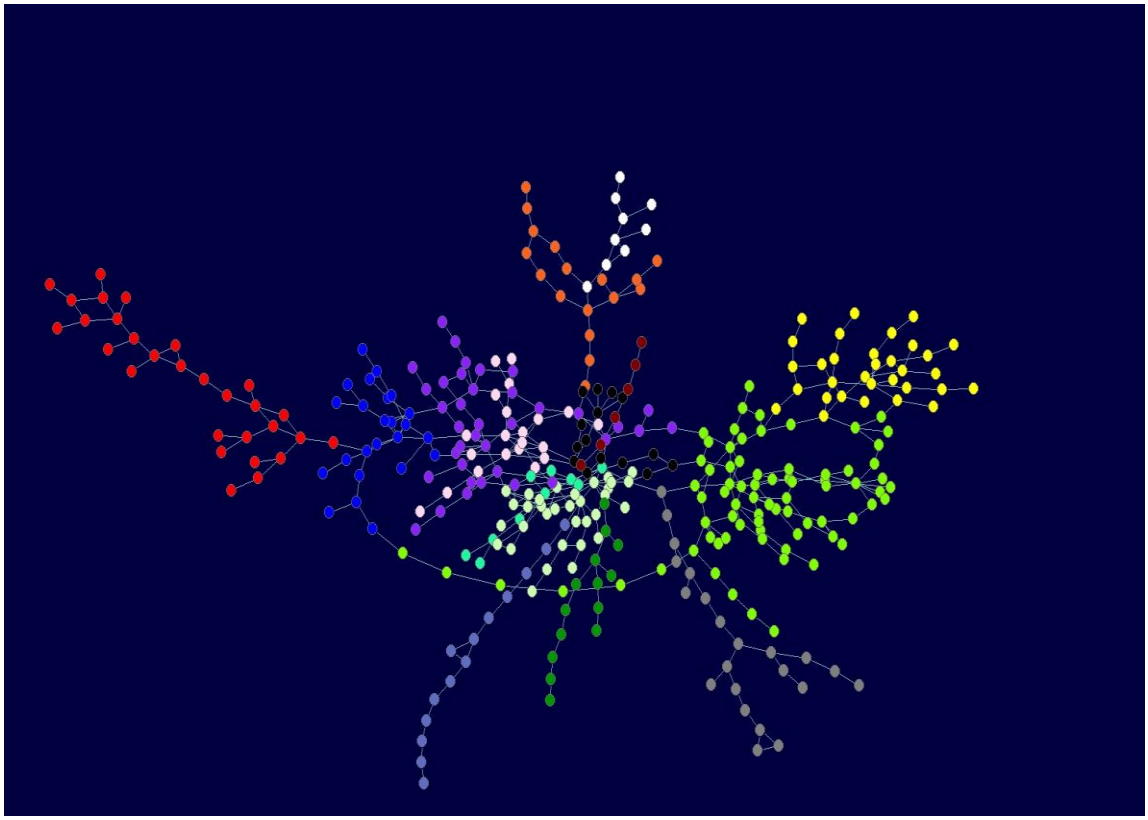
Results and Observations:

When the cluster outputs were analyzed, there was a clear discernable trend. For higher values of expansion, the reach of clusters grew, but with higher inflation parameters the granularity of the clusters increased. Thus, it was a balancing act between Expansion and Inflation parameters.

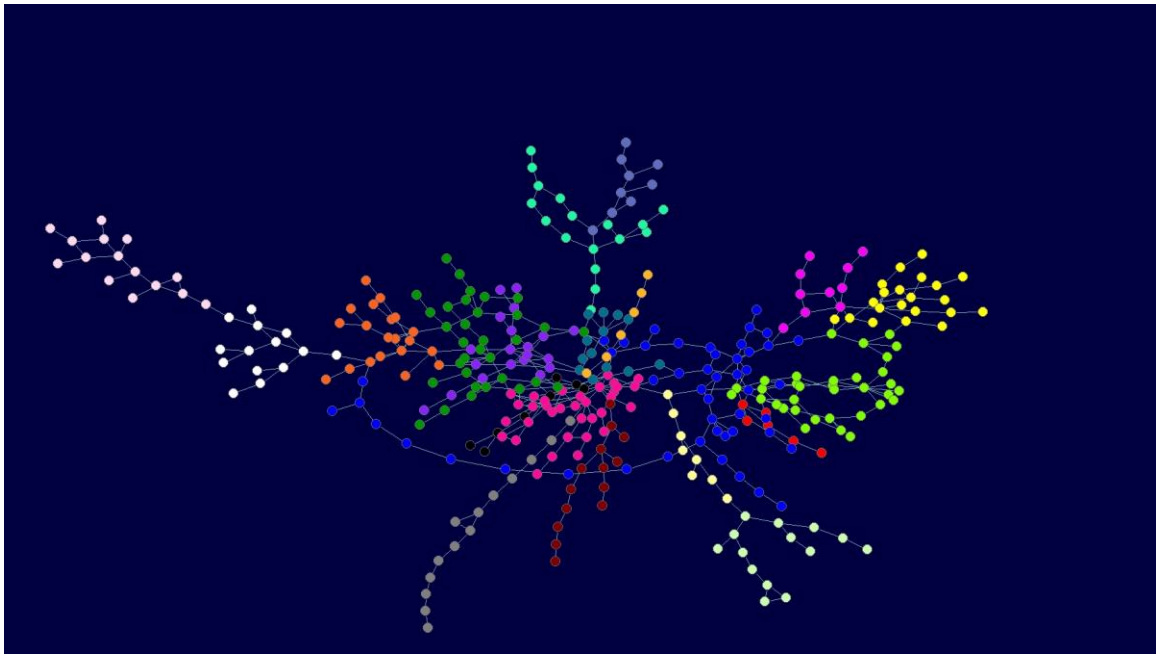
This corresponds to the comments of the author of MCL algorithm :
"Expansion corresponds to computing random walks of higher length, which means random walks with many steps. It associates new probabilities with all pairs of nodes, where one node is the point of departure and the other is the destination. Since higher length paths are more common within clusters than between different clusters, the probabilities associated with node pairs lying in the same cluster will, in general, be relatively large as there are many ways of going from one to the other. Inflation will then have the effect of boosting the probabilities of intra-cluster walks and will demote inter-cluster walks. "[3]

Because a perfect match was not found (compared to given reference visualization), The best matches for each data set is given below:

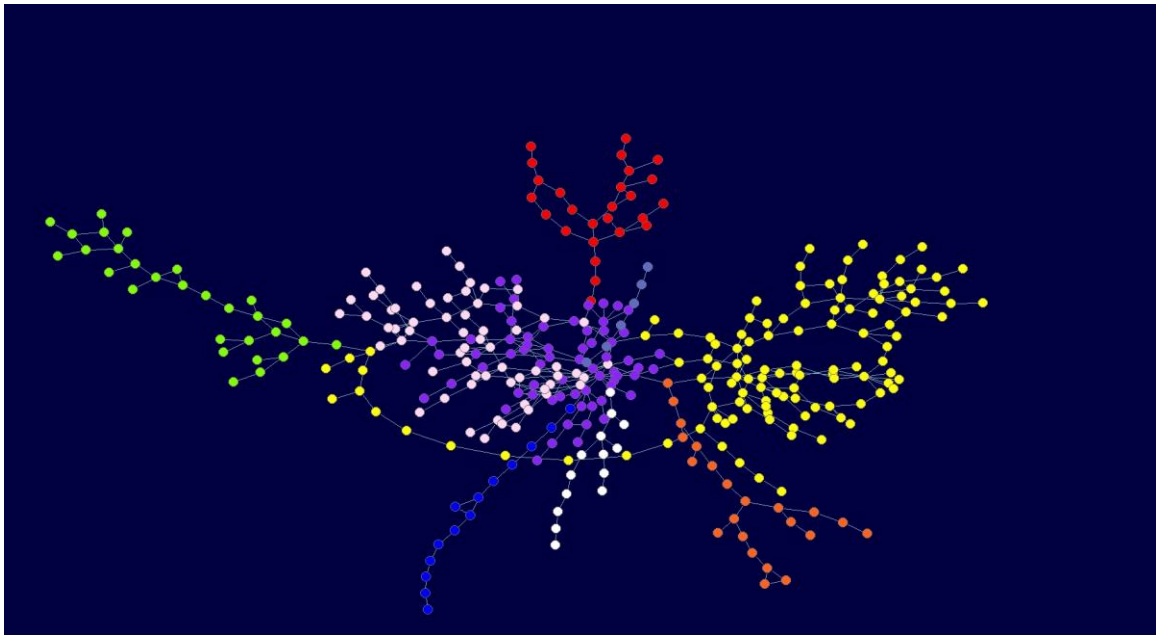
1. For "**yeast_undirected_metabolic**" dataset :
    a. Expansion : 5 , Inflation : 1.8



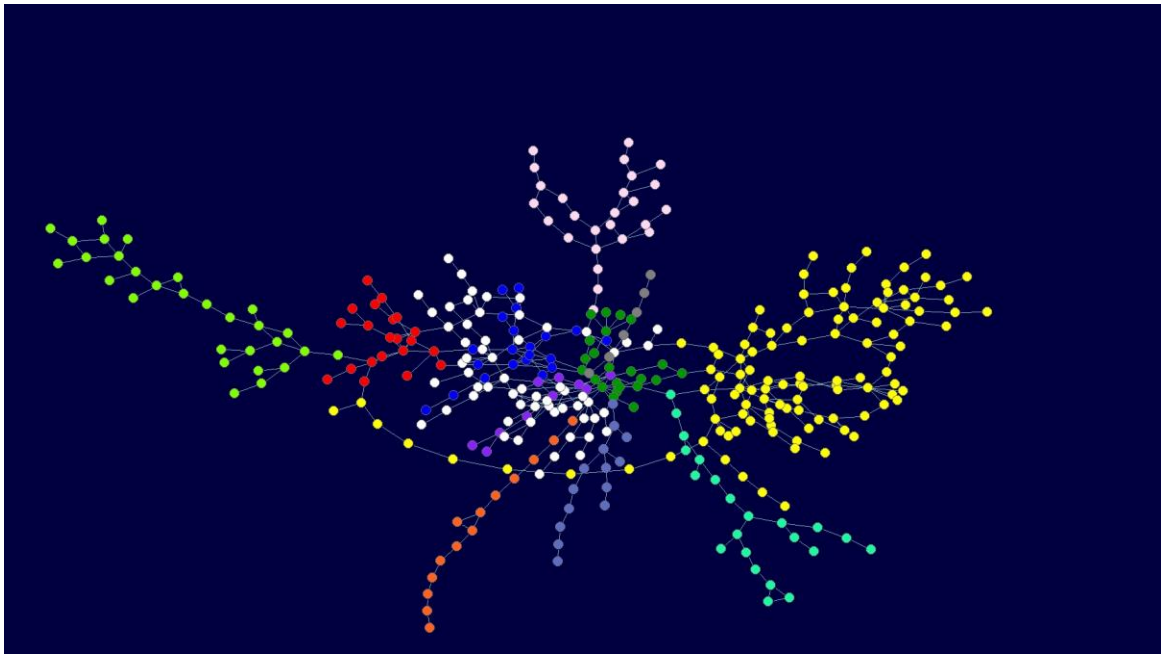
    b. Expansion : 5 , Inflation : 1.7



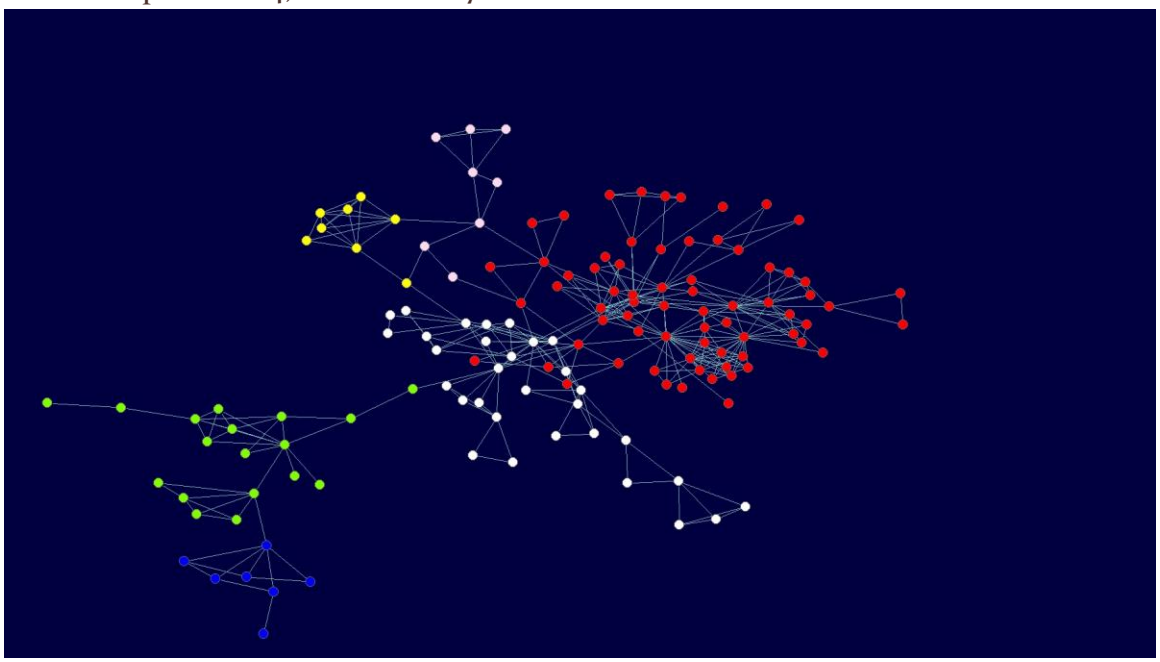
c. Expansion: 4 , Inflation : 1.5



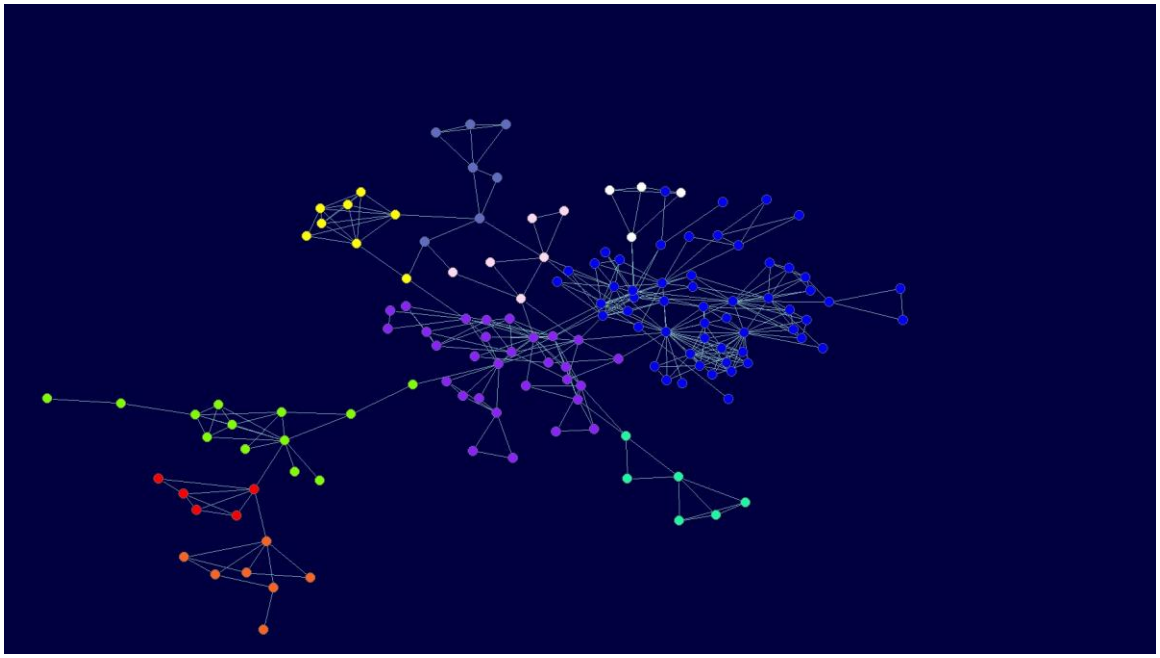
2. For "**physics_collaboration_net**" dataset :
    a. Expansion: 4 , Inflation : 1.6



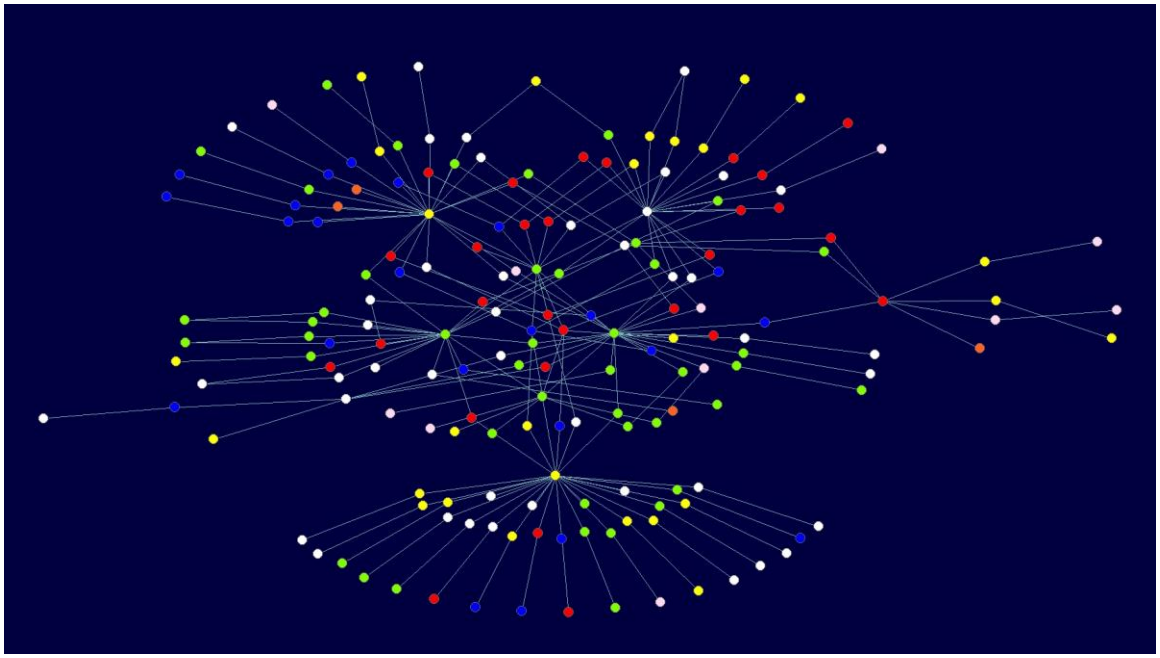
b. Expansion: 4, Inflation : 1.7



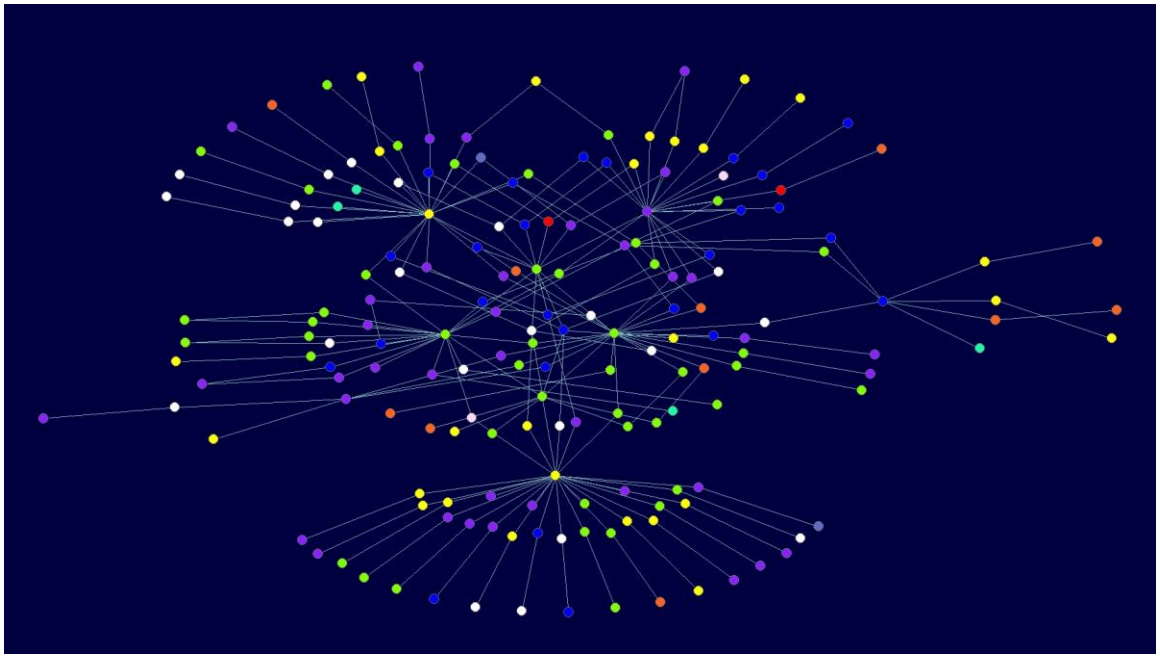
c. Expansion:3 , Inflation : 1.7



3. For "**attweb_net**" data set:
    a. Expansion 2: , Inflation: 1.5



    b. Expansion : 2, Inflation: 1.6



Additional Notes:

In the current implementation, pruning of stochastic matrix after expansion is not used. In the mcl tool[4] , the pruning is done in a number of different ways such as precision, selection number ,etc. With pruning included, we are confident that our program would provide more accurate results.

Along with the code, the data collected(.clu – cluster files) are provided in a folder called cluster_results.

References:

[1] - Pajek (http://vlado.fmf.uni-lj.si/pub/networks/pajek/).
[2] - http://micans.org/mcl/
[3] - http://micans.org/mcl/index.html?sec_description1
[4] - http://micans.org/mcl/man/mcl.html#opt-v-pruning