An Effective Comparison of Graph Clustering Algorithms via Random Graphs

Reena Mishra Shashwat Shukla Dr. Deepak Arora Mohit Kumar

Department of Computer Science and Engineering,
Amity University, Lucknow, Uttar Pradesh, India.

ABSTRACT
Many graph clustering algorithms have been proposed in recent past researches, each algorithm having its own advantages and drawbacks. All these algorithms rely on a very different approach so it's really hard to say that which one is the most efficient and optimal if we talk in the sense of performance. It is really hard to decide that which algorithm is beneficial in case of highly complex networks like PPI networks which consist of thousands of nodes. The paper proposes an effective data comparison of RNSC (Restricted Neighbourhood Search Clustering) and MCL (Markov Clustering) algorithms based on Erdos-Renyi and Power-Law Distribution graphs. The basic parameters used for comparison are Edge Density, Run Time, Number of Nodes, Cluster Size and Singleton Cluster. Our approach is an effective one because firstly we have used two types of graph generators, Erdos-Renyi and Scaled-Free for generation of input graphs which are very much closer to the real input graphs and secondly we have generated input graphs having more than 1000 nodes, so in our approach we have used both the algorithms for clustering highly complex input graphs just like PPI networks. For comparison and analysis purpose we have collected data sets and generated some graphs based on these parameters. The proposed approach depicts which algorithm is best to be used for clustering such complex graphs and also some fields for extension if possible in both them. All graphs used in this thesis are unweighted and undirected.

General Terms
Graph Clustering, Data mining et. al.

Keywords
RNSC, MCL, Erdos-Renyi, Scaled-Free, Edge Density, Singleton Cluster, Run Time, Number of Nodes, Cluster Size.

1. INTRODUCTION
Cluster Analysis [8] is the mathematical study of methods for recognizing natural group within a class of entities. The process of identifying similar structures in terms of grouping of data elements is called clustering. Graph clustering [1] and [5] is grouping of vertices a graph so that the vertices in one cluster have high intra connectivity as compared to inter-connectivity between different clusters. Graph clustering is closely related to graph partitioning. In a good clustering, the clusters have high density, i.e. are nearly complete graphs, and there are few edges in whose endpoints lie in deferent clusters. If G is a weighted graph, then we demand that in good clustering, each Ci contains a high edge weight sum and the sum of the weights of edges in G between these sub graphs is low [5]. In recent past researches various graph algorithms came into existence, like Markov Clustering (MCL), Restricted Neighbourhood Search Clustering (RNSC), Super Paramagnetic Clustering (SPC), and Molecular Complex Detection (MCODE), Local Clique Merging Algorithm (LCMA), Highly Connected Sub graph algorithm (HCS), SideS algorithm, Genetic algorithm, K-Means algorithm etc., each algorithm has its own strategy and relies on a very different approach. So it's really hard to say that which one is the most efficient and optimal if we talk in the sense of performance and robustness. In 2002, the yeast interactome was estimated to contain up to 80,000 potential interactions thus a highly complex PPI networks were discovered which consisted of thousands of nodes and vertices. Many authors have tried to find the most optimal one in their past researches, but the factor mainly depends on the network given for which clustering is done. Many authors in their researches have given their views about which algorithms to consider for PPI [10] networks. Mainly two algorithms attracted attention of many authors which are RNSC and MCL, while others were weaker under most conditions. In my research we will also see that which algorithm evaluates the "goodness" of the clustering's of a graph. Thus in my thesis work we are comparing these two algorithms under various parameters, and finally to generate a run time graph to compare their performances. The first segment of the paper depicts an overview of graph clustering algorithms (RNSC, MCL) and the graph generators which are to be used in our comparison. The next segment describes all the parameters to be used for comparison. In the last section we have provided all the results and discussions.

2. AN OVERVIEW OF GRAPH CLUSTERING ALGORITHMS AND GRAPH GENERATORS
In our approach we have used two graph clustering algorithms and two types of graph generator tools which we will be discussing in this segment.

2.1 RNSC
The RNSC [11] and [5] is a local search clustering algorithm [5].It uses two cost functions: Naïve and Scaled function. The naïve function is used as a preprocessor to the scaled cost function. Finally the scaled function tries to optimize the output from naïve function and reach to the global optimal solution. It is a local-search technique so only the neighbourhood moves are considered at every step, i.e. those clusters which can be reached from current cluster by moving a single vertex.

2.2 MCL

The MCL algorithm is a fast and scalable unsupervised clustering algorithm based on simulation of stochastic flow in graphs [7].It is a more natural and organic clustering algorithm. It is based on flow simulation technique. A graph is described as a flow between the vertices. Its clustering uses flow expansion and inflation to produce a natural grouping of highly flow-connected vertices or clusters, the algorithm takes flow expansion and inflation as inputs from the input graph which uses these two operators for transforming one set of probabilities into another, algorithm detects cluster structures in graphs by a mathematical bootstrapping [7] procedure. The process deterministically computes the probabilities of random walks through a graph by using the language of stochastic matrices (also called Markov [1] matrices), which capture the mathematical concept of random walks on a graph. Furthermore MCL uses threads also.

2.3 Erdos-Renyi

It is a graph generator tool based on probability distribution. This is a simple random graph in which the graph is represented in this form: $G_E(n, p)$, where G_E represents the Erdos-Renyi [2] graph and n is the number of nodes in the graph. Then every pair of vertices is check for edge and connected with probability p ($0 \leq p \leq 1$).

Algorithm: 1 Erdos-Renyi generator (Author's Proposed Algorithm)

n = number of graph nodes
p =probability of selection
$random()$ generates a random number
for $i = 1$ *to* n
    for $j = i + 1$ to n
        $temp = random()\%100 + 1$;
        if $temp \leq p$
            j is adjacent to i
            edgeCount++;
        end if
    end for
end for

2.4 Scaled-Free

It is also termed as Power-Law [6] Graph. It is also a graph generator tool but it is different from previous tool discussed. Scaled-Free [3], [4] and [6] graphs are good models for certain type of biological graph, web graphs and other naturally-occurring networks. In this graph the vertex degree follows a power-law distribution, i.e., there are large number of vertices with small degree and few vertices with very high degree (these vertices are also called hubs). The Scale free graph is represented by the notation $G_S(n, K)$, such that, G_S represent the scale free graph, n is the number of vertices in the graph, K is used to construct the graph by the following method:

Algorithm: 1 Scaled-Free generator (Scaled-Free generator code) [3], [5] and [8]

The integer n represents number of nodes in graph. Choose n and K such that $n > K$. Now a Power law graph can be constructed by numbering the vertices as $1, 2, \ldots n$. Make a set $S(K)$ of first k vertices, now start from k+1 vertex and construct a set $S(i)\ from\ S(i-1)$ where $i = K+1, K+2 \ldots . n$, by joining vertex $i\ to\ K$ random vertices in set $S(i-1)$.Choose a vertex v in $S(i-1)$ with probability proportional to $1 + degG(i-1)(v)$ and not allowing multi edges. That is, we put each vertex v in a bucket $1 + degG(i-1)(v)$ times and draw K non identical vertices. So $S(K+1)$ is a claw, and graph $G(n)$ is always connected. This graph contains $K(n-K)$ edges.

3. INTRODUCING PARAMETERS USED FOR COMPARISON

In this segment we will be describing all the parameters which we have used to generate results. These parameters are-

3.1 Edge Density

The density of an unweighted graph or cluster is the proportion of possible edges that are present [5].The density of a sub graph is calculated by the formula

$$\frac{2|E|}{|V|(|V|-1)}$$

Where $|E|$ is the number of edges and $|V|$ is the number of vertices of the sub graph.

3.2 Run Time

Run Time denotes the computation time taken by the algorithm to generate clusters from given input graph. Here we have calculated the runtime of both the algorithms by using Time Command in Linux. The algorithm having a lower computation time is said to be an optimal one.

3.3 Cluster Size

It is also termed as Number of Clusters. Cluster size denotes the total number of clusters generated by the algorithm. Cluster size may be viewed from the output graph.

3.4 Singleton Cluster

Singleton cluster is a cluster containing only single vertex. The algorithm which forms less singleton clusters is said to be an optimal algorithm.

3.5 Graph Size

It is also termed as Number of Nodes denoted by the variable (N). It is the total number of nodes in a Graph.

4. RESULTS AND DISCUSSIONS

4.1 Results for Erdos-Renyi Graph

This section contains all the results and discussions regarding Erdos-Renyi graphs.

4.1.1 Run time vs Edge density for RNSC and MCL

The table contains computed values of Run time and Edge densities for RNSC and MCL graph clustering algorithms.

Table 1. Dataset for Erdos-Renyi Graph

S. No.	For a graph of 1000 nodes		
	Edge Density	**Run Time (RNSC)** in seconds	**Run Time (MCL)** in seconds
1.	1	0.05	0.63
2.	10	4.24	4.51
3.	20	47.82	5.24
4.	30	56.25	5.60
5.	40	38.31	6.10
6.	50	28.11	6.79
7.	60	8.75	7.18
8.	70	11.71	7.39
9.	80	13.99	7.89
10.	90	16.50	8.84
11.	100	17.41	1.42

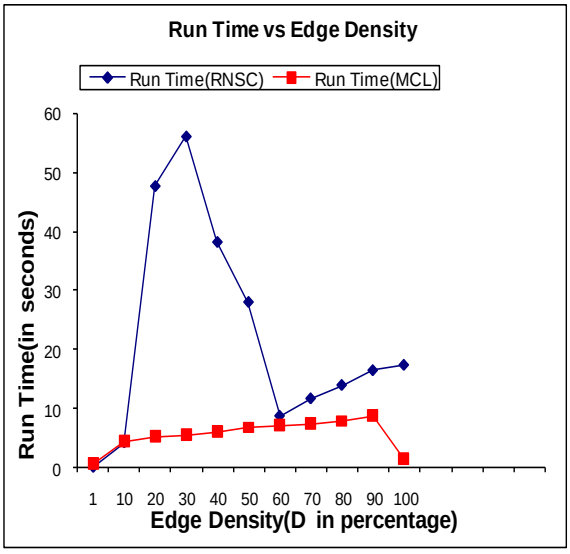

Fig 1: A line graph representing Run Time vs Edge Density.

Discussion: Fig 1 shows a line graph representing Run Time versus Edge Density for a graph of 1000 nodes. In case of MCL the run-time increases linearly with the edge density, but as the Edge-Density reaches 100% or fully-connected the run-time of the graph decreases drastically. This is evident from the fact that now the graph is completely connected and the cost of the clustering will decrease drastically in each step as more and more vertices are added to same cluster at each step. Thus the time to compute when graph is completely connected is much lower. The run-time of the RNSC clustering increases sharply at the edge density of 25-50%; this is attributed to the fact that in this range half the edge-pairs are connected. The number of moves is very high because the cost keeps fluctuating in this range due to diversification steps. As the Edge-Density crosses 50%, the edge-connectivity of the graph increases and the diversification step has low effect on such high Edge-Density. So the run-time decreases, but still increases linearly as the Edge-Density increases.

4.1.2 Edge Density vs Cluster Size in RNCS and MCL

The table contains computed values of Edge Density and Cluster Sizes for RNSC and MCL graph clustering algorithms.

Table 2. Dataset for Erdos-Renyi Graph

S. No.	For a graph of 1000 nodes		
	Edge Density	**Cluster Size (RNSC)**	**Cluster Size (MCL)**
1.	0.1	666	251
2.	1	304	903
3.	2	236	998
4.	3	195	1000
5.	4	172	1000
6.	5	124	1000
7.	6	83	1000
8.	7	47	1000
9.	8	17	1000
10.	9	8	101

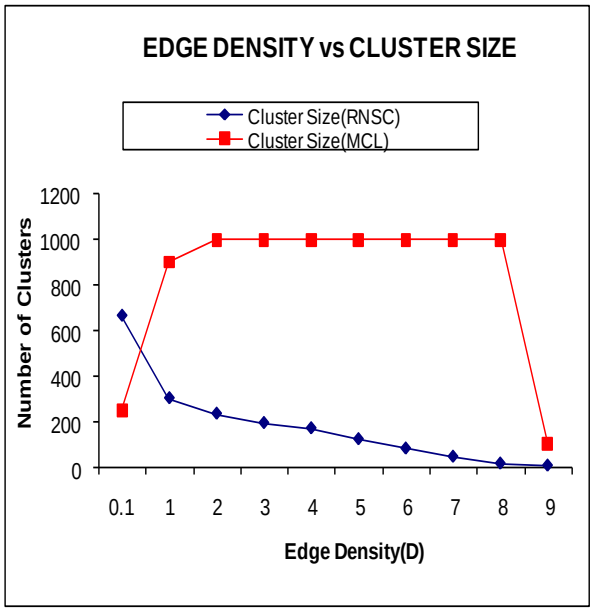

Fig 2: A line graph representing Edge Density vs Cluster Size.

Discussion: Fig 2 shows a line graph representing Edge Density versus Cluster Size for a graph of 1000 nodes. In case of the MCL clustering there is not much change in the number of clusters formed as the Edge-Density changes from 1 to 8, but on further increasing the Edge-Density, the number of cluster formed in the graph decreases. This is attributed to the fact that now there are more edges in the graph between the vertices. So the clustering cost decreases as the cluster size decreases. In the RNSC clustering we can observe that the number of clusters formed is constantly decreasing as the Edge-Density is increasing. This is obvious because now there are more edges in the graph and thus more compact clusters can be formed with lower costs. Thus the change in the number of clusters is easily observed in the RNSC clustering.

4.1.3 Number of Singleton Cluster vs Edge Density

The table contains computed values of Singleton Cluster and Edge densities for RNSC and MCL graph clustering algorithms.

Table 3. Dataset for Erdos-Renyi Graph

S. No.	For a graph of 1000 nodes		
	Edge Density	Singleton Cluster (RNSC)	Singleton Cluster (MCL)
1.	0.01	894	0
2.	0.1	402	5
3.	0.2	158	73
4.	0.3	58	172
5.	0.4	13	298
6.	0.5	5	411
7.	1	0	815
8.	1.5	0	962
9.	2	0	996
10.	2.5	0	1000
11.	3	0	1000
12.	9	0	101

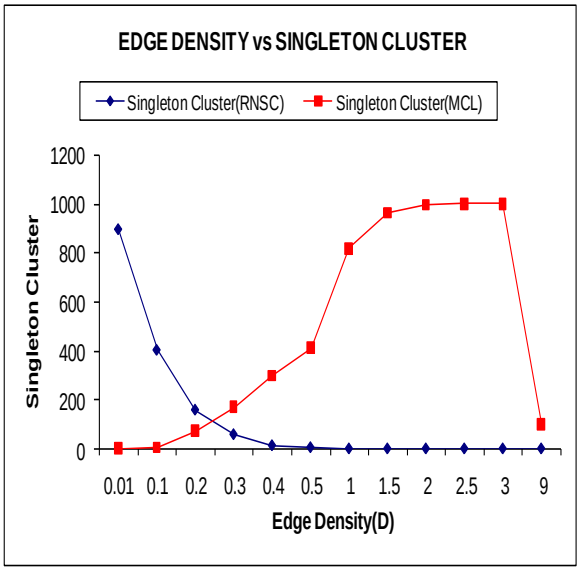

Fig 3: A line graph representing Edge Density vs Singleton Cluster.

Discussion: Fig 3 shows a line graph representing Edge Density vs Singleton Cluster for a graph of 1000 nodes. This graph shows the variation in the number of singleton clusters formed as the Edge-Density of the graph increases from 0 to 10% in Erdos-Renyi graph with 1000 nodes. The number of singleton cluster formed is high when the Edge-Density is below 0.3, but as the Edge-Density increases the number of singleton cluster cease to exist, which is good for any clustering algorithm. But on the other hand in the MCL clustering initially the number of singleton clusters were zero and as the Edge-Density increases the number of singleton cluster also increase. Thus the RNSC out-performs the MCL clustering in the singleton cluster count.

4.1.4 Run Time vs Number of Nodes

The table contains computed values of Run Time and Number of Nodes for RNSC and MCL graph clustering algorithms.

Table 4. Dataset for Erdos-Renyi Graph

S. No.	For a graph of 1000 nodes (Edge Density 5%)		
	Size of Graph	Run Time (RNSC)	Run Time (MCL)
1.	100	0	0.40
2.	200	0.02	0.14
3.	300	0.04	0.31
4.	400	0.10	0.76
5.	500	0.23	1.46
6.	600	0.34	2.57
7.	700	0.70	4.13
8.	800	1.16	6.30
9.	900	1.99	9.04
10.	1000	3.05	12.40
11.	1100	4.46	16.51
12.	1200	4.68	19.23
13.	1300	9.06	21.20
14.	1400	13.60	27.22
15.	1500	22.08	33.31
16.	1600	23.56	36.58
17.	1700	31.97	39.24
18.	1800	48.06	42.57
19.	1900	73.58	48.92
20.	2000	119.14	54.42

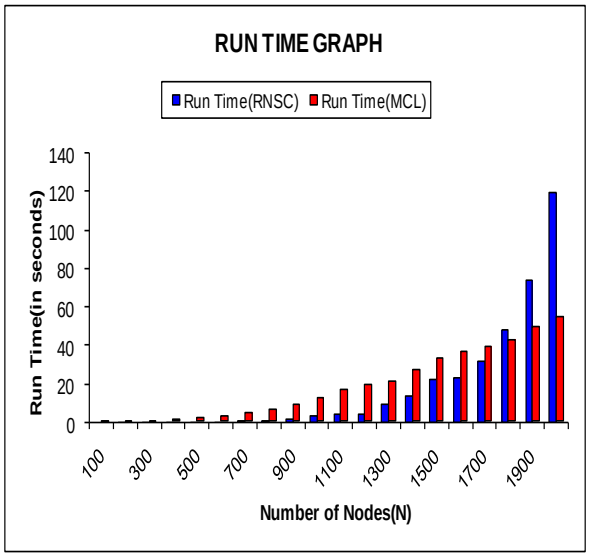

Fig 4: A bar graph representing Run Time vs Graph Size.

Discussion: Fig 4 shows a bar graph representing Run Time versus Graph Size for a graph of 1000 nodes. This graph shows that for graph of small sizes ranging from 100 to 1700 nodes, the run-time of RNSC clustering is lower than that of MCL clustering, but for more than 1800 nodes the run-time for RNSC increases drastically for Erdos-Renyi graphs. When the number of nodes in the graph reaches 1900 to 2000 nodes then the run-time of MCL algorithm is not increasing because the MCL

algorithm uses threads of execution for computing the clustering. The computation time is decreased due to the usage of threads, but the RNSC algorithm is serial algorithm, so the run time increases beyond MCL run-time. But after a certain limit, i.e. nearly 5k nodes the MCL algorithm run-time increases much more than the RNSC, this is evident from the other graph which shows the run-time for both algorithms for the graphs of order 10k to 20k nodes. When the order of nodes reaches 10k then even the threaded version of the MCL takes more time to compute than the RNSC algorithm on the dual-core machine.

4.2 Results for Scaled-Free Graph

This section contains all the results and discussions regarding Scaled-Free graphs.

4.2.1 Number of Nodes vs Cluster Size

The table contains computed values of Number of Nodes and Cluster Sizes for RNSC and MCL graph clustering algorithms.

Table 5. Dataset for Scaled-Free Graph

S. No.	For a graph of more than 10000 nodes		
	Number of Nodes	Cluster Size (RNSC)	Cluster Size (MCL)
1.	10000	7084	2254
2.	11000	7913	2462
3.	12000	8389	2497
4.	13000	9164	2916
5.	14000	10888	3106
6.	15000	10601	3419
7.	16000	11292	3518
8.	17000	12885	3781
9.	18000	13330	4885
10.	19000	13643	4186
11.	20000	14118	4556

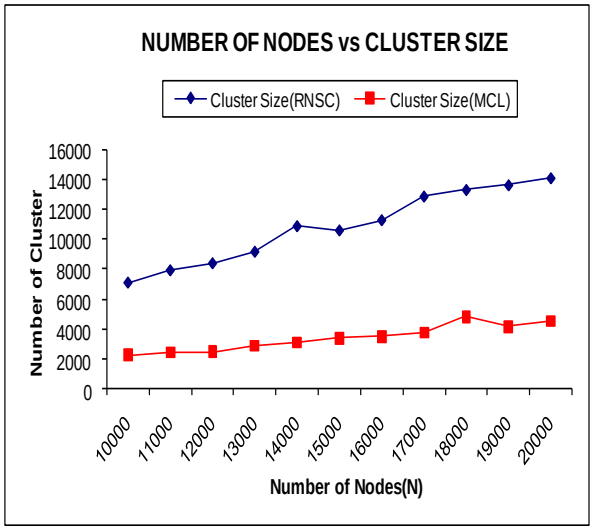

Fig 5: A line graph representing Graph Size vs Cluster Size.

Discussion: Fig 5 shows a line graph representing Graph Size versus Cluster Size for a graph of, more than 10000 nodes. This graph shows that as the number of nodes increases in the graph the number of clusters formed increase linearly in the graph, with the condition that the graph follows the same attribute like Edge-Density and connectivity. The number of clusters formed in the RNSC clustering is much higher than the MCL clustering for power-law graphs. This is due to the fact the number of singleton clusters is very large in this type of graph. Singleton cluster is a cluster containing only single vertex. Singleton clusters result due to the fact that the cost of the cluster decreases sometime with singleton cluster sparsely connected graphs like power-law graphs. The number of clusters formed in RNSC can be minimized by changing the CLUSTER_LIMIT parameter in the algorithm. This parameter sets the maximum limit on the number of clusters formed. The above graphs are obtained by keeping the CLUSTER_LIMIT parameter as null. When the parameter is not set then the CLUSTER_LIMIT is set as the maximum number of nodes in the graph. The above graph is obtained by keeping the CLUSTER_LIMIT unchanged, i.e., maximum number of nodes in the graph.

4.2.2 Graph size vs Run Time

The table contains computed values of Graph Size and Run Times for RNSC and MCL graph clustering algorithms.

Table 6. Dataset for Scaled-Free Graph

S. No.	For a graph of more than 10000 nodes		
	Number of Nodes	Run Time(RNSC) in seconds	Run Time(MCL) in seconds
1.	10000	0.71	5.7
2.	11000	1.31	6.8
3.	12000	1.06	11.6
4.	13000	1.25	12.46
5.	14000	2.14	9.67
6.	15000	1.68	7.79
7.	16000	1.68	17.01
8.	17000	2.2	13.5
9.	18000	6.12	9.23
10.	19000	3.04	24.27
11.	20000	2.82	12.63

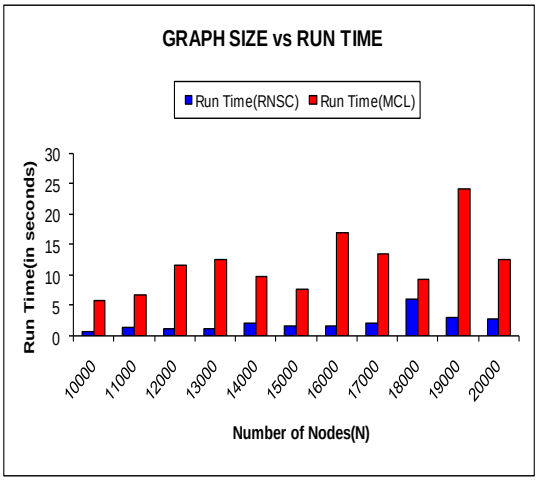

Fig 6: A bar graph representing Run Time vs Graph Size.

Discussion: Fig 6 shows a bar graph representing Run Time versus Graph Size for a graph of more than 10000 nodes. As the order of the graph increases from 10000 nodes to 20000 nodes, the run-time of both RNSC and MCL increase but the run-time of RNSC is much smaller as compared to that of MCL. The graph here is power--law graph with alpha value 2.5. The RNSC performs much better for power law graph of order 10k-20k nodes. This can be attributed to that fact that the edge connectedness in the power law graph is not very high for all the vertices. So the change in the clustering due to the change in diversification is not very large, thus easily a final naive and scaled cost is achieved and the number of moves taken is less. In the case of MCL algorithm, the clustering is obtained by calculating the matrix multiplication which becomes very expensive for the graph of the order of 10-20k nodes. The run-time of the RNSC clustering scheme can be further optimized by implementing parallelization in the algorithm.

5. CONCLUSIONS AND FUTURE WORK

In our approach we successfully implemented RNSC and MCL graph clustering algorithms in C++ for graphs having more than 1000 nodes. With the help of these graphs we were able to compare both the algorithms with different parameters and in different conditions. In case of Erdos-Renyi graphs run time of RNSC algorithm is better as compared to MCL for graph having nodes less than 1800 but as nodes keep on increasing the run time of RNSC increases drastically while run time of MCL does not increase, so MCL is better in case of Erdos-Renyi graph having more than 1800 nodes but after a certain limit of about 5k nodes and more and also due to high connectivity between the nodes it performs poorly as compared to RNSC. Future work will focus in optimizing the run time of RNSC algorithm by implementing parallelization in the algorithm and also by improving the heuristic approach of RNSC algorithm. MCL algorithm performs very poorly in case of sparse graphs while RNSC is better than MCL in case of sparse graphs i.e. in the case of Scaled-Free graphs. Speed in case of MCL may be improved through pruning.

6. ACKNOWLEDGMENTS

The authors are very thankful to their respected Mr. Aseem Chauhan, Additional President, Amity University, Lucknow, Maj. Gen. K.K. Ohri, AVSM (Retd.), Director General, Amity University, Lucknow, India, for providing excellent computation facilities in the University campus. Authors also pay their regards to Prof. S.T.H. Abidi, Director and Brig. U.K. Chopra, Deputy Director, Amity School of Engineering, Amity University, Lucknow for giving their moral support and help to carry out this research work. We are very much thankful to all the technical and non-technical staffs of the A.S.E.T for their assistance and co-operation. Lastly we thank all of our colleagues who directly or indirectly supported us.

7. REFERENCES

[1] Sauta Elisa Schaeffer, "Survey Graph clustering," Elsevier Computer Science Review, vol. I, pp. 27-64, 2007.

[2] P. Erdos and A. Renyi. On the evolution of random graphs. Publ. Math. Inst. Hungar. Acad. Sci., 5:17-61, 1960.

[3] A.-L. Barabasi and R. Albert. Emergence of scaling in random networks. Science 286(5439) (1999) 509-512.

[4] A.-L. Barabasi and Z. N. Oltvai. Network biology: Understanding the cell's functional organization. Nature Reviews Genetics, 5:101-113, 2004.

[5] A.D. King, Graph clustering with restricted neighbourhood search. Master's Thesis, University of Toronto, 2004.

[6] X. Hu and J. Han. Discovering clusters from large scale-free network graph. In ACM SIG KDD Second Workshop on Fractals, Power Laws and Other Next Generation Data Mining Tools, August 2003.

[7] S. Enright, A.j.van Dongen, C.A. Ouzounis, An efficient algorithm for large-scale detection of protein families, Nucleic Acids Res. 30(7) (2002) 1575-1584.

[8] S. M. Van Dongen. Graph Clustering by Flow Simulation. PhD thesis, University of Utrecht, May 2002. [Online].Available: http://www.svdthesis.pdf

[9] Scaled-Free graph generator code. [Online].Available: http://www-rp.lip6.fr/~latapy/FV/

[10] King, A. D., Przulj, N., and Jurisica, I. (2004) Bioinformatics 20, 3013-20.

[11] King, A. D. (2005), McGill University, Montreal.