Unit III Extraction And Mining In Social Networking Data.

Extracting evolution of Web community from a series of Web Archive, Detecting communities in social Networks, Definition of Community, Evaluating communities, Methods for community detection and Mining Applications of community mining algorithm, Tools for detecting Communities in social Network infrastructures and Communities, Big data and Privacy.

3.1 Extracting Evolution of Web community from a series of Web Archive.

⇒ The propose a method for observing the evolution of Web communities. [A web community is a collection of Web pages created by individuals or associations with a common interest, on a topic such as fan pages of a baseball team and official pages of computer vendors.]

⇒ We can identify a web community on a topic by extracting densely connected structure in the Web graph in which nodes are web pages and edges are hyperlinks.

[Web archive It is the process of collecting portions of the world wide Web to ensure the information is preserved in an archive for future researchers]

Fig Typical graph of hubs and authorities.

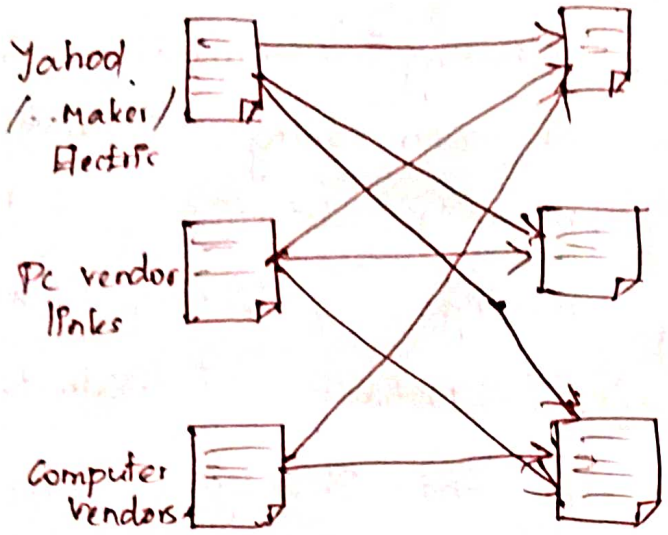


fig 1

Web Community Chart:

Web community chart is a graph that consists of Web communities as nodes and coefghted edges between related Communities.

The coefght of each edge represents the relavance of communities at both ends

Fig An example of derivation Relatfonship.

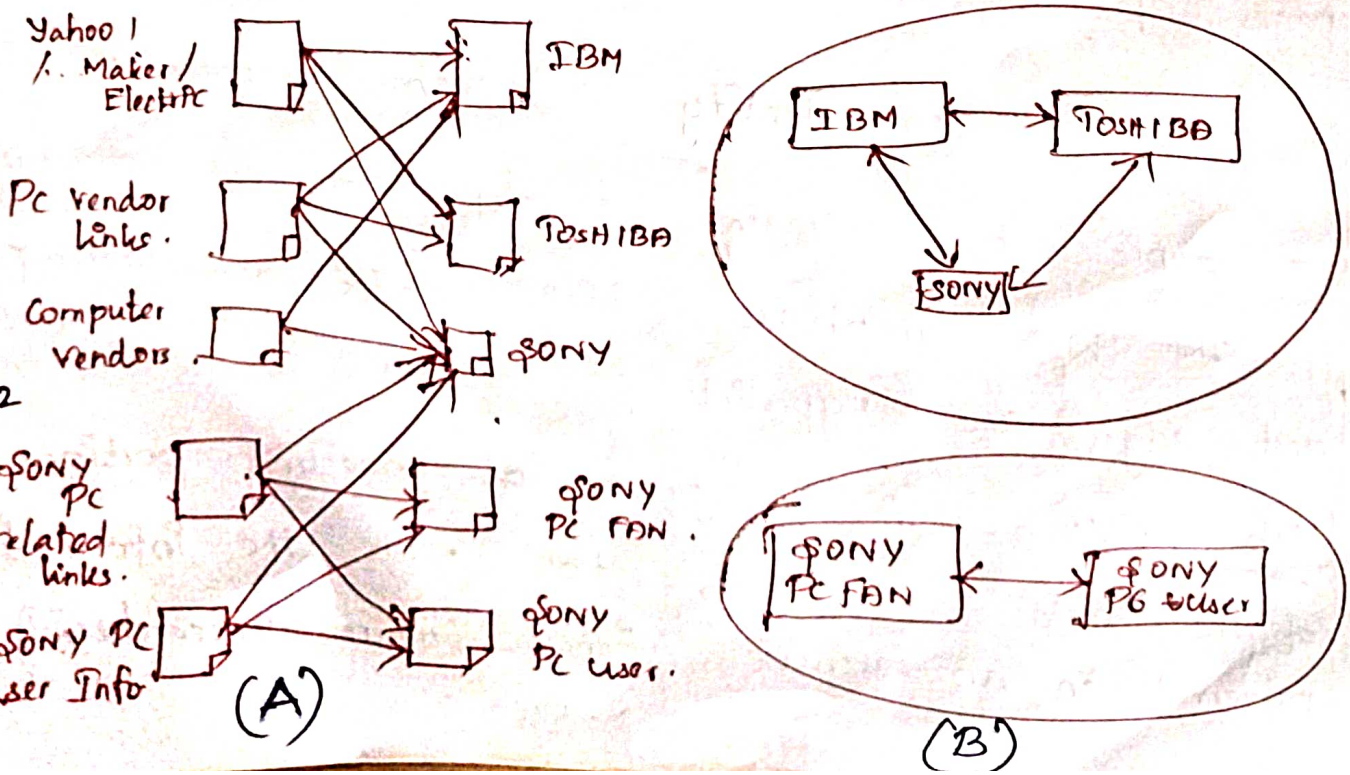


fig 2

⇒ To identify web communities and to deduce their relationship, we put focus on relationship between a seed page and related pages derived by companion.

⇒ In the above diagram, we can see that symmetric derivation is a strong relationship and asymmetric derivation is a weak relationship.

⇒ Under these observations, we define that a community is a set of pages densely connected by symmetric derivation relationship, and two communities are related if there is an asymmetric derivation relationship between members of them.

⇒ In fig 2 (A) shows each seed is pointed by hub pages and the directed graph.

⇒ (B) shows how each seed describes each other as related pages.

The extraction of Web community utilize Web community chart.

- A graph of communities, in which related communities are connected by weighted edges.

The main advantage of the Web community chart is existence of relevance between communities.

Notations :- Evolution of Communities

$t_1, t_2 \ldots t_m$: Time when each archieve crawled. Currently, a month is used as the unit time.

community can be evolved by the metrices. such as growth rate, & novelty

$W(t_k)$: The Web archieve at time t_k.

$C(t_k)$: The Web community chart at time t_k.

$c_{(t_k)}, d_{(t_k)}, e_{(t_k)} \ldots$ Communities in C_{t_k}

$W(t_k)$: The Web archive at time t_k

$C(t_k)$: The Web community chart at time t_k.

$c(t_k), d(t_k), e(t_k) \ldots$ communities in $C(t_k)$

$W(t_k)$: The Web archive at time t_k.

$C(t_k)$: The Web community chart at time t_k.

$c(t_k), d(t_k), e(t_k), \ldots$ Communities in $C(t_k)$

Types of Changes :-

⇒ Evolution of communities from a series of Web archieves. Building Web Community chart

→ $C(t_1), C(t_2), \ldots C(t_n)$ from all Web archives.

⇒ Evolution of Communities, backward and forward.

Backward Examination

⇒ By comparing $C(t_{k-1})$ and $C(t_k)$

Forward Examination.

⇒ By comparing $C(t_k)$ and $C(t_{k+1})$

Emerge

A Community $c(t_k)$ emerges in $C(t_k)$, when $c(t_k)$ shares no URLs with any Community in $C(t_{k-1})$

Dissolve :-

A Community $c(t_{k-1})$ in $C(t_{k-1})$ has dissolved, when $c(t_{k-1})$ shares no URLs with any Community in $C(t_k)$

Grow and Shrink :-

The community grows when new URLs are appeared in $c(t_k)$, and

shrinks when URLs disappeared from $c(t_{k-1})$

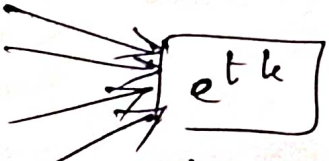
Split

A Community $c(t_{k-1})$ may split into smaller Communities.

In this case $c(t_{k-1})$ shares URLs with multiple Communities in $C(t_k)$

Merge:-

When multiple communities $(c(t_k-1), d(t_k-1) \ldots)$ share URLs with a single community $e(t_k)$ These communities are merged into $e(t_k)$



Evolution Metrics:-

Evolution metrics measure how particular community $c(t_k)$ has evolved.

The metrics are defined by differences between $c(t_k)$ and its Corresponding Community $c(t_k-1)$

Attributes Used.

$N(c(t_k))$ Number of URLs in the $c(t_k)$

$N_{sh}(c(t_k-1))$: Number of URLs shared by $c(t_k-1)$ and $c(t_k)$

$N_{dis}(c(t_k-1))$: Number of disappeared URLs from $C(t_k-1)$ that exists in $c(t_k-1)$ but do not exist in any community in $C(t_k)$

$N_{sp}(c(t_k-1), c(t_k))$: Number of URLs split from $c(t_k-1)$ to communities at t_k other than $e(t_k)$

$N_{ap}(c(t_k))$ Number of newly appeared URLs in $c(t_k)$ that

exist in $c(t_k)$ but do not exist in any community in $c(t_{k-1})$

Nmg $(c(t_{k-1}), c(t_k))$:

Number of URLs merged into $c(t_k)$ from communities at t_{k-1} other than $c(t_{k-1})$

Evolution metrices:-

Evolution metrices measure how a particular community $c(t_k)$ has evolved.

Growth Rate:-

⇒ The growth rate, $R_{grow}(c(t_{k-1}), c(t_k))$ represents the increase of URLs per unit time.

⇒ It allows us to find most growing or shrinking communities.

$$R_{grow}(t_{k-1}, c(t_k)) = \frac{N(c(t_k)) - N(c(t_{k-1}))}{t_k - t_{k-1}}$$

Disappearance Rate:-

The number of disappeared URLs from $c(t_{k-1})$ per unit time.

Higher disappear rate means that the community has lost URLs mainly by disappearance

$$R_{disappear}(c(t_{k-1}), c(t_k)) = \frac{N_{dis}(c(t_{k-1}))}{t_k - t_{k-1}}$$

Merge Rate

⇒ The number of absorbed URLs from other communities by merging per unit time.

⇒ Higher merge rate means that the community has obtained URLs mainly by merging.

$$R_{merge}(C(t_{k-1}), C(t_k)) = \frac{N_{mg}(C(t_{k-1}))}{t_k - t_{k-1}}$$

Evolution Metrices:-

⇒ By combining these metrics, some complex evolution patterns can be represented.

⇒ A community has stably growth when its growth rate is positive, and its disappearance and split rate are low.

Web Archives and Graphs:-

⇒ Web archiving is the process of collecting portions of the web to ensure the information is preserved in an archive.

⇒ Web Crawlers:- are used for automated capture due to the massive size and amount of information on the web.

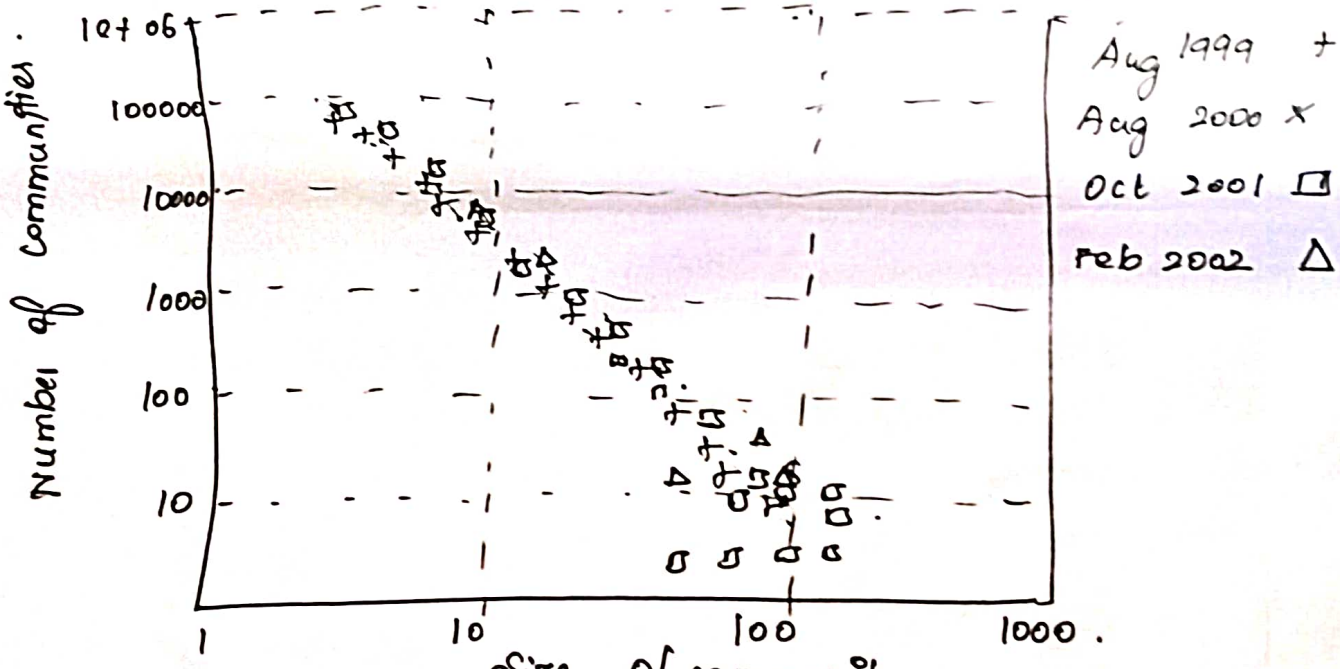
⇒ From each archive, a web graph is built with URLs and links by extracting anchors from all pages in the archive.

Evolution of Web community chart:

⇒ The size of distribution of communities also follow the power Law and its exponent did not change so much over time.

⇒ Although the size distribution of communities is stable, the structure of communities changes dynamically.

⇒ The structure of chart changes mainly by split and merge, in which more than half of communities are involved.



size distribution of communities.

Split and Merged communities:

⇒ Both distributions roughly follow the power Law, and show that split or merge rate is small in most cases.

⇒ Their shapes and scales are also similar.

⇒ This symmetry is part of the reason why the size distribution of communities does not change so much.

The graph included not only URLs inside the archive, but also URLs outside pointed to by inside URLs.

By comparing these graphs, the web was extremely dynamic.

3.2 Detecting Communities In Social Networks.

Relations of real-world entities are often represented as Networks, such as social Networks connected with friendship.

⇒ In many Cases, Real social Network contain,

* denser parts &
* sparser parts.

Denser Parts:-

⇒ Correspond to group of people that are closely connected with each other. Such denser subnetworks are called "communities" in this unit.

⇒ Detecting Communities from given social Network.. are ~~partially~~ practically important for the following reasons.

1) Communities can be used for information recommendation because members of the Communities often have similar tastes and Preferences.

Student communities books

⇒ Membership of detected Communities will be the basis of collaborative filtering

2) Communities will help us to understand the structure of the given social Networks.

Communities contain the components of the given social Networks.

They will classify the functions and Properties of the networks.

8) Communities will play the important role when we look for the large scale Network.

Relations of the communities clarify the Processes of Information sharing and Information diffusions that may use of networks in the future.

Detecting communities is not an easy task. Because Detecting Communities are very expensive proces

§3 Definition of Community :-

Community means a subnetwork whose edges connecting inside of it are denser than the edges connecting outside of it.

Three categories :

(i) Local definition.
(ii) Global definition.
(iii) Definition based on vertex similarity.

(i) Local definition :-

The attention is focused on the vertices of the subnetwork under investigation and on its immediate neighborhood.

Local definitions of community can be further divided into.

1) Self Reforming ones.
2) Comparative Ones.

The former considers the subnetwork alone, and the latter compares mutual connections of the vertices of the subnetwork with the connections with external neighbors.

1) Self referring definitions (14) are clique

(a maximal subnetworks where each vertex is adjacent to all the others)

n-clique. (a maximal subnetwork such that the distance of each pair of vertices is not larger than n)

k-plex.

(a maximal subnetwork such that each vertex is adjacent to all the others except at most k of them)

Global definitions:-

Global definition of community characterize a subnetwork with respect to the network as a whole.

2) Comparative definitions:-are is set

⇒ (a subnetwork where each vertex has more neighbors inside than outside of the subnetwork)

Weak community.

⇒ The total degrees of the vertex inside the community exceeds the number of edges lying between the community and the rest of the network.

(ii) Global Definitions: -

⇒ Global definitions of community characterize a subnetwork with respect to the network as a whole.

⇒ These definitions usually starts from a null model, in another words, a network which matches the original network in some of its topological features.

⇒ The most popular null model consists of randomized version of the original network, where edges are reviewed at random under the constraint that each verter keeps its degree. This null model is the basic concept behind the definition of modularity.

(iii) Definition Based on Verter Similarity: -

⇒ Communities are groups of vertices which are similar to each other.

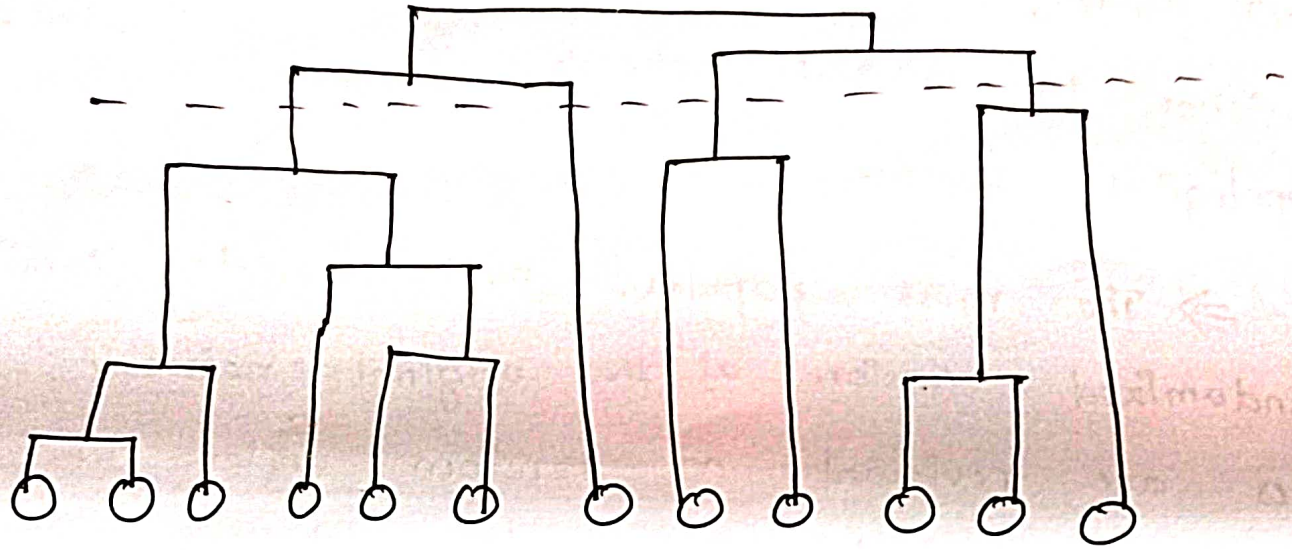
⇒ Some quantitative criterion is employed to evaluate the similarity between each pair of vertices

Hierarchical Clustering:

It is a way to find several layers of communities that are composed of vertices similar to each other.

⇒ Repetitive merges of similar vertices based on some quantitative similarity measures will generate a structure shown in fig.

Dendogram



⇒ This structure is called dendrogram, and highly similar vertices are connected in the lower part of the dendogram.

⇒ Subtrees obtained by cutting dendogram with horizontal line correspond to communities.

Communities of different granularity will be obtained by changing the position of the horizontal line.

3.4 Evaluating Communities :-

⇒ In general, there are many ways of partitioning given network into communities.

→ It is necessary to establish which partition exhibits a real community structure. Therefore we need a quality function for evaluating how good a partition is.

⇒ The most popular quality function is the Modularity of Newmann and Girivan. It can be written in several ways.

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j)$$

Where the sum runs over all pairs of vertices.

A is the adjacenty Matrix.

k_i is the degree of vertex i and

m is the total number of edges of the ~~A/to~~ network.

⇒ The Element of A_{ij} of the adjacent matrix is 1 if vertices i and j are connected.

otherwise it is 0.

⇒ The 1-function yields 1 if vertices i and j are in the same community.

0 otherwise

Modularity can be recorition as follows,

$$Q = \sum_{s=1}^{nm} \left[\frac{ls}{m} - \left[\frac{ds}{2m} \right]^2 \right]$$

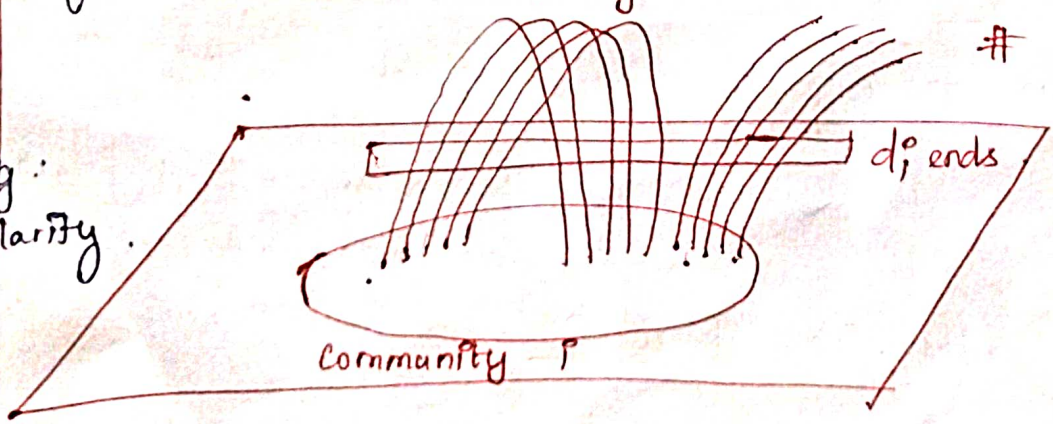
Where nm is the number of communities.

ls is the total number of edges joining vertices of community s,

ds is the sum of the degree of the vertices of s.

⇒ In the above formula, the first term of each command is the fraction of edges of the network inside the community.

⇒ The second term represents the expected fraction of edges that would be there if the network were a random network with the same degree for each vertex.



# of edges : M
# of ends : 2M

Fig: Modularity.

$e_{ii} = \frac{l_i}{M}$: Fraction of edges in Community i

$a_i^2 = \left[\frac{di}{2M}\right]^2$: Fraction of edges when connected randomly.

⇒ The comparison between real and expected edges is expressed by the corresponding summand of the above formula Above Figure Illustrates the Meaning of Modularity.

⇒ The latter formula shows the definition of community : A sub network is a community if the number of edges inside it is larger than the expected number in modularity's null model.

⇒ If this is the case, vertices of the subnetwork are more tightly connected than expected.

* Large positive values of Q are expected to indicate good partitions.

* The Modularity of the whole network, taken as a single community, is zero.

* Modularity is always smaller than one and it can be negative as well.

* Modularity has been employed as quality function in many algorithms.

2.5 Methods for Community Detection and Mining.

⇒ These are naive methods for dividing given networks into subnetworks. such as 1) graph partitioning, 2) hierarchical clustering and 3) k-means clustering.

⇒ Conversely, These methods need to provide the number of clusters

⇒ It is desirable to devise methods that have abilities of extracting a complete information about the community structure of networks.

The methods for detecting communities are classified into following categories.

* Divisive Algorithms.
* Spectral Algorithms.
* Modularity optimization.
* other Algorithms.

Applications of Community Mining Algorithms.

Divisive Algorithm.

⇒ A simple way to identify communities in a network is to detect the edges that connect vertices of different communities and remove them, so that the communities get disconnected from each other.

The most popular is that proposed by Girvan and Newman. In this algorithm, the edges are selected by according to the values of measures of edge centrality.

The steps of the algorithm are as follows.

1) Computation of the centrality of all edges.
2) Removal of edge with largest centrality.
3) Recalculation of centralities on the running network.
4) Iteration of the cycle from step 2.

⇒ Girvan and Newman focuses on the concept of edge betweenness.

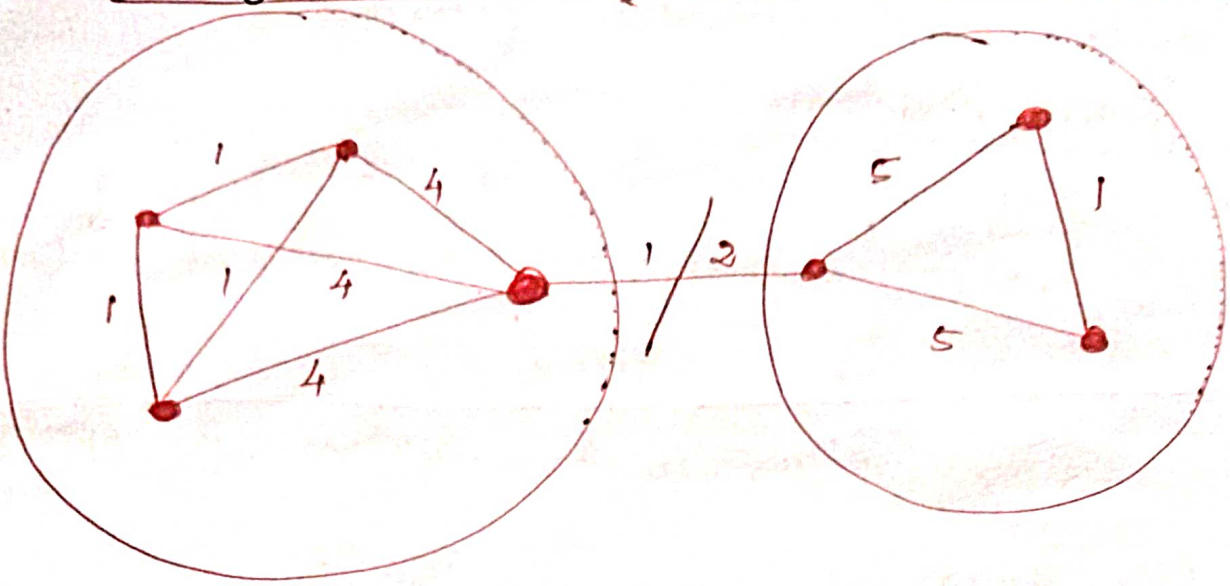
⇒ Edge Betweenness is the number of shortest paths between all vertex pairs that run along the edge.

⇒ It is intuitive that intercommunity edges have a large value of the edge betweenness. because many shortest paths connecting vertices of different communities will pass through them.

In the below figure.

⇒ Two communities will be obtained by removing the central edge, whose edge betweenness is the largest.

Detecting communities Based on Edge Betweenness.



2. Modularity Optimization :-

Modularity is the quality function for evaluating partition.

The partition corresponding to its maximum value on a given network should be the best one. This is the main idea for modularity optimization.

By the way of Modularity optimization,

↳ NP hard Problem is used

currently, several algorithms are used to fastly find good approximation of the modularity maximization in a reasonable time.

Famous algorithm for Modularity optimization, is CNM algorithm, proposed by clauset.

Another examples of the algorithm are greedy algorithm and simulated annealing.

Spectral Algorithms :-

⇒ spectral algorithms are to cut given networks into pieces so that the number of edges to be cut will be minimized.

⇒ One of the basic algorithm is spectral graph bipartitioning.

As Procedure.

* Laplacian matrix L of given network is used.

* Laplacian matrix L of a network is an $n \times n$ symmetric matrix, with one row and column for each vertex.

* Laplacian matrix is defined as $L = D - A$,

Where A is the adjacency matrix

D is the diagonal degree matrix with

$$D_{ii} = \sum_k A_{ik}.$$

All eigenvalues of L are real and non-negative, and L has full set of n real and orthogonal eigen vectors.

Inorder to minimize the above cut, vertices are partitioned based on the sign of the eigen vector

In general, community detection based on repetitive bipartitioning is relatively fast.

Other Algorithms :-

There are many algorithms for detecting communities, such as the methods focusing on random walk.

and the ones searching for overlapping cliques.

3.b Tools for detecting communities social network infrastructures and communities.

Several tools have developed for detecting communities.

These are roughly classified into following two categories.

* 1) Detecting communities from large scale networks,
* 2) Interactively analysing communities from small network.

2) Tools for ~~Large Scale Networks~~ Interacting Analysis of small N/w :-

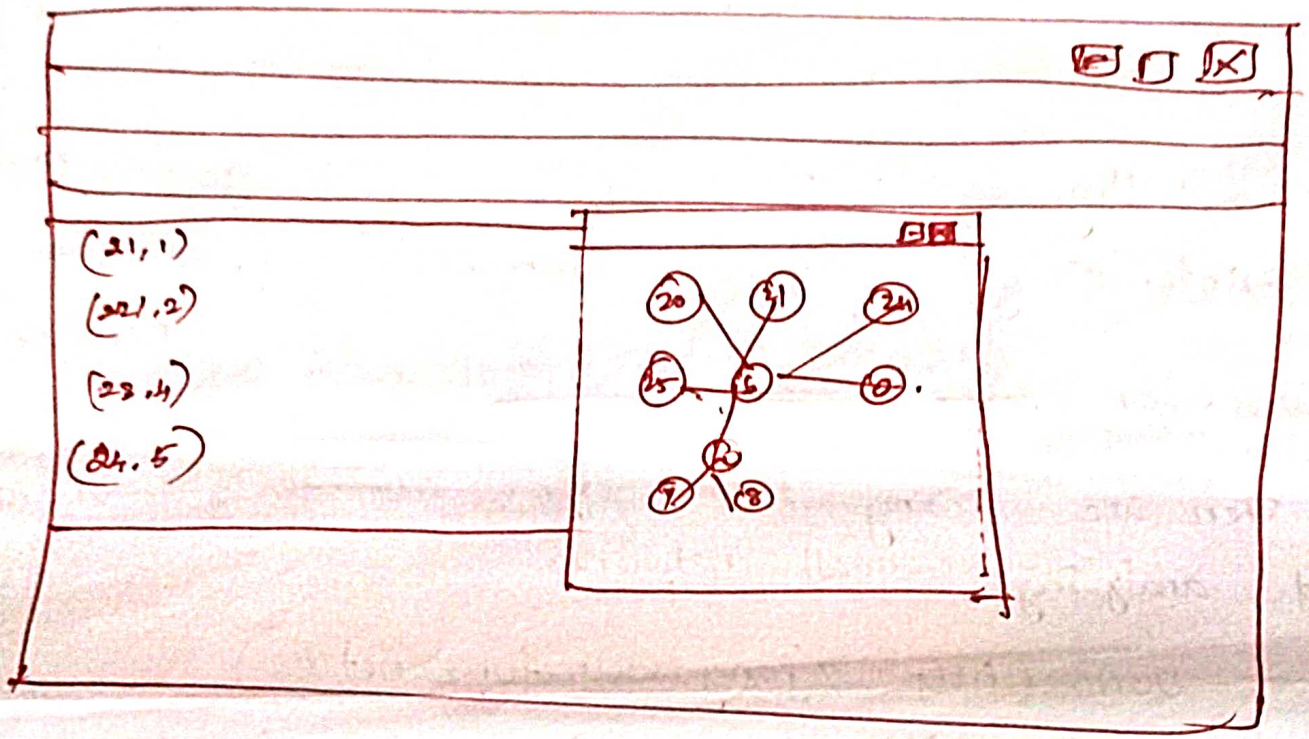
There are many tools for interactively visualizing and analysing small networks

JUNG (http://jung.sourceforge.net/)
Netminer (http://www.netminer.com/NetMinor/overview01.jsp)
Pajek (http://vlado.fmf.uniij.si/pub/networks/pajek/)
Igraph (http://igraph.sourceforge.net/)
SONIVIS (http://www.sonivis.org/)
Commetrix (http://www.commetrix.de/)
NetworkWorkbench (http://nwb.slis.Indiana.edu/)
Visone (http://visone.info/)
CFinder (http://www.cfinder.org/)

Igraph is a free software for network analysis,

1) Tools for Large scale network: Clauset et al propose CNM algorithm of community detection based on modularity optimization. This algorithm is applicable for Large scale N/w of million of vertices

and it can be used as the libraries of C programming language, and as the combination with R, a programming language for statistical analysis.



Screenshot of Rigraph

Details of R are available at http://cran.r-project.org/

Details of igraph. are available at http://igraph.sourceforge.net/

Karate club network:

=> shows the relations of certain Karate club members.

=> Karate club network represented in GML format

Graph Network Modelling language (GML) is one of the format for representing networks.

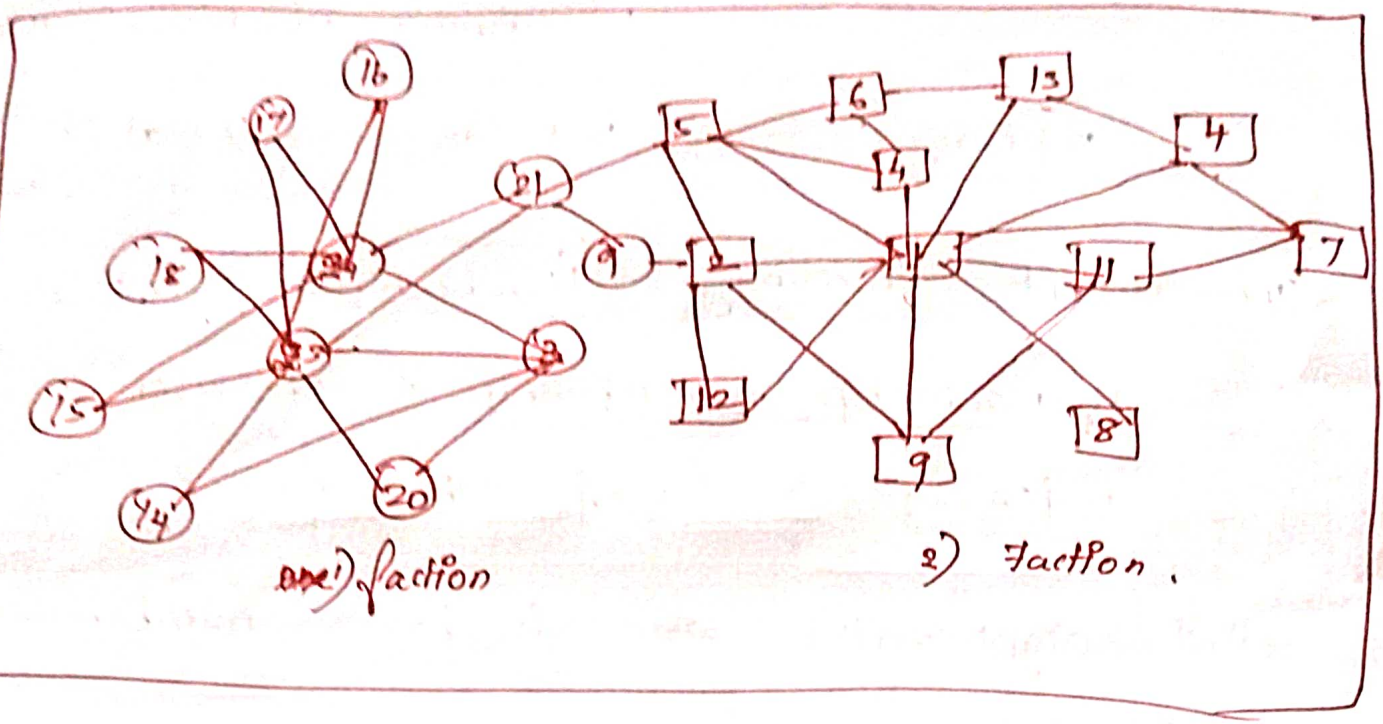


Fig. Karate club Network.

The command read, Graph Loads network data.

⇒ command tkplot perform visualization.

Initial positions are assigned randomly and users can select major algorithms for visualization such as

⇒ 1) Kamada - kawai and 2) Fruchterman Reingold.

Position of vertices can be adjusted manually.

The command fast greedy is for maximizing modularity greedily,

Results stored in variable gr.

⇒ The variable gr is composed of

gr$modularity (a list of modularity values in the process of maximization)

gr$merge (vertexID that are merged at each step.

⇒ The command community.to.membership generates

m$membership - membership of each vertex.

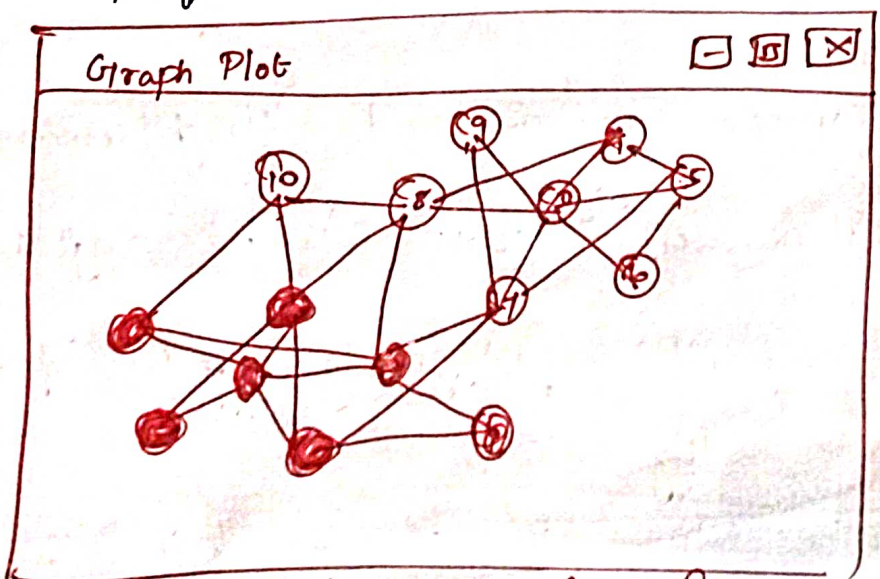
m$csize - size of each community.

⇒ First eight vertices are of lower numbers belong to community 0.

size of two communities are both 17.

After storing membership m$membership to V(g)$color.

⇒ second tkplot command specifies a visualization algorithm and colors of vertices in visualization order to perform visualization.



Community detection by fastgreedy algorithm

↳ Other methods for detecting are also available in igraph.

↳ community detection based on edge Betweenness.

The command edge.betweenness.community detect communities by repetively removing edge of high edge betweenness.

⇒ Contents of the obtained variables is different from previous fastgreedy community.

ed$removed.edges (list of removed edges)

ed$edge.betweenness (list of edge betweenness)

ed$merges. (list of vertices that are merged in a dendogram.)

ed$bridges (list of bridging edges.

Edge betweenness divides given network in a top-down manner, while ed$merges shows list of vertices in a reverse order. (Bottom-up manner.

_ shows the explanation of each comman

Fig Community detection by fastgreedy in R cigraph.

```
>library("igraph")                    # Load igraph library
    >setwd("R")                       # Change directory.
>dir()                                # show files.
[1] "Karate.gml"
>g<- read.graph("karate.gml",
                format="gml")         # Load network data.
>tkplot(g)                            # visualize network.
[1] 1
>gr<- fastgreedy.community(g)
>gr$modularity
[1] 0.04980276  -0.03763971  -0.1775148  0.04906312
[31] 0.37598619  0.38067061  0.37179487  0.00000000
>gr$merge
     [,1] [,2]                        # list of merged vertices
[1,]   5   16
- - - -
[33,] 65   63
>m<- community.to.membership(g, gr$merge,
                steps=nrow(gr$merge)-1)
                                      # store membership and size
                                      # show the contents.
>m
$membership
[1] 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 1 1 0 0 1 1 1 1 1 1 1 1
```

```
$csize
[1] 17 17
>V(g)$color <- mem$membership
>V(g)$color                    #store membership
[1] 0000000001000001100101010111111111
tkplot(g, layout = layout.kamada.kawai, vertex.
                    color = V(g)$color)
                    #visualize network.
[1] 2
>
```

Fig Community Detection Based on Edge Betweenness In Rigraph.

```
>ed <- edge.betweenness.community(g)   #detect
                                        communities.
>ed                                    #show result
$removed.edges.                        # list of removed
                                        edges.
[1] 46 1 15 63 67 50 44 2 5 13 26 72 77.
 - - -
[13] 57 70 48 58 74 77
$edge.betweenness.
[1] 71.392857 66.895177 77.317399 82.002906
73 1.50000 3.00000 1.00000 2.00000
$merges
     [,1] [,2]
[1,] 32 33
```

[38] 65 64

$ bridges

[1] 78 77 75 74 71 70 68 67 66 51 50 47

[25] 33 31 29 27 25 24 18 14 11

> community.to.membership(g, ed$merges, steps = nrow(ed$merges)-1)

membership of two communities

$membership

[1] 1101111100 111100 11 010 10 10 10000000000

$csize

[1] 19 15

> community.to.membership(g, ed$merges, steps = nrow(ed$merges)-2)

membership of three communities.

$membership

[1] 00100000 12 0000 1100101010 111 1111

$csize

[1] 15 18 1

> community.to.membership(g, ed$merges, steps = nrow(ed$merges)-3)

membership of four communities.

$membership

[1] 1102222103211102002101000000