Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/081L49

Social Network Analysis To Detect Inherent Communities based On Constraints

Debnath Bhattacharyya[1,∗], *Soumita Seth*[2,∗] *and Tai-hoon Kim*[3,∗]

[1] Department of Computer Science and Engineering, FET, NSHM Knowledge Campus, Durgapur-713212, West Bengal, India
[2] Center for Softcomputing Research, Indian Statistical Institute,Kolkata-700108, West Bengal, India
[3] Dept. of Convergence Security, Sungshin Women's University, 249-1, Dongseon-dong 3-ga, Seoul, 136-742, Korea

Received: 25 May. 2013, Revised: 19 Sep. 2013, Accepted: 20 Sep. 2013
Published online: 1 Apr. 2014

Abstract: Social network analysis (SNA) is used to analyze social networks or structures made up individuals called nodes, which are tied by one or more specific types of interdependency such as relationships, connections, or interactions. Often it is used in many internet-based applications like, social networking websites, on-line viral marketing, and recommendation network based applications to improve the performance of user-specific information dissemination. Detecting communities, which are basically sub-graphs or clusters, within a social network has been the central focus of this work. Here, we present a divisive hierarchical clustering algorithm for detecting disjoint communities by removing minimum number of edges to obey minimum edge-cut principle, like CHAMELEON: Two Phase Agglomerative Hierarchical Clustering. The stopping criteria of this algorithm depends on two threshold constraints namely, balance constraint (BC) and MINSIZE (MS) like CHAMELEON. As a measure of the quality of community, we follow network centrality measure clustering coefficient. Our experimental results, using some well-known benchmark social networks, also show that our method determines similar communities with good average clustering coefficient as the other existing well known methods of various research papers.

Keywords: Social network analysis, Community detection, Hierarchical clustering, Minimum edge-cut principle, Clustering coefficient

1 Introduction

Networks are ubiquitous in the scientific models around us. A social network perspective is employed to model the structure of a social group, how this structure influences other variables, or how structures change over time. The study of these structures uses methods in social network analysis to identify influential nodes, local and global structures, and network dynamics. Social networks and the analysis of them is an inherently interdisciplinary academic field which emerged from social psychology, sociology, statistics, and graph theory. Jacob Moreno is credited with developing the first socio grams in the 1930s to study interpersonal relationships as structures in which people were points and the relationships between them were drawn as connecting lines. These approaches were mathematically formalized in the 1950s and theories and methods of social networks became pervasive in the social and behavioral sciences by the 1980s [3]. In recent years, growth of web-based communication has triggered an enormous amount of social interactions among the users on Internet. This data can generate a large social network with millions or even billions of interactions. Such social networks are difficult to handle because of its large size but consist of interesting inherent properties like presence of social communities and the structures of those communities. Generally we represent social network by a graph which can be defined by $G = (V,E)$, where $V = \{v_i : 1 \leq i \leq n\}$ is the set of nodes and E is the set of edges or links. We write $n = |V|$, $m = |E|$, $e_{ij} = (v_i, v_j) = (v_j, v_i)$ for the undirected edge connecting v_i and v_j $(\neq v_i)$, $Nghb(v_i) = \{v_j : e_{ij} \in E\}$ and $d_i = degree(v_i) = |Nghb(v_i)|$ [4].

Cluster analysis or clustering is the task of assigning a set of objects into groups (called clusters) so that it can maximize the intra-cluster similarity and minimize the inter-cluster similarity [2]. Clustering is a main task of explorative data mining, and a common technique for

[∗] Corresponding author e-mail: debnathb@gmail.com, soumita.seth@gmail.com, taihoonn@daum.net

© 2014 NSP
Natural Sciences Publishing Cor.

statistical data analysis used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, and bio informatics. Cluster analysis itself is not one specific algorithm, but the general task to be solved. It can be achieved by various algorithms that differ significantly in their notion of what constitutes a cluster and how to efficiently find them. Popular notions of clusters include groups with low distances among the cluster members, dense areas of the data space, intervals or particular statistical distributions. Cluster analysis as such is not an automatic task, but an iterative process of knowledge discovery or interactive multi-objective optimization that involves trial and failure. It will often be necessary to modify preprocessing and parameters until the result achieves the desired properties. Researchers coming from the fields of data mining and machine learning often use same algorithms but have different goals.The various types of clustering are: [5]Hierarchical or nested clustering, Partitional or unnested clustering, Exclusive clustering, Overlapping clustering, Fuzzy clustering, Complete clustering, Partial clustering.

The most important clustering techniques are Partitional and hierarchical clustering. The division of a set of data objects into non-overlapping subsets or clusters where each data object resides in exactly one subset, this clustering technique is known as partitional clustering. Hierarchical clustering permits cluster to have sub clusters. In hierarchical clustering we can organize a set of nested clusters as a tree, such that each node (cluster) of the tree, excluding leaf nodes, is the union of its children (sub clusters), the root node is nothing but the cluster containing all objects. Sometimes, the leave nodes are singleton clusters of individual data objects.We can generate hierarchical clustering by two basic approaches [5].

In Agglomerative approach, we start with all points as individual clusters and, at each step, merge the closest pair of clusters. Define the notion of cluster proximity is needed in this approach [5]. In Divisive approach, we start with one, all-inclusive cluster and at each step, split a cluster until only singleton clusters of each individual points found or reach to a certain number of clusters which is desired. At each step, which cluster to split and how to do the splitting are completely our decision [5]. Fig. 1. represents Agglomerative and Divisive Hierarchical Clustering. As we represent social network by a graph, we need to follow graph based clusters where a cluster is nothing but a connected component which is a groups of objects that are connected to one another, but there is no connection to objects outside the group. Clique is a graph based cluster i.e., a set of nodes in a graph that are completely connected to each other. When we add connections between objects in the order of distance between each-other, a cluster is formed if the set of objects forms a clique. Existing clustering algorithms are K-means, PAM [5,2], CLARANS [10,2], DBSCAN, CURE, ROCK, CHAMELEON [5,2] which we will

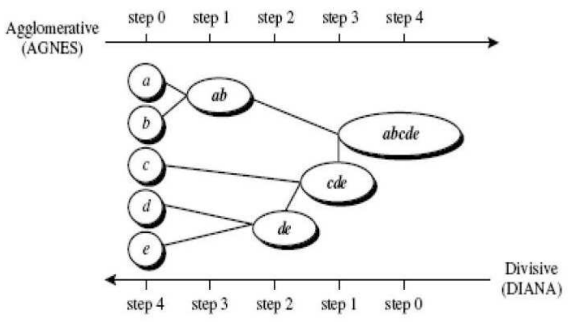

Fig. 1: Agglomerative And Divisive Hierarchical Clustering

discuss in next section. Apart from these, for detecting communities in social network we saw some well known methods in various research papers. From those methods we choose two methods, Girvan Newman which is based on edge betweeness [14] and Strength of Link which is an informal notation of Granovetter, based on social ties [18], for comparing our results. We will also discuss above two methods in next sections.

2 Related Works

Partitional clustering attempts to break a data set into K clusters such that the partition optimizes a given criterion. K-means is a centroid-based approach, its goal is assign points to a specific cluster such that the mean square distance between points and the centroid of the assigned cluster is minimized. Centroid-based approaches are only suitable for Euclidean space where centroid computation of a given set of points is possible [2]. K-Medoid is a medoid-based approach, chooses data points as center, which is termed as medoid, nothing but the object of cluster which is most similar with all the objects in the cluster and most centrally located point of dataset, and minimized the squared error, distance between points in cluster and its medoid. It is more robust to noise and outliers than K-means because of minimizing a sum of pair wise dissimilarities instead of square Euclidean distances [7]. Partitioning Around Mediods (PAM) is nothing but a K-medoid algorithm. It generally works on data similarity that means data in an arbitrary similarity space find medoid and minimizes the sum of distances of points from their closest medoid [2]. Clustering Large Applications Based on R A N domized Search (CLARANS) is nothing but a K-medoid algorithm which finds the medoid based on a randomized search of a graph to represent the clusters. The major drawback of partitional techniques which are they fail in such case where points in a given cluster are closer to the center of another cluster than to the center of their own cluster, it may happen for large variation in cluster sizes or if the

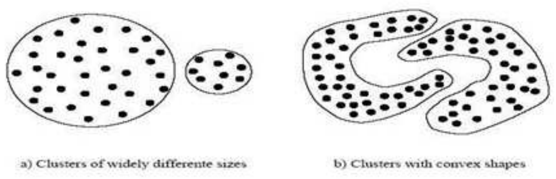
a) Clusters of widely differente sizes
b) Clusters with convex shapes

Fig. 2: Datasets on which centroid and medoid approaches fail

cluster shapes are convex (refer Fig 2.) [2].

Single link or MIN is an agglomerative hierarchical clustering. In this algorithm, at first individual data points are represented by each cluster and then proximity of two clusters is computed by the minimum of the distance or maximum of the similarity between any two points of two different clusters [5]. Unlike single link, in complete link the proximity of two clusters is computed by maximum of the distances and minimum of the similarity between any data points in two different clusters. In this method a group of points can form a cluster only if they are completely linked, i.e., form a clique [5]. Compared to single link, it is less sensitive to noise and outliers but it can break large clusters and it prefers the globular shapes of clusters [5]. DBSCAN is a density based clustering algorithm. It locates regions of high density that are separated from one another by region of low density [5]. It assumes that all points within genuine clusters are density reachable and points across different clusters are not. It is not sensitive to noise and can handle clusters of arbitrary shapes and sizes. It can find more clusters than K-means. However, it faces trouble when the clusters have widely varying densities and in high dimensional data where density is more difficult to define [5].

Clustering Using Representation (CURE) is a hierarchical clustering algorithm which uses a variety of different techniques to create a approach for handling large data sets, outliers and clusters with non-spherical shapes and non-uniform sizes [5]. CURE represents each cluster via multiple representative points instead of using a single centroid to represent a cluster. Here, the similarity between two clusters is measured by the similarity of the closest pair of the representative point belonging to different clusters. For the merged clusters new representative points are determined by selecting a constant number of well scattered points from all the data points and according to a shrinking factor, shrinking them towards the centroid of the cluster [2]. Only the minimum distance between the representative points of two clusters is considered but the aggregate interconnectivity among the two clusters is not considered by the selection mechanism of CURE [2].

ROCK is a recently developed agglomerative clustering which can operates on a derived similarity graph; it scales the aggregate interconnectivity among pairs of items belonging to different clusters with respect to a user-specified interconnectivity model. The sub clusters belonging to the same cluster will tend to have high interconnectivity.However,the aggregate interconnectivity between two clusters are depends on size of clusters, generally cluster with large size will have higher interconnectivity. ROCK can overcome this problem by normalizing aggregate similarity between a pair of clusters respect to a user specified static interconnectivity model and ignores the potential variations in the inter-connectivity of different clusters within the same data set [2]. ROCK can breakdown if the choice of parameters in the static model is incorrect or it is not adequate to capture the characteristics of clusters with respect to given data set. Though it scales aggregate interconnectivity across the pair wise clusters with respect to a static interconnectivity model, but it ignores the value of the strongest edges across clusters that mean information about closeness between two clusters [2].

Most of above described algorithms breakdown when the data consists of clusters that are of diverse shapes, densities, and sizes. To overcome those problems one existing renowned hierarchical clustering algorithm is CHAMELEON [2,5] that measures the similarity of two clusters based on a dynamic model. The clustering process, which is used in CHAMELEON, not only consider the inter-connectivity and closeness (proximity) between two clusters but also consider the internal inter-connectivity of the clusters and closeness of items within the clusters and if both are comparable then two clusters are merged. CHAMELEON is efficient in a number of data sets that contains points in 2D space, and contain clusters of different shapes, densities, sizes, noise and artifacts [2].

CHAMELEON represents the set of data items as a sparse graph based on k-nearest neighbour graph approach [5,2]. Using this sparse graph representation of dataset CHAMELEON can scales large data sets and operates on dataset successfully that are available in similarity space and not in metric spaces. CHAMELEON clustering algorithm consist of two phases. In first phase, using a graph partitioning algorithm CHAMELEON cluster the data items of K-nearest neighbour graph into large number of relatively small sub-clusters. In second phase, it uses an agglomerative clustering algorithm to obtain the optimal clusters by repeatedly combining together these sub-clusters (refer Fig. 3) [2,5]. CHAMELEON can overcome the drawbacks of CURE and ROCK by determining the pair of most similar sub-clusters taking into account both the inter-connectivity as well as the closeness of the clusters. Beside this, CHAMELEON uses novel approaches to model the degree of inter-connectivity and closeness between each pair of clusters that takes into account the

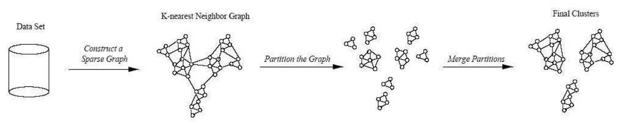

Fig. 3: Overall framework CHAMELEON

internal characteristics of the clusters themselves.
In this paper we present a Divisive Hierarchical Algorithm whose idea is based on the partition technique of first phase of CHAMELEON [5,2]. For finding the initial sub clusters CHAMELEON uses a graph partitioning algorithm to partition the k-nearest neighbour graph of the data set into a large number of partitions in such a way by which the edge cut i.e. the sum of the weight of the edges whose removal give partitions, is minimized..We know, the each edge of k-nearest neighbour graph represents the similarity among data points, so minimizing edge cut implies effectively minimizes the relationship among data points across the resulting partitions. So, the data points which reside in same partition are highly related to each other.
For partitioning a graph into several sub clusters CHAMELEON uses a graph partitioning algorithm which is part of hM_ETiS library, it can produce high quality partitioning for a wide range of unstructured graph and hyper-graphs. Using hM_ETiS CHAMELEON split a cluster C_i into two sub-clusters C_i^A and C_i^B such that edge cut between C_i^A and C_i^B is minimized and each of these two sub-clusters have to be consisted of at least 25% of the nodes which are present in cluster C_i, this lower bound conditions is termed as *balance constraint* (BC), an integral part of using a graph partitioning approach for finding sub-clusters. hM_ETiS can effectively find out bisection where edge cut is minimized within the specified *balance constraints*, which can compel hM_ETiS to break into a natural cluster [2].
CHAMELEON initially starts with all the points in a same clusters and bisect it using hM_ETiS and repeatedly selects the largest sub-cluster among the current set of sub-clusters to bisect if *balance constraint* condition is satisfied. This process terminates when the larger sub-cluster contains fewer than a specified number of vertices, termed as MINSIZE. MINSIZE generally controls the granularity of the initial clustering solution.If MINSIZE should be a smaller value than size of most clusters that we expect to find from the data set. However, if we set a large value as MINSIZE it can gives us sub-clusters contains large number of nodes which allow us to evaluate the inter-connectivity and closeness of the items in each sub-cluster in a meaningful fashion. Generally setting MINSIZE to about 1% to 5% of the overall data points worked good enough. CHAMELEON is efficient to cluster spatial data also in presence of noise and outliers and it can fairly work on clusters of different shapes, sizes and density. CHAMELEON assumes that after sparsification and graph partitioning produced partitions are in most cases sub-clusters. That means, most of the points in a partition belong to a true cluster [2].
The central goal of this paper is to detect community from a social network by using a divisive hierarchical clustering method. The other goals are finding out optimal communities or clusters by measuring the quality of a community or a cluster using notion of *clustering coefficient* [8]. We also want to present an algorithm which will be able to find out clusters when no or partial prior knowledge is available about similarity degree function of the network. Beside these, we compare our result with two well known community detection methods Girvan Newman (GN), Strength of Link (SOL).
Girvan Newman (GN) algorithm is based on edge betweenness. Vertex betweenness has been studied in the past as a measure of the centrality and influence of nodes in networks. For any node i, vertex betweenness is defined as the number of shortest paths between pairs of nodes that run through it. It is a measure of the influence of a node over the flow of information between other nodes, especially in cases where information flow over a network primarily follows the shortest available path. The GirvanNewman algorithm extends this definition to the case of edges, defining the edge betweenness of an edge as the number of shortest paths between pairs of nodes that run along it. If there is more than one shortest path between a pair of nodes, each path is assigned equal weight such that the total weight of all of the paths is equal to unity. If a network contains communities or groups that are only loosely connected by a few intergroup edges, then all shortest paths between different communities must go along one of these few edges. In a complete graph, average values for both node and edge betweenness are zero, as all the nodes are directly connected to all the other nodes. Newman proposed modularity to measure goodness of a community or a module inside a network. The idea behind modularity is to observe how random re-arrangement of edges (keeping degrees of nodes same) in a network changes the arrangement of links inside a community with respect to the network [13]. Girvan-Newman (GN) algorithm [14, 15] remains a popular divisive hierarchical clustering algorithm. GN algorithm calculates the edge betweenness for all edges of a network and removes the edge with the highest edge betweenness. In the next step it recalculates edge betweenness for all the edges that has been affected by the removal of the edge in the previous step and then again removes the edge with the highest edge betweenness. The algorithm's steps for community detection are summarized below
1. The betweenness of all existing edges in the network is calculated first.
2. The edge with the highest betweenness is removed.
3. The betweenness of all edges affected by the removal is

recalculated.
4. Steps 2 and 3 are repeated until no edges remain.

The fact that the only betweennesses being recalculated are only the ones which are affected by the removal, may lessen the running time of the process' simulation in computers. However, the betweenness centrality must be recalculated with each step, or severe errors occur. The reason is that the network adapts itself to the new conditions set after the edge removal. For instance, if two communities are connected by more than one edge, then there is no guarantee that all of these edges will have high betweenness. According to the method, we know that at least one of them will have, but nothing more than that is known. By recalculating betweennesses after the removal of each edge, it is ensured that at least one of the remaining edges between two communities will always have a high value.
The idea of communities in a social network goes back to Granovetters hypothesis which states that, the people or nodes inside a community are strongly linked with each other whereas the individuals or the nodes that do not belong to the same community are connected only by weak links. But Granovetter did not come up with an empirical definition for measuring the strength of a tie in a network from a social network analysis perspective and nor did he use it for community detection. However, according to M. Granovetter [16] if we remove the weak links then the connected component of the remainder of the social network will correspond to the communities. Put another way, the challenge to determine the communities in a social network is to come up with a suitable method for defining the strength of a link in a social network. Clearly, the strength of a link must account for both the local and global structure of the network. Therefore, links to or from a person can be a mix of both strong and weak links. In extreme cases, all of them are either strong links or weak links. Marsden et al. [17] indicates that the total time spent in communication between two individuals during a given period of time is a suitable measure to represent the strength of the link between them. A perfect example for this case would be a call graph (a graph made out of call records of individuals calling each other) where, for a given period time (say, a week or a month) the talk time between any pair of nodes varies. Hence, such edges could be assigned weights based on the talk time [18] .
In [18] to formulate the strength of the links the number of connections between two adjacent communities and their respective average clustering coefficients are used. Lesser the number of the links between two groups of nodes they are more likely to be distinctive. The first part of the formulation of strength of a link consists of calculation of the number of connections between the neighbors of the nodes of the link. Lower the number of connections, weaker is the link in term of strength. The link itself is not counted towards the number of connections. The second part consists of calculation of the average clustering coefficient of the neighbors of the nodes of the link. Higher the average clustering coefficient value of the neighbors, more connected the neighbors are and weaker is the link in terms of strength. While calculating average clustering coefficient, the nodes of the links are excluded from each others list of neighbors as the primary goal is to find the cliquishness among the nodes that are separated by the link. SOL detects disjoint communities based on strength of link with more cliquish than those produced by GN method with recalculation.
The rest of the paper is organized as follows. Section 3 describes primary goal of this paper. We present our new clustering algorithm in Section 4. Section 5 gives the experimental results. Section 6 contains conclusions and directions for future work.

3 Primary Goal

At first we have selected social networks with prior knowledge about the groups. Then we have compared the groups or clusters produced by our algorithm with the benchmark data. Our primary goal is to detect clusters representing communities in the social network. The networks have been represented in form of undirected and unweighted graph $G = (V,E)$ where V is the number of vertices and E is the number of edges. In CHAMELEON [2], this is a two phase clustering algorithm, we have seen that in phase 1 to find initial sub-clusters they primarily use hM_ETiS to split a cluster C_i into two sub-clusters C_i^A and C_i^B such that the edge-cut between C_i^A and C_i^B is minimized, which is called as *Minimum edge-cut principle*. In this paper, we introduced a divisive hierarchical clustering as a replacement of hM_ETiS but we have not implement agglomerative method of CHAMELEON since almost all points are placed true cluster after hM_ETiS [2]. But some properties of hM_ETiS has changed in our divisive algorithm to get maximum number of true clusters. Here we have used *adjacency-list* representation to represent the graph, i.e. $adjList[G]$. Using $adjList[G]$ we can find the neighbors or adjacent vertices for each vertex $v \in V(G)$. From the neighbors list of each vertex we have found the list of common neighbors between a pair of vertices. Then we have cut or removed the edges between such pair of vertices which have no or lesser number of common neighbors. In this way we can find out clusters by removing minimum number of edges. After removing edges we have used depth-first search traversal operation [6] to find out a connected component of graph G. Our aim is to keep every vertex in such group where it has maximum number of neighbors. This means each partition or cluster has to be nothing but a dense sub-graph and hence produce good quality clusters. Like in CHAMELEON clustering algorithm, we have used a local constraint or threshold value and using this threshold value we can specify the minimum percentage of vertices of a cluster that have to be present in a sub cluster. It has been termed as *balance constraint* (BC). There is also a global constraint which can specify the minimum percentage of total vertices of a graph that have to be present in a cluster, referred to as MINSIZE (MS). *Balance constraint*(BC) and MINSIZE (MS) are nothing but the stopping criteria. In experimental results of CHAMELEON algorithm we have already seen that they fixed the *balance constraints*(BC) to 25% and varied the MINSIZE (MS) from 2% to 5% to observe the result [2]. In our algorithm, we have varied both local and global constraints and observed the result in each time. Here, we have used notion of average clustering coefficient to measure the correctness of clusters. We will discuss our algorithm in next section. We need to calculate average local clustering coefficient for all sub graphs of each iteration.

Clustering coefficient is nothing but a measure of degree to which nodes in a graph tend to cluster [8]. Clustering coefficient has two types: Local Clustering Coefficient and Global Clustering Coefficient. Here we will discuss Local Clustering Coefficient.

Local Clustering Coefficient calculates the probability of number of adjacent nodes are connected with a single node. It is calculated by the fraction of permitted edges between the neighbors of a single node to them that actually exist for these neighbors. Three nodes can build a triple if a node is connected to an unordered pair of other nodes (neighbors). The number of triples refers the number of permitted edges between the neighbors of a node. When two neighbors of a node are also connected by an edge then these three nodes build a triangle. Counting the number of triangles reveals the number of edges that actually exist for the neighbors of a certain node: [9]

$$C_i = \frac{number\ \ of\ \ triangles\ \ connected\ \ to\ \ node\ \ i}{number\ \ of\ \ triples\ \ centered\ \ on\ \ node\ \ i} \tag{1}$$

Fig. 4. represents local clustering coefficient.

According to Watts and Strogatz, the clustering coefficient for whole network is defined by the average of local clustering coefficients of all the vertices *n*: [8]

$$\bar{C} = \frac{1}{n}\sum_{i=1}^{n} C_i \tag{2}$$

In our experiment, we have used notion of network average clustering coefficient because the goal of cluster analysis is that the objects which belong within a group form a dense region by their closeness that means objects in same clusters are strongly connected and clusters are separated by a sparse region. So, we have used network average clustering coefficient as quality function to measure the goodness of clusters on basis of benchmark.

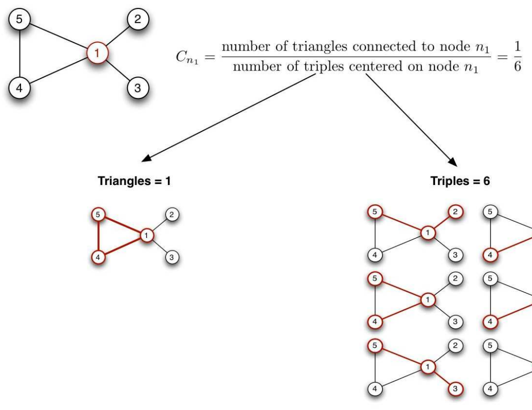
$C_{n_1} = \frac{\text{number of triangles connected to node } n_1}{\text{number of triples centered on node } n_1} = \frac{1}{6}$
Triangles = 1
Triples = 6

Fig. 4: Local Clustering Coefficient

4 Finding Communities from a Social Network

Communities, in a social network, are generally defined as a group of nodes that are densely connected. The density in some cases has been defined by degrees [11] and in some cases by its cliquishness, i.e, its average clustering coefficient [8]. As cliquishness denotes the true coherence between all the nodes of a group, we choose clustering coefficient as our measure to determine the quality of a community.

In social networks, we find less number of large communities and many small communities [12]. In social networks, two communities are connected by only a few edges. Generally, a good cluster S has many edges internally and few edges pointing outside which is obtained by a simplest objective function *Conductance*. *Conductance* is denoted by $\phi(S)$, defined by

$$\phi(S) = \frac{Number\ \ of\ \ edges\ \ outside\ \ S}{Number\ \ of\ \ edges\ \ inside\ \ S} \tag{3}$$

Fig. 5. shows that small conductance corresponds to good clusters. That means number of edges inside a cluster always greater than outside the cluster. We always divide a network by clusters through a sparse region to get good clusters. It can justify our claim, detecting communities from social networks by removing minimum number of edges so that clusters will be a dense region separated by sparse region.

In this paper we have proposed a recursive hierarchical divisive clustering algorithm to detect clusters and used

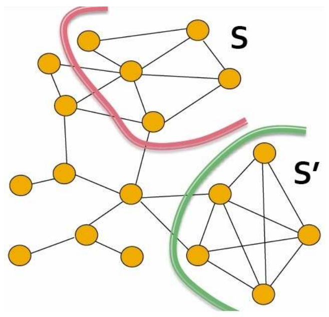

Fig. 5: Small conductance corresponds to good clusters

notion of average clustering coefficient to measure the correctness of clusters. Here we will discuss our algorithm:

4.1 Recursive Hierarchical Divisive Clustering Algorithm

At first, we will define all the terms which we have used in our algorithm

$MINSIZE(MS) \leftarrow$ Minimum percentage of the total number of vertices of graph G have to be present in clusters.
$adjM(G) \leftarrow$ Adjacency matrix of graph G
$NCM(G) \leftarrow$ Numbers of neighbors for each vertex $v \in V_G$
$adjL(G) \leftarrow$ Adjacency list of graph G
$CNM(G) \leftarrow$ Number of common neighbors for a pair of verices $u,v\ \forall u,v \in V_G$
$min \leftarrow$ Minimum number of neighbors between a pair of vertices.
$resultM(G) \leftarrow$ Result of a mathematical operation by adjM(G) and CNM(G)
$BalanceConstraints(BC) \leftarrow$ Minimum percentage of vertices in a specific cluster of G have to be present in sub-clusters
$NgB_v \leftarrow$ Set of neighbors for each vertex $v, \forall v \in V_G$
$DEL \leftarrow$ Set of distinct elements of resultM(G) which are greater or equal 1
$CL1(G) \leftarrow$ Set of vertices in first partition/cluster of Graph.
$CL2(G) \leftarrow$ Set of vertices in second partition/cluster of Graph.
$DFSSourceSelection \leftarrow$ Select a vertex by which we can get maximal connected component through DFS
$DFSRecursive(adjL, Source, V) \leftarrow$ Performs DFS
$dfsVisit(v) \leftarrow$ Performs DFS Visit $\forall v \in V_G$

In our algorithm our goal is getting several clusters by following Minimum edge cut principal. For this purpose here we always try to take care about how many number of edges essential to remove to get proper clusters. Our claim is this number of edges should be minimum so that

Algorithm 1: Recursive Divisive Clustering Algorithm

```
Input:Graph G(V,E) Output: Set of Clusters
Begin
if $|V_G| > MS$ and $MS > 2$ then
    Compute $adjM(G)$, $NCM(G)$ and $adjL(G)$
    $\forall u,v \in V_G, CNM_{u,v} \leftarrow |NgB_u \cap NgB_v|$
    $min \leftarrow min\{|NgB_u|, |NgB_v|\}$
    $resultM_{u,v} \leftarrow adjM_{u,v} + \frac{1}{min} CNM_{u,v},\ \forall u,v \in V_G$
    Compute BalanceConstraint(BC)
    forall the $u,v \in resultM_{u,v}(G)$ do
        if $resultM_{u,v}(G) \geq 1$ then
            $i \longleftarrow |DEL|$,/*Is Element Exist? */
            while $i > 0$ and $resultM_{u,v}(G) \neq DEL_i$ do
                Decremented $i$ by 1
            end
            if $i = 0$ then
                /*Found Distinct Element */
                Incremented $|DEL|$ by 1
                $DEL_{|DEL|} \leftarrow resultM_{u,v}(G)$
            end
        end
    end
    Call MergeSort($DEL$)
    for $i = 1$ to $|DEL|$ do
        forall the $resultM_{u,v}(G) = DEL_i$ do
            $adjM_{u,v}(G) \leftarrow 0$ /*Edge cut          */
        end
        Compute $NCM(G)$ and $adjL(G)$
        $V_1 \leftarrow DFSSourceSelection(adjL, V_G)$ $CL_1 \leftarrow V_1$
        and $V_2 \leftarrow V - V_1$
        if $|V_2| < BC$ or $|V_2| < MS$ then
            Deallocate $adjL, NCM, CL_1$
        end
        else
            $V_{21} \leftarrow DFSSourceSelection(adjL, V_2)$
            $CL_2 \leftarrow V_{21}$
            if $V_{21} < V_2$ then
                $V_{22} \leftarrow V_2 - V_{21}$
                forall the $v \in V_{22}$ do
                    if $|NgB_v| \in CL_1 > |NgB_v| \in CL_2$
                    then
                        $CL_1 \leftarrow v$
                    end
                    else
                        $CL_2 \leftarrow v$
                    end
                end
            end
            if $|CL_1|, |CL_2| \geq BC$and $MS$ then
                two proper cluster found and break
            end
            else
                Deallocate $CL_1, CL_2, adjL, NCM$
            end
        end
    end
end
if $E(G) \neq NULL$ then
    Call RecursivelDivisiveClustering($G'$), $G' \subset G$
end
End
```

Algorithm 2: DFSSourceSelection

```
    Input  : Set of Vertice
    Output: Set of Vertice which is come by DFS operation
1   Begin
2   forall the $v \in V - u$ do
3       $Source \leftarrow u$, where $u \in V$ and $|NgB_u| \geq |NgB_v|$
4   end
5   $V' \leftarrow DFSRecursive(adjList, Source, V)$ where $V' \subset V$
6   Return $V'$
7   End
```

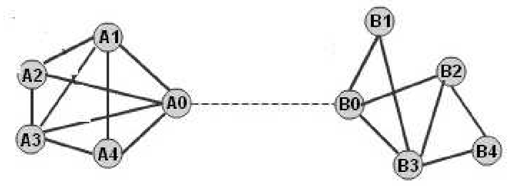

Fig. 6: Connected Graph with set of vertices

all communities are separated by sparse region . To achieve our goal here we used an objective function which is defined at Line 5 of our Algorithm 1. Here,

$$CNM(v_i, v_j) = |NgB(v_i) \cap NgB(v_j)| = d_{ij}. \quad (4)$$

So, we can conclude that d_{ij} represents how connected v_i and v_j are to their common neighbors. d_i and d_j represent the degree of v_i and v_j which is nothing but $|NgB(v_i)|$ and $|NgB(v_j)|$ respectively. So in mathematical notation our main objective function is

$$result_{ij} = \frac{d_{ij}}{min(d_i, d_j)} \quad (5)$$

When $result_{ij}$ gives minimum value that implies that we can split v_i and v_j into two clusters by removing minimum number of edges i.e. through sparse region. So for removing edges if we check this objective function in every iteration, we may achieve to our goal. Lets take an example, In Fig. 6, there is a connected graph G which have a set of vertice $V(G) = \{A_0, A_1, A_2, A_3, A_4, B_0, B_1, B_2, B_3, B_4\}$. Now we will apply our objective function on this graph. Here,

$$result(A_0, A_1) = \frac{d(A_0, A_1)}{min\{d(A_0), d(A_1)\}} = \frac{3}{4} = 0.75$$

This value is same for (A_0, A_2),(A_0, A_3),(A_0, A_4),(A_1, A_2), (A_1, A_3),(A_1, A_4),(A_2, A_3),(A_2, A_3),(A_3, A_4) as this subgraph is a completely connected graph. Now

$$result(A_0, B_0) = \frac{d(A_0, B_0)}{min\{d(A_0), d(B_0)\}} = \frac{0}{min\{4,3\}} = 0$$

You may check rest of vertice B_0, B_1, B_2, B_3, B_4 in same way but here minimum value is 0. So if we remove only one edge $e(A_0, B_0)$, we will get two connected components which are separated by sparse region.
For checking connectivity and cycle of the graph we used Depth First Search (DFS) operation (Refers Algorithm 3 & 4) [6]

Algorithm 3: DFSRecursive Algorithm

```
    Input  : Graph G(V,E)
    Output: A connected set of graph(G)
1   Begin
2   foreach $u \in V[G]$ do
3       $color[u] \longleftarrow WHITE$ /*Initial Color of
            vertices                                   */
4       $\pi[u] \longleftarrow NIL$ /*Predecessor of u          */
5       $time \longleftarrow 0$
6       foreach $u \in V[G]$ do
7           if $color[u] = WHITE$ then
8               Call dfsVisit($u$)
9           end
10      end
11  end
12  End
```

Algorithm 4: dfsVisit Algorithm

```
    Input  : vertex $u \in V(G)$
    Output: Connected vertices reachable from $u$
1   Begin
2   $color[u] \longleftarrow GRAY$ /*Vertex u is traversed */
3   $d[u] \longleftarrow time \longleftarrow time + 1$
4   foreach $v \in Adj[u]$ do
        /*Explore edge $(u, v)$                           */
5       if $color[v] = WHITE$ then
6           $\pi[v] \longleftarrow u$
7           Call dfsVisit($v$)
8       end
9       $color[u] \longleftarrow BLACK$ /*Blacken u; it is
            finished                                   */
10      $f[u] \longleftarrow time \longleftarrow time + 1$
11  end
12  End
```

5 Experimental Result

In this section we discussed our experimental result by two subsections. In first subsection we gave a glance on experimental environment and nature of inputs. In next subsection we elaborated outputs for each and every input and compare results with some well known benchmark methods.

5.1 Experimental Setup

We have written a C program for the experimental study, and executed it on a Compaq 510 laptop with 2.00 GHz Intel(R) Core(TM)2 Duo CPU T8570 microprocessor and 2 GB RAM memory, running Fedora (2.6.23.1-42.fc8). One more experiment has been performed for computing network average clustering coefficient, using cluster sets extracted from the various social network databases.
In our experiment, we have used three different social networks for detecting community from each network. Our first social network is Zachary's Karate Club Members group, which has 34 nodes and 78 edges. From this group data we have found different sets of clusters based on different *balance constraint*(BC) and MINSIZE(MS) which is said in CHAMELEON's first phase and apart from this we have followed min-cut edge principle which is also mentioned in CHAMELEON. Our algorithm has given us several sets of clusters in each iterations depending on different set of BC 's and MS 's . After getting all clusters we have computed average clustering coefficient for every step of iterations to check in which iteration our algorithm gives optimal result or good clusters. After that, we use our algorithm on two more social networks, one is Dolphins Social Network, which has 62 nodes and 159 edges and other is American Football Network, which has 115 nodes and 613 edges. Here we have to mention one thing, for every network it is not possible to use same set of BC and MS to get cluster sets, the determination of values of BC and MS is completely depends on nature of graph. So, for different graphs we may have to use different set of thresholds which we can choose by trial and error method.

5.2 Result For Different Data sets

Zachary Karate Club Data is a social network of 34 karate club members in an US university in 1970s. W. Zachary observed and recorded the interactions between individuals for 2 years and this social network data has been used for benchmarking in many papers related to community detection. The club got divided because of a conflict between administrator and instructor.In Fig. 7, we show a visualization of Karate Data.
For this network, we used different set of BC 's which are $\{15\%, 25\%\}$ and MS 's which are $\{3\%, 5\%\}$. In our experiment ,for MS 3% or 5% we varied BC from 15% to 25% and computed average clustering coefficient for every clusters record. Fig. 8. represents average clustering coefficient in each iteration of Zachary's Karate Club Members groups. In this figure, we are seeing that for every combination of BC and MS, the maximum average clustering coefficient lies in second iteration when number of clusters is 2 , the maximum average clustering coefficient is 0.686. Here we also compare average clustering coefficient of our clusters with the clusters record of well known benchmark algorithm Strength of

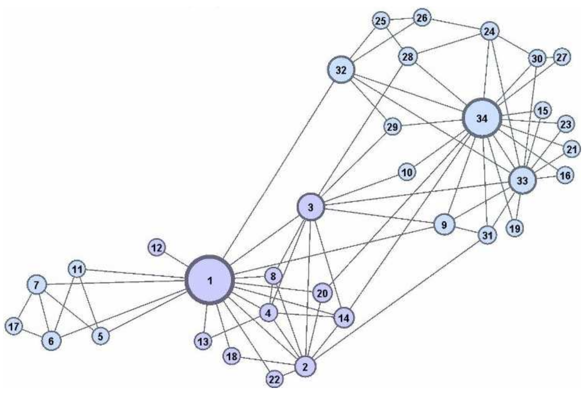

Fig. 7: Visualization of Karate Club Data

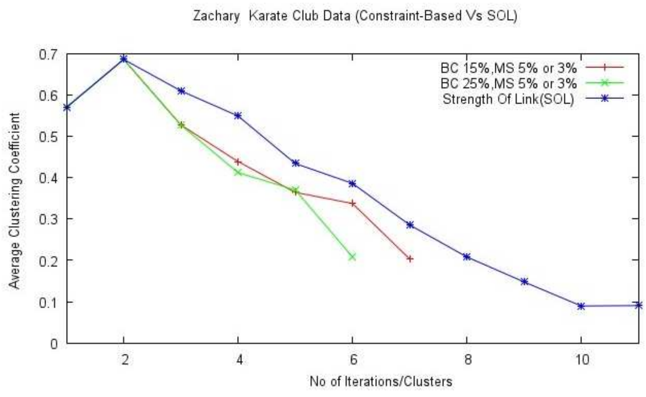

Fig. 8: Average Clustering Coefficient Plot of Zachary Karate Club Data and Compare With Benchmark Clustering Algorithm Strength Of Link(SOL)

Link(SOL). It's clear that our algorithm gives similar clusters of SOL. We can firmly say that the number of edges which we removed to get all clusters from this network is minimum compare to other algorithms.

Dolphins Social Network has 62 nodes and 159 edges. For this network we used a fixed MS 3% and a set of BC's which are $\{10\%, 15\%, 25\%\}$, computed average clustering coefficient for every clusters record.
Fig. 9. represents average clustering coefficient in each iteration of Dolphins Social Network. In this figure, we are seeing that when BC 10% and MS 3%, the maximum average clustering coefficient lies in eighth iteration when number of cluster is 8, the maximum average clustering coefficient is 0.566, when BC 15% and MS 3%, the maximum average clustering coefficient lies in sixth iteration when number of cluster is 6, the maximum average clustering coefficient is 0.589, when BC 25% and

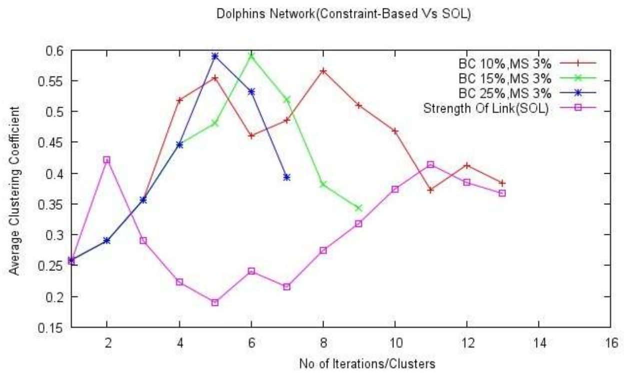

Fig. 9: Average Clustering Coefficient Plot of Dolphins Social Network and Compare With Benchmark Clustering Algorithm Strength Of Link(SOL)

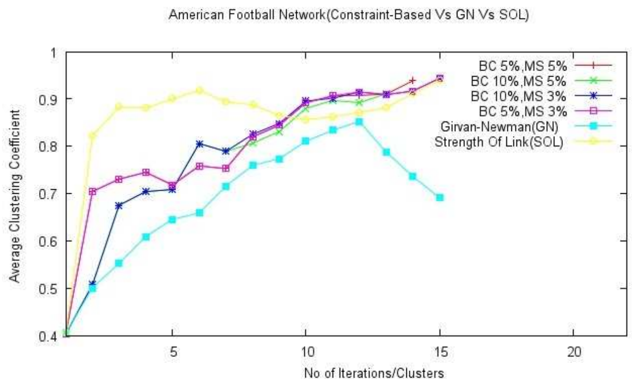

Fig. 10: Average Clustering Coefficient Plot of American Football Network and Compare With Benchmark Clustering Algorithms Girvan-Newman(GN) and Strength Of Link(SOL)

MS 3%, the maximum average clustering coefficient lies in fifth iteration when number of cluster is 5, the maximum average clustering coefficient is 0.589 Here we also compare average clustering coefficient of our clusters with the clusters record of well known benchmark algorithm Strength of Link(SOL). It's clear that our algorithm gives similar clusters of SOL. Almost in all iterations average clustering coefficient of communities detecting by our algorithm is greater that the communities detecting by SOL for this network.

American Football Network has 115 nodes and 613 edges. For this network we used a set of MS's which are $\{3\%, 5\%\}$ and a set of BC's which are $\{5\%, 10\%\}$, computed average clustering coefficient for every clusters record.

Fig. 10. represents average clustering coefficient in each iteration of American Football Network. In this figure, we are seeing that when BC 5% and MS 3%, the maximum average clustering coefficient lies in fifteenth iteration when number of cluster is 15, the maximum average clustering coefficient is 0.943, when BC 10% and MS 3%, the maximum average clustering coefficient lies in fifteenth iteration when number of cluster is 15, the maximum average clustering coefficient is 0.943, when BC 5% and MS 5%, the maximum average clustering coefficient lies in fourteenth iteration when number of cluster is 14, the maximum average clustering coefficient is 0.938, when BC 10% and MS 5%, the maximum average clustering coefficient lies in thirteenth iteration when number of cluster is 13, the maximum average clustering coefficient is 0.911. Here we also compare average clustering coefficient of our clusters with the clusters record of well known benchmark algorithms Girvan-Newman(GN) which is based on Edge Betweeness and Strength of Link(SOL). It's clear that our algorithm gives better clusters than GN and similar clusters of SOL. In SOL we also got maximum average clustering coefficient 0.939 when number of clusters is 15. Lastly, we can claim that implementation of this algorithm is easier than other methods.

6 Conclusion and Future Scope

In this paper, we have presented a divisive hierarchical clustering algorithm based on minimum edge-cut principle to detect disjoint communities in a network. We have used average clustering coefficient as a quality function to measure the quality of clusters. The experiment shows that for every network which we have used in our experiment, like, Zachary's Karate Club Member Network, Dolphins Social Network, American Football Network which have inherent communities, our algorithm can detect communities and give meaningful clusters that are similar to the clusters generated by other sophisticated graph clustering methods. In our algorithm our objective is detect communities from the network by cutting minimum number of edges and from our experimental result we can say that our criteria has fulfilled, we have got dense region of graphs as clusters divided by sparse region.

For the extension of this paper we need to consider more factors in objective function to get better clusters than SOL in each and every iteration of any network. Apart from this we have to measure our clusters quality by the notion of modularity as quality function and compare the result with results of some benchmark methods which are based on modularity. Beside this, in future we want to include the concept of overlapping clusters in this algorithm. An interesting extension of this work would be to test our algorithm on large datasets that can be represented as weighted graphs.

Acknowledgement

This work was supported by the Sungshin Women's University Research Grant of 2013.

References

[1] http://en.wikipedia.org/wiki/Social_network_analysis
[2] George Karypis, Eui-Hong(Sam) Han and Vipin Kumar, "CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling," *Technical Report #99-007*, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455, USA.
[3] http://en.wikipedia.org/wiki/Social_network
[4] R. S. Salaria, *Data Structures & Algorithms Using C*, Khanna Book Publishing Co. (P) LTD., New Delhi, (2004).
[5] Pang-Ning Tan, Vipin Kumar and Michael Steinbach, *Introduction To Data Mining*, Prearson Education, Inc. , New Delhi, (2006).
[6] Thomas H. Cormen, Charles E. leiserson, Ronald L. Rivest and Clifford Stein, *Introduction To Algorithms*, Prentice-Hall India (PHI), New Delhi, (2001).
[7] Jiaweli Han and Micheline Kamber *Data Mining: Concepts and Techniques*, Second Edition, Morgan Kaufmanu Publishers, March (2006)
[8] http://en.wikipedia.org/wiki/Clustering_coefficient
[9] http://sonivis.org/wiki/index.php/Local_Clustering_Coefficient
[10] Erica Kolatch "Clustering Algorithms for Spatial Databases: A Survey," *Technical Report CMSC 725*, Department of Computer Science, University of Maryland, College Park, March (2001).
[11] Vinayaka Pandit, Natwar Modani,Sougata Mukherjea, Amit Anil Nanavati,Sambuddha Roy, Amit Agarwal "Extracting dense communitiesfrom telecom call graphs", COMSWARE, 82-89 (2008).
[12] Jure Leskovec, Ajit Singh, Jon M. Kleinberg "Patterns of Influence in a Recommendation Network", PAKDD, 380-389 (2006).
[13] M. E. J. Newman "Modularity and community structure in networks". Proc. National Academy of Sciences, USA, **103**, 8577-8582 (2006).
[14] M. Girvan and M. E. J. Newman "Community structure in social and biological networks", Proc. Natl. Acad. Sci., USA, **99**, 7821-7826 (2002).
[15] M. E. J. Newman "Fast algorithm for detecting community structure in networks", Phys. Rev. E, **69**, 066133 (2004).
[16] Mark Granovetter "The Strength of Weak Ties", American Journal of Sociology, **78**, 1360-1380 1973.
[17] Peter V. Marsden and Karen E. Campbell "Measuring Tie Strength", Social Forces, **63**, 482-501 (1984).
[18] Partha Basuchowdhuri and Jianhua Chen "Detecting Communities Using Social Ties", 2010 IEEE Internal Conference on Granular Computing, Digital Enterprise Research Institute, Ireland.

Debnath Bhattacharyya M.Tech (Computer Science and Engineering) from West Bengal University of Technology), Ph.D. (Tech., Computer Science and Engineering) from University of Calcutta, currently, associated with Computer Science and Engineering Department, Faculty of Engineering and Technology, NSHM Knowledge Campus Durgapur, as a Professor and Head. Dr. Bhattacharyya has 17 years of experience in Teaching and Research. He published 5 Text Books for B.Tech, and MCA, so far. He also published 135 Research Papers in International Journals and Conferences. His Research Interests include Biometric Recognition, Pattern Recognition and Image Processing. He is also associated with West Bengal University of Technology, University of Calcutta and many leading National and International Universities as the Ph.D. Supervisor. He is a Member of IEEE, IACSIT and CSI. He conducted many International Conferences as a General Chairs and delivered Keynote Speeches in many International Conferences.

Soumita Seth M.Tech (Computer Science and Engineering) and B.Tech (Information Technology) from West Bengal University of Technology. She worked in The Heritage Academy, Kolkata, affilated by West Bengal University of Technology, as an Assistant Professor during one semester(2011-12), and after that she was in Vinod Gupta School of Management, IIT Kharagpur, as an Academic Associate up to June 2012. Currently she is associated with Center for Softcomputing Research, Indian Statistical Institute, Kolkata, as a Junior Research Fellow leading PhD. Her main research interests are: Dynamic Network Analysis, Multiagent System, Pattern Recognition, Granular Computing, Datamining, Softcomputing.

Tai-hoon Kim received his M.S. degrees and Ph.D. in Electrics, Electronics & Computer Engineering from the Sungkyunkwan University, Korea. And he got his 2nd Ph.D. in School of Information and Computing from University of Tasmania, Australia. After working with Technical Institute of Shindoricoh 2 years as a researcher and working at the Korea Information Security Agency as a senior researcher 2 years and 6 months, he worked at the DSC (Defense Security Command) about 2 years. After working with Hannam University four and a half year as an associate professor, now he is currently working at Sungshin W. University. He wrote 17 books about the software development, OS, and computer hacking & security. And he published about 200 papers by 2012. He is a member of IEEE, ACM, KIIT and SERSC. He was a General Chair or Program Committee chair from many international conferences.