ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology
Vol. 2, Issue 5, May 2015

Understanding Developer Relations in FOSS Ecology using SNA Parallelization Technique

Poornesha B D[1], Sujay Shalawadi[2]

M S Ramaiah Institute of Technology Bangalore, India[1,2]

Abstract: Social network analysis aims at understanding the organization of a social network at various levels. This paper involves analysis of BigData related to social interaction among developers in Free and Open Source Software ecosystem. The analysis involves spotting influence, predicting future links and Clustering entities. The analysis was first made sequentially. However, as the size of the data kept increasing sequential computing was inefficient. In this paper, the analysis has been taken forward in parallel implementation on multi-node Hadoop cluster to improve computation time. Effective performance benefits have been achieved by considering the terms that affect the performance.

Keywords: Influence, Link Prediction, Proximity measure, Spanning Tree, Clique, Purity.

I. INTRODUCTION

There has been constant research in the field of graph mining with respect to massive data sizes. The main questions that are trying to be answered are how do we find patterns of data that have billions of entities or the

sheer size of data runs in terabytes or petabytes. Specially in the study of social network analysis where there are enormous number of interactions being done at regular intervals of time, we need to scale up our computational resources efficiently in order to study the patterns of these ever growing communities.

In this paper, we have taken the data from SourceForge.net, which is a source code repository and is the first to offer this service to the Free and Open Source Software (FOSS) community [1]. The analysis involves finding patterns among developers interacting while collaborating their work of research. We have made use of graph mining concepts to find the influential nodes, predict future links and form clusters of most similar developers utilizing a distributed network with help of Hadoop and R[15] to improve the scalability with respect to increase in size of data which was a major drawback with sequential computation.

II. RELATED WORK

The first idea of analysis of FOSS data originated in 2002 by Gregory Madey, where the developer projection interaction was defined and has been gaining attention[2] and there has been advances in the analysis where the methods of communication between developers have also been defined by the year 2005[11][12].

A study from Syeed and Hammouda in the year 2014 explains the many developers will be working on the FOSS projects, tracking resembling open source projects by exploiting the information of which developers contribute to which projects [3]. Social network study to analyze data and resemble with respect to properties such as project application domain, programming language used and project size.

Social network analysis using graph mining as the principle to finding patterns is not just limited to FOSS community but is used extensively in other fields such as viral marketing, social networking sites, computational biology to name a few. The use of graphs as way of representing the data sets is beneficial specially for unsupervised and semi-structured data [4].

The analysis of big data using graph mining can be termed as big graph mining. The principle of graph mining has provided good results in answering questions like what are the distinguishing characteristics of the graph?, are there any patterns in the graph? and how do these graphs evolve over time?

III. SOCIAL NETWORK ANALYSIS USING GRAPH MINING TECHNIQUES

The concepts made use are PageRank for spotting influence, Proximity Measure for Link Prediction and Graph Based Clustering.

A. PageRank

In a social network, the graph of relationship and interaction within a group of individuals plays a important role as a medium for the spread of ideas, information among its members [5]. Influence maximization is the problem of finding a small subset of nodes i.e. seed nodes in a social network that will maximize the spread of influence [6]. Influence node depends on the few parameters like connectedness of a node, priority and position of a node in social network.

The use of PageRank algorithm was inspired for its ability to rank web pages in linked web page structure. We

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
Vol. 2, Issue 5, May 2015

wanted to make use of the same concept on the FOSS data set by representing developers as web pages and using the interactions between them to rank them accordingly[7]. It is based on the random walk process of PageRank equation as shown below, where d is the damping factor which can be set between 0 and 1, PR(N) is the PageRank of page A, PR(Ti) is the PageRank of page Ti which link to page N and C(Ti) is the number of outbound links on page Ti.

$$PR(N) = (1-d) + d(\frac{PR(T1)}{C(T1)} + ... + \frac{PR(Tn)}{C(Tn)})$$

Table 1 shows the most influential nodes in a dataset, using this approach we can find out the most influential or centralized-coordinated developer of the projects.

Table 1: Top 5 Influential Developer ID's in the dataset.

Influential developer ID's	Rank
1562343	1
3359051	2
3134053	3
3119739	4
2935535	5

B. *Link Prediction*

In a Link Prediction method, its possible to identify the new interaction between the developers who have never interacted before. When a Link Prediction is applied to developer-developer relation dataset, it predicts the relation that occurs in the future based on the developers interests in particular type of projects. There are few efficient measures which are introduced for choosing the feature set like proximity feature [8]. We have made use of similarity proximity measure where the number of mutual neighbors determine the probability of future links[7]. Higher the mutual neighbors, the future is most likely to occur. Based on an analysis, the accuracy of the similarity measure was higher than the dissimilarity measure. Therefore, similarity measure with correlation.

Data: Rawdataframe.
Result: Predicted list of Edges.
Step 1: Convert Rawdataframe into graphs.
Step 2: Convert Graph into a matrix.
Step 3: Use the matrix to compute proximity matrix so that an adjacency matrix is formed.
Step 4: Normalize values in the adjacency matrix between 0 and 1.
Step 5: Apply threshold of 0.75 if similarity is used and 0.25 if dissimilarity measure is used.
Step 6: Retain values above 0.75 and below 0.25 for similarity and dissimilarity respectively.
Step 7: Extract corresponding row and column values which will give the list of the predicted edges.

Algorithm 1: Graph Processing for Input data

Below are the results of Link Prediction for FOSS developer-developer data set taken for our analysis:

- Total number of predicted links = 14203
- Number of links involving influential nodes = 4457
- Percentage of probable links from the influential nodes = 32\%

As per the threshold set as 75%, the social network is bound to grow 32% as per the existing relationship between developers.

C. *Graph Based Clustering*

The various information that could be generated using the graph cluster methods are the number of developers in the data set, number of clusters formed, minimum and maximum number of developers, average number of developers in the cluster, the average number of developers in a cluster can be calculated for only those methods where the developer IDs don't repeat themselves in more than one cluster. The reason for this is the count of the developers in all the clusters greatly exceeds the value of the total number of developers of the data set.

The clustering process should also be evaluated to check for the performance, so the purity of clustering is checked. Purity is defined as number of cluster containing average number of developers by number of cluster not containing average number of developers. This measure can only to be used on a method where the average number of developers in a cluster can be calculated. The value of purity of clustering will be in the range of 0 to 1. Value closer to 1 indicates good clustering and the values closer to 0 indicate the poor clustering.

1. Highly Connected Subgraph(HCS) Clustering:

The input will have to be given with a predefined cut threshold. The cuts are made on the graph iteratively to form many clusters that satisfy the condition that the number of edges is greater than half the number of vertices[9][19].

Data: G(V,E), t
Result: Set of HCS cluster of dataset
while i <= t
do
    if K(Hi) > n/2 then
        RESULT<-Hi;
    else
        goto HCS(Hi);
    end
end

Algorithm 2: Highly Connected Subgraphs Clustering.

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
Vol. 2, Issue 5, May 2015

Below are the analyzed results of the HCS clustering.

- Total number of developers in the dataset = 14374
- Total number of developers interaction in dataset = 35305
- Total number of clusters formed = 3851
- Minimum number of developers in a cluster = 1
- Maximum number of developers in a cluster = 176
- Average number of Developers in a cluster = 3.73
- No of cuts specified is = 3
- Number of clusters with average number of developers = 827
- Purity of the clustering algorithm = 0.275

The variation between the maximum number of developers and minimum number of developers in a cluster is low and hence its purity value is better than the other algorithms. The clusters are formed with developers only if a developer has a mutual friend with all the other developers in the cluster.

2. Maximal Clique Enumeration

The concept is mainly used to form clusters such that each developer is connected to all other developers in that cluster. The clique becomes a Maximal Clique only if it does not form a subset of bigger clique[10].

Data: G (V;E)
Result: Set of Maximal Cliques involving all
Developers
Step: C=0
foreach v in V of G(V,E)
do
find a clique c with vertex v
remove all edges in c from G and c to C
done

Algorithm 3: Maximal Clique Clustering

Below are the analyzed results of the Maximal Clique Enumeration clustering.

- Total number of developers in the dataset = 14374
- Total number of developers interaction in dataset = 35305
- Total number of cliques formed = 6362
- Minimum number of developers in a clique = 2
- Maximum number of developers in a clique = 18

The average number of developers in a clique cannot be found because the developers repeat themselves in different cliques. All the maximal cliques formed are unique, i.e. the same sets of developers in one clique are not repeated in another clique. Maximal Clique shows the neighbors of developer connected each other in a network.

3. K - span Clustering.

A minimum spanning tree (MST) is obtained where all the lesser weighted edges are appropriately cut off using the theory of Prim's algorithm [11]. The MST is used as where k-1 highest weighted edges are cut off to form clusters [12].

Data: G(V,E), K

Result: K number of Clusters.

Step 1: Applying PRIMS to get
MST P(V,E) <- G(V,E)
Step 2: Removing K-1 highest weighted edges
from P(V,E).
Step 3: C={C1...Ck} K clusters formed from
step 2

Algorithm 4: KSpan

Below are the analyzed results of the Kspan clustering.

- Total number of developers in the dataset = 14514
- Total number of developers interaction in dataset = 33344
- Total number of clusters formed = 1135
- Minimum number of developer in a cluster = 1
- Maximum number of developer in a cluster = 7089
- Average number of Developers in a cluster = 12.78
- Number of cuts specified = 4
- Purity of the clustering algorithm = 0.0862

Kspan shows that how effectively a developer can reach all other developers in a network for communication with less weight or cost.

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
Vol. 2, Issue 5, May 2015

4. Betweenness Centrality

Betweenness Centrality quantifies the degree to which a vertex or edge occurs on the shortest path between all the other pairs of nodes [13][17]. It helps to get the more precise rank of the developers. It has two variations

4.1 Vertex Betweenness:

It is the number of shortest paths in the graph G that pass through a given node [13][17]. The centrality of each node is calculated using the degree count of the nodes. The highest centrality of the measured node will be identified and the graph will be split. Clusters are formed until the centrality measure of the highest node in the cluster is less than the specified centrality threshold.

Below are the analyzed results of the Vertex Betweenness clustering.

- Total number of developers in the dataset = 14374
- Total number of developers interaction in dataset = 35305
- Total number of clusters formed = 6346
- Minimum number of Developers in a cluster = 2
- Maximum number of Developers in a cluster = 115

Threshold Specified=0.2

Threshold = 0.2 means that a cluster will be formed only if the highest centrality in the cluster corresponding to a developer is less than the threshold. The developers are repeated so the average number of developers cannot be found.

4.2 Edge Betweenness:

It is the number of shortest paths in the graph G that pass through given edge. The centrality is calculated for every edge based on the betweenness in the graph [14][18]. The process is similar to the Vertex Betweenness clustering.

Below are the analyzed results of the Edge Betweenness clustering.

- Total number of developers in the dataset = 14374
- Total number of developers interaction in dataset = 35305
- Total number of clusters formed = 1032
- Minimum number of developers in a cluster = 2
- Maximum number of Developers in a cluster = 9623
- Average number of Developers in a cluster = 13.92
- Purity of the clustering algorithm = 0.00321

Threshold Specified=0.2

Threshold=0.2, which means that a cluster will be formed only if the centrality in the cluster corresponding to a pair of developers is less than the threshold. The developers are not repeated hence the average number of developers working on a project together can be found. The variation between the clusters containing maximum and minimum developers is large hence the purity is very low. Purity of a cluster is inversely proportional to the variation between the maximum and the minimum clusters containing the developers.

Betweenness helps in identifying centralized edges and vertices which helps to connect the other subgraphs of a network. These edges and vertices play an important role in connectedness of network.

IV. PERFORMANCE MEASURE

A. Experiment 1:

Experiments was carried out with RHadoop application cluster with one master and four slaves of computer systems, each system has Intel Core i5 quad core processor i.e. each system has four cores with 4GB RAM and average load of the system is balanced between 3.5 to 4.4. We have performed two experiments, one with the variable number of slaves/systems with constant file size and another with constant number of slaves/system with variable number of file size. First experiment carried out with constant file size and with the variable number of slaves. The time of execution is measured for each operation as shown in the Table 2

Table 2: Variable number of slaves with constant file size

RHADOOP RESULTS					
Execution Time(Seconds)					
Technique	Number of slaves				
	1	2	3	4	5
PageRank	11	10	10	9	10
Prediction	14101	13025	6740	6837	6752
HCS	911	630	448	498	502
Clique	70	71	67	65	66
Kspan	16	16	13	14	13
Betweenness	673	496	363	378	367

From the Table 2, Clique, Kspan and PageRank data mining technique completed in almost constant amount of time for variable number of slaves and it also confirm that only one slave is enough to compute these technique and HCS, Link Prediction and Betweenness completed in variable amount of time.

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
Vol. 2, Issue 5, May 2015

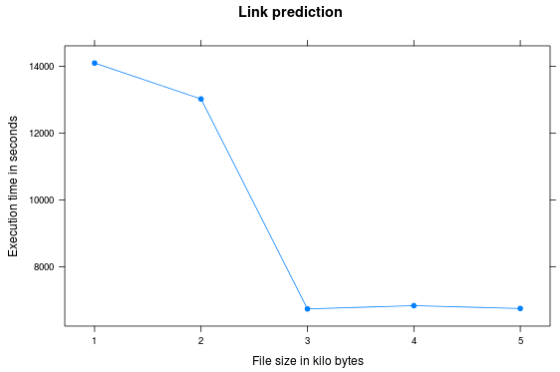

Figure: 1. Link Prediction performance for variable number of slaves.

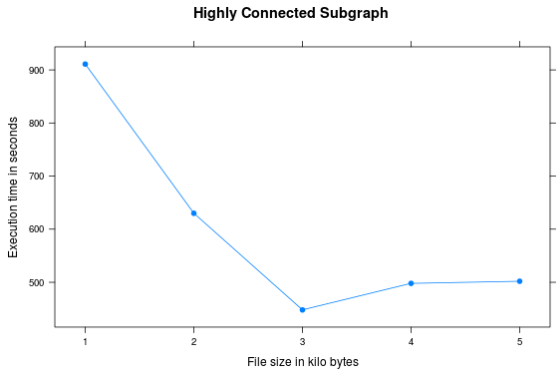

Figure: 2. HCS cluster performance for variable number of slaves

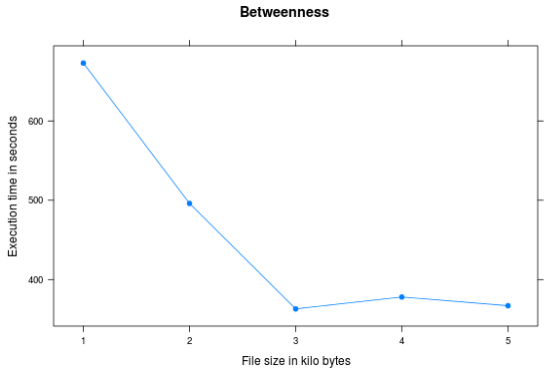

Figure: 3. Betweenness performance for variable number of slaves

As shown in the Figure 1, 2, and 3 as the number of slaves increases in a cluster, the time of execution decreases. At slaves 4 and 5 we can observe that time of execution of HCS, Link Prediction and Betweenness increases because of the communication delay between the slaves means if we start adding number of slaves to the cluster, the performance of the application degrades and will not result in any improvement in terms speed of execution

B. Experiment 2:

Second experiments carried out with the constant number slaves in a cluster, that is three slaves in a cluster and with the variable file size. The time of execution is measured for each operation. From the below Table 3, We can confirm that as the size of the file increases time of execution also increases for the constant number of slaves in a cluster. From the Figure 4, 5 and 6, we can confirm that as the file size increases, the execution time will increases almost linearly for this dataset.

Table 3: Variable file size with constant three slaves

RHADOOP RESULTS					
Execution Time(Seconds)					
Technique	File Size in KiloBytes				
	100	200	300	400	500
PageRank	10	10	11	11	10
Prediction	611	3289	5108	6165	6740
HCS	165	231	352	408	448
Clique	14	35	50	49	67
Kspan	12	14	13	13	13
Betweenness	121	232	298	330	363

From the Table 3, we can confirm that as the size of the file increases time of execution also increases for the constant number of slaves in a cluster.

From the Figure 4, 5 and 6, we can confirm that as the file size increases, the execution time will increases almost linearly for this dataset.

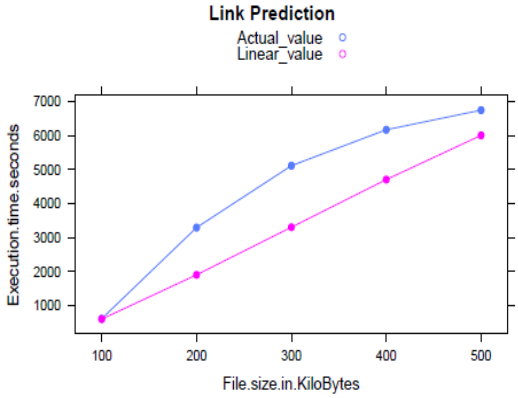

Figure: 4. Link Prediction performance for variable file size

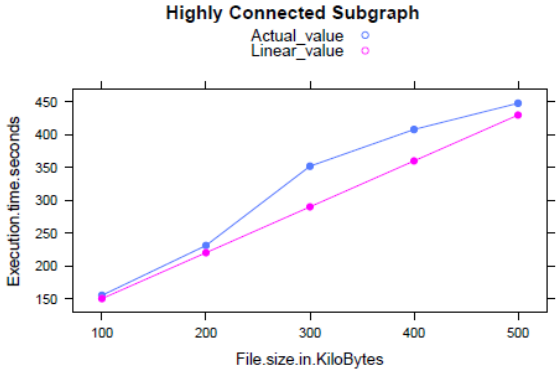

Figure: 5. Highly connected Subgraph performance for variable file size.

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
Vol. 2, Issue 5, May 2015

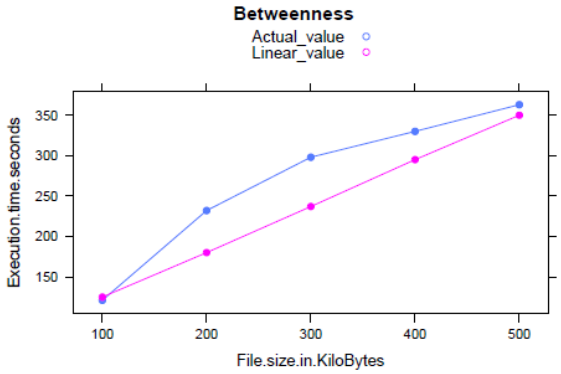

Figure: 6. Betweenness performance for variable file size

As the file size increases, in this case from 100kb to 500kb the actual time taken for execution varies the most with the linear value. This difference reduces gradually and remains constant for 200kb and 300kb. The difference reduces to half for 400kb and further reduces for 500kb. It can be inferred that as the data size increases, the difference will become negligible which shows the adherence to linear increase of execution time for larger data sets.

The equation of the approximated linear line and constant difference between any ith and (i+1)th operation for the given values as shown in the below Table 4.

Table 4: Approximate linear equation and constant difference for ith and (i + 1)th iteration.

Technique	Linear equation y= mx+c	Difference
Prediction	y=15.13x-157.6	1513
HCS	y=0.763x+89.9	76.3
Betwenness	y=0.582x+94.2	58.2

Give the execution time of varying data sizes the execution time of ith data size has a relationship with (i+1)th value as given below.

(i + 1)th Execution Time = (i)th Execution Time * (1+Percentage of increase or decrease)

V. CONCLUSION

SNA is gaining attention due to the growth of online social network interactions and exploring of information involving such interactions. The main concern is analyzing the interactions of the social network, fetch and understand the hidden information and perform a knowledge discovery on the FOSS data.

Influence spotting in the network are found effectively using a PageRank algorithm and most influential developers are found. future links can be found using a Link Prediction algorithm for the developer-developer relation dataset. Clustering is done using HCS, Kspan, Maximal Clique and Betweenness Centrality techniques to gain more insights such as average number of developers working together, maximum number of developers working together, minimum number of developers working together.

As part of the environment performance, our study confirms that increase in size of the file increases the time of execution for constant number of slaves and if we add more number of slaves to cluster, there will be a performance degradation in the environment because of communication delay between the slaves/systems and another study confirms that as the data size increases, there is a linear increase of execution time for larger data sets.

Many data mining techniques used aims at deriving useful information from the dataset containing very basic interactions among developers. Analyzing such data for more information like the top active developers in the entire network, predicting the possible interests of the developers based on their history, grouping the developers together to analyze betweenness and to find the groups with similar interests of developer.

VI. FUTURE WORK

The first area is to re-design the big graph analytic platform to provide fast and scalable computation infrastructure. A challenge for the direction is to balance the speed and scalability. MapReduce is a disk based system, and thus it is scalable and robust while it is not optimized for speed[16].

The second area is to transform existing serial algorithms into distributed algorithms. A challenge here is to remove dependency in serial algorithms so that the resulting distributed algorithms run in parallel [16].

The third area is to improve visualization and understanding of graphs. A graph forms a complicated object with many interactions between nodes. Visualization of graphs helps us better understand the structure and the interactions in graphs. The challenge is to effectively summarize the graphs so that users can easily understand the graphs in a screen with limited resolution [16].

The work can be continued to be implemented on various other distributed platforms to check as to what would be the best choice for graph mining. The work can also be extended to implementation on Graphical Processing Unit (GPU) in order to assess scalability.

REFERENCES

[1] Renee Tynan, Vince Freeh and Gregory Madey. Research project on the free and open source software (FOSS) development phenomenon, Available at: http://www3.nd.edu/oss/Data/data.html [online] Last Accessed : 23-Mar-2015.

[2] Vincent Freeh, Gregory Madey and Tynan Renee. The open source software develop ment phenomenon an analysis based on social network theory. page 247, 2002.

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology
Vol. 2, Issue 5, May 2015

[3] MM Mahbubul Syeed and Imed Hammouda,Who contributes to what? exploring hidden relationships between floss projects. In Open Source Software: Mobile Open Source Technologies, pages 21–30. Springer, 2014.
[4] Tang, Lei and Liu, Huan. Graph mining applications to social network analysis, Managing andMining Graph Data. pages 487–513, Springer 2010.
[5] Yajun Wang Wei Chen and Siyu Yang.Efficient influence maximization in social networks. pages 1–9, 2002.
[6] Ashish Jain, Rajeev Sharma, Gireesh Dixit, and Varsha Tomar. Page ranking algorithm in web mining, limitations of existing methods and a new method for indexing web pages. In Communication Systems and
Network Technologies(CSNT), 2013 International Conference on, pages 640– 645. IEEE,2013
[7] Rogers, Ian. The Google Pagerank algorithm and how it works. 2002
[8] Tsuyoshi Murata and Sakiko Moriyasu. Link prediction of social networks based on weighted proximity measures. In Web Intelligence, IEEE/WIC/ACM international conference on, pages 85–88. IEEE, 2007.
[9] Johnson, Stephen C. Hierarchical clustering schemes, journal Psychometrika Volume 32, pages 241-254, Springer 1967.
[10] Modani, Natwar and Dey, Kuntal. Large maximal cliques enumeration in sparse graphs, Proceedings of the 17th ACM
conference on Information and knowledge management.pages1377–1378, ACM 2008.
[11] Raidl, Gnther R and Julstrom, Bryant A. A weighted coding in a genetic algorithm for the degree-constrained minimum spanning tree problem, Proceedings of the 2000 ACM symposium on Applied computing- Volume 1, pages 440-445, ACM 2000.
[12] Samatova, Nagiza F and Hendrix, William and Jenkins, John and Padmanabhan, Kanchana and Chakraborty, Arpan. Practical Graph Mining with R, CRC Press 2013.
[13] Betweenness and centrality, available at: http://en.wikipedia.org/wiki/betweenness centrality [online] last accessed : 08-Apr- 2015.
[14] LongJason Lu and Minlu Zhang. Edge betweenness centrality. In Encyclopedia of Systems Biology, pages 647–648. Springer, 2013.
[15] Vignesh Prajapati. Big data analytics with R and Hadoop. Packt Publishing Ltd,2013.
[16] Kang, U and Faloutsos, Christos. Big graph mining: algorithms and discoveries, journal ACMSIGKDD Explorations Newsletter, volume 14 page 29-36, ACM 2013.
[17] Betweenness and centrality, available at: http://en.wikipedia.org/wiki/betweenness centrality [online] last accessed : 08-apr- 2015.
[18] LongJason Lu and Minlu Zhang. Edge betwenness centrality. In Encyclopedia of Systems Biology, pages 647–648. Springer, 2013.
[19] Hcs clustering, available at: http://en.wiki pedia.org/wiki/hcsclustering algorithm.