AN ENHANCED COMMUNITY DETECTION METRIC FOR WEIGHTED AND DIRECTED GRAPH-BASED NETWORK

AUTHORS

Md. Hasibul Kabir - 134409

Md. Anower Jahan -134412

A Thesis Submitted to the Academic Faculty in Partial Fulfillment of the Requirements for the Degree of

BACHELOR OF SCIENCE
IN COMPUTER SCIENCE AND ENGINEERING

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

AN ENHANCED COMMUNITY DETECTION METRIC FOR WEIGHTED AND DIRECTED GRAPH-BASED NETWORK

AUTHORS

Md. Hasibul Kabir - 134409

Md. Anower Jahan -134412

Department of Computer Science and Engineering (CSE)
Islamic University of Technology (IUT)
Organization of Islamic Cooperation (OIC)
Board Bazar, Gazipur-1704
Bangladesh

Declaration of Authorship

This is to certify that the work presented in this thesis is the outcome of the analysis and experiments carried out by Hasibul Kabir Emon and Md. Anower Jahan bearing Student No. 134409 and 134412 respectively of Academic Year 2016–2017 under the supervision of Dr. Abu Raihan Mosotofa Kamal, Associate Professor of Department of Computer Science and Engineering (CSE), Islamic University of Technology (IUT), Gazipur, Bangladesh. It is also declared that neither of this thesis nor any part of this thesis has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Authors

_____ _____

Hasibul Kabir Emon Md. Anower Jahan

Approved by

_____

Dr. Abu Raihan Mostofa Kamal
Associate Professor
Department of Computer Science and Engineering
Islamic University of Technology
Board Bazar, Gazipur-1704, Bangladesh.

Acknowledgement

First of all, We would like express our heartiest gratitude to the Almighty Allah for providing me the strength to complete this thesis work. After the Almighty, it is my great pleasure to express gratitude to the people who made this thesis possible.

We would like to express our grateful appreciation to **Dr. Abu Raihan Mostofa Kamal**, Associate Professor, Department of CSE, IUT for being our advisor and mentor. His motivation, suggestions and insights for this thesis have been invaluable. Without his support and proper guidance this research would not have been possible. His valuable opinion, time and input provided throughout the thesis work, from first phase of thesis topics introduction, subject selection, proposing algorithms, modifications till the project implementation and finalization which helped us to do our thesis work in a proper way. We are really grateful to him.

Our deep appreciation extends to all the respected jury members of our thesis committee for their insightful comments and constructive criticism of our research work. Surely, they have helped us to improve this research work greatly. Lastly, we would like to thank the faculty members of CSE Department, IUT who have helped to make our working environment a pleasant one, by providing a helpful set of eyes and ears when problems arose.

Abstract

Community detection algorithm tries to find the densely connected units in a large network. For this objective different matrices have come into light. Most of them assume all the vertices in a community belong equally to the community. But these matrices identify the communities as whole, these doesn't give any information at which content the nodes are connected in a community. Moreover these also face resolution limit for larger networks. For resolving this issue, another matrices named permanence has been applied. But this metric is not defined for weighted and directed graph. Our approach will be to implement the permanence on directed and weighted graph.

Contents

Chapter 1

Introduction

1.1 Background

Many real-world systems can be represented as complex networks such as Protein Interaction Network [Ref], Indian Railway Network [ref], and Coauthor Ship Network [ref]. In this real world networks, community structure is frequently being observed.

By the term community structure, we mean that a network can be decomposed into its constituent sub graphs where within each sub graph the vertices are densely connected and between sub graphs the vertices are sparsely connected. Each constituent sub graph is called community. Now a day, detecting this community in a network is a major issue.

1.2 Motivation

Community Detection Metrics is also a top of the topics in current days. Earlier several metrics such as Modularity (Newman M. E., Modularity and community structure in networks, 2006), Conductance (Jure Leskovec, 2009), and Cut-Ratio (Fortunato S. , 2010) has been proposed. But this metrics suffers from resolution limit and degeneracy of solutions (Barthelemy, 2007) and more ever from this metrics we cannot know to what extent a vertex belongs to its community. We can only get the community metric value over the network as a whole.

Then another metric Permanence (Tanmoy Chakraborty, 2016) has been proposed which mitigates this problem. But this metric does not provide the information of how the value of permanence will be affected if the graph is weighted and directed. But in real world some graphs are weighted in nature and their unweighted version loses some important information. For example, if users of social networking sites are represented as vertices and their interaction as edges then from weighted version we can easily derive how intimate the two users is but in case of unweighted version this information is lost. Some graphs are also directed in nature

such that map of a city where one way road exists, dependencies in a software module, hyperlink connecting webpages etc. So, in this paper, we propose a metric which shows how permanence value is affected in case of weighted and directed graph.

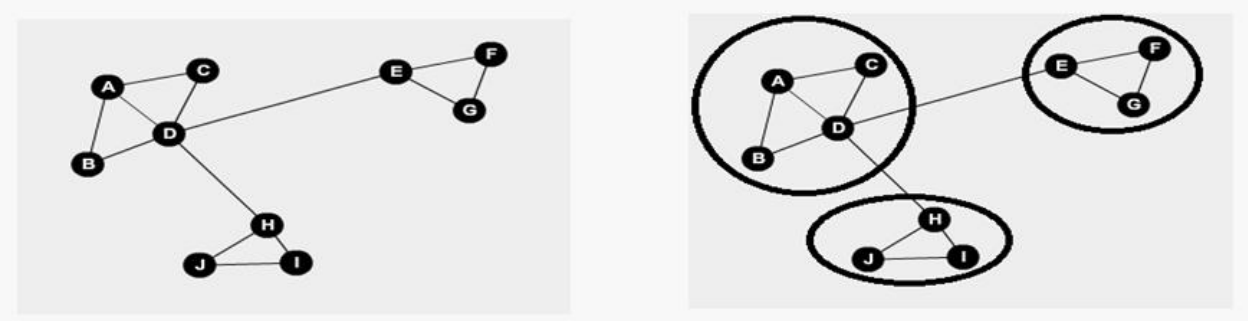


Figure 1: A graph with community substructure(Left), After Detecting Community (Right)

In this article the background information about community detection has been described and different matrices were discussed. After discussing every matric there limitations were also defined and then the permanence definition and its performance were discussed. After discussing about permanence and the community detection algorithms, we discussed about our proposal and implementation.

1.3 Problem Statement

The best metric till now called the permeance is not define for sighted and directed graph. But what can be done on detection the community when the substructure will be based on a weighted and directed graph? That is our concern to solve.

1.3 Our Approach

We tried to modify the permeance and its definition according to weighted and directed graphs. We modified the MaxPerm based on the enhanced metrics.

Chapter 2

Related Works

In the past there are two types of work that has been done in the field of Community Detection. There are two parts in the related works done in this area of community detection-

1. Community Detection algorithms and
2. Evaluation metric for obtained community.

Several community evaluations metric has been proposed. The most widely used one is Modularity (Barthelemy, 2007). It is defined as difference between the actual fraction of edges that lies with in communities and the expected fraction of the same thing.

At first it was defined for the unweighted and undirected graph. Later it was defined for the weighted (Newman M. E., Analysis of weighted networks, 2004) and directed graph (Newman E. A., 2008).But modularity suffers from resolution limit (Newman M. E., Modularity and community structure in networks, 2006) and degeneracy solutions (Benjamin H Good, 2004).Multi-view modularity (Peter J Mucha, 2010) was also proposed for multi-view network.

Many other metric for evaluation was also proposed like Conductance (Fortunato S. , 2010), Cut ratio (Jure Leskovec, 2009), weighted conductance (Zongqing Lu, 2013). In the past years several algorithms has been proposed for detecting communities. Algorithm proposed by Blonde (Vincent D Blondel, 2008) is based on local optimization of modularity (Barthelemy, 2007). There are several other approaches like label propagation (Usha Nandini Raghavan, 2007), random-walk based approach (Pasquale De Meo, 2013), spectral graph-partitioning algorithm (Newman., 2013). Most of the algorithms are highly dependent on vertex ordering (Fortunato A. L., 2012) (i.e., the order in which the vertices are processed). To mitigate this problem MaxPerm (Tanmoy Chakraborty, 2016) which is based on Permanence Maximization is proposed.

Most of the research in community detection algorithms are based on the idea that a community is a set of nodes that has more and/or better links between its members than with the remainder

of the network.Work in this area encompasses many different approaches including, modularity optimization [Blondel et al. 2008; Clauset et al.2004; Guimera and Amaral 2005; Newman 2004b, 2006], spectral graph-partitioning algorithm [Newman 2013; Richardson et al. 2009], clique percolation [Farkas et al. 2007; Palla et al. 2005], local expansion [Baumes et al. 2005; Lancichinetti et al. 2009], fuzzy clustering [Psorakis et al. 2011; Sun et al. 2011], link partitioning [Ahn et al.2010; Evans and Lambiotte 2009], random-walk-based approach [De Meo et al. 2013; Pons and Latapy 2006], information theoretic approach [Rosvall and Bergstrom 2007, 2008], diffusion-based approach [Raghavan et al. 2007], significance-based approach [Lancichinetti et al. 2010], and label propagation [Raghavan et al. 2007; Xie and Szymanski 2011, 2012].

However, most of these algorithms produce different community assignments if certain algorithmic factors, such as the order in which the vertices are processed, change. Lancichinetti and Fortunato [2012] proposed consensus clustering by re-weighting the edges based on how many times the pair of vertices were allocated to the same community, for different identification methods. Several pre-processing techniques [Bader et al. 2013; Riedy et al. 2011] have been developed to improve the quality of the solution. These methods form an initial estimate of the community allocation over a small percentage of the vertices and then refine this estimate over successive steps. Recently, Chakraborty et al. [2013] pointed out how vertex ordering influences the results of the community detection algorithms. They identified invariant groups of vertices (named as "constant communities") whose assignment to communities is not affected by vertex ordering.

Now, from this works some related works will be described in brief. At first, modularity will be explained.

2.1 Community Detection Metrics:

Modularity:

Modularity Defines as ratio between sum of weight of internal edge and sum of weight of edges. It is calculated using following formula.

$$\frac{\sum_{uv} A_{uv} \quad \delta(c_u, c_v)}{\sum_{uv} A_{uv}}$$

Here, A_{uv} denotes the value of Adjacency matrix of the graph. $\delta(c_u, c_v)$ returns 1 when node u and node v is in same community and 0 when different.

This general modularity has some limitations. If we make all the vertices in a single community then the modularity gives the highest value. But that is misleading. Because that fails the purpose of community detecting.

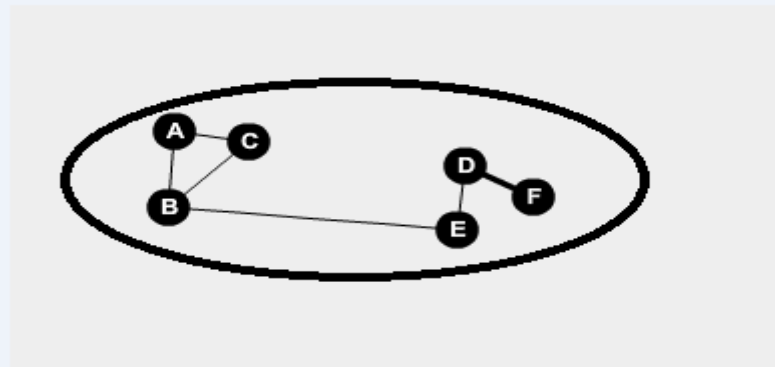


Figure 2 Gives the highest value 1 if we include all the nodes in a single community

In (Newman M. E., Analysis of weighted networks, 2004), the modularity has been modified. It resolves the problem of previous modularity which gives the highest modularity when all the vertices are considered to be in a single community.

It is defined as the difference between the actual and the expected sum of weight of edges inside a given community. It is calculated using following formula:

$$\frac{\sum_{uv} \left(A_{uv} - \frac{K_u K_v}{\sum_{uv} A_{uv}}\right) \delta(c_u, c_v)}{\sum_{uv} A_{uv}}$$

Here, A_{uv} denotes the value of Adjacency matrix of the graph. $\delta(c_u, c_v)$ returns 1 when node u and node v is in same community and 0 when different.K_v is the sum of weight of edges incident upon node v.

This modularity can be used for only undirected graph. But there is a slight modification in it which has been expanded to directed graph. It is calculated using following formula:

$$\frac{\sum_{uv} \left(A_{uv} - \frac{k_u^{in} k_v^{out}}{\sum_{uv} A_{uv}}\right) \delta(c_u, c_v)}{\sum_{uv} A_{uv}}$$

Here, A_{uv} denotes the value of Adjacency matrix of the graph. $\delta(c_u, c_v)$ returns 1 when node u and node v is in same community and 0 when different. k_u^{in} is the sum of weight of edges that has direction towards node u. k_v^{out} is the sum of weight of edges that has direction from node v to outside?

Limitations of modularity:

1. The optimal score of modularity can only be obtained over the network as a whole because the algorithm notions that all the vertices in a community belong equally to the community.

2. The information about how placement of vertices affects the community structure is lost using these current measurements.

Conductance:

Conductance [Leskovec et al. 2009] is the ratio between the number of edges inside the cluster and the number of edges leaving the cluster [Kannan et al. 2000; Shi and Malik 2000]. More formally, conductance _(*S*) of a set of nodes *S* is defined as follows:

$$\Phi(S) = \frac{C_S}{min(Vol(S), Vol(V \setminus S))},$$

where C_S denotes the size of the edge boundary, $C_S = |(u, v) : u \in S, v \notin S|$, and $Vol(S) = \sum_{u \in S} d_u$ where d_u is the degree of vertex u.

Cut Ratio:

Cut-ratio is a standard metric in graph clustering [Fortunato 2010;Leskovec et al. 2010], which is defined as the fraction of all possible edges leaving the cluster *S*. Formally, given an undirected graph *G*(*u*, *v*), the cut-ratio *θ*(*S*) of a set of nodes *S* is defined as follows:

$$\theta(S) = \frac{C_S}{n_S(n - n_S)},$$

where C_S is defined earlier, and $n_S = |S|$.

Note that the higher is the value of modularity, the better is the quality of the community structure; however, for conductance and cut-ratio, the opposite argument is applicable. Therefore, to make these two measures comparable to modularity and permanence, we measure (1-Con) and (1-Cut) for conductance and cut-ratio, respectively.

Permanence:

Permanence is also another community detection metric which resolves the problems and limitations of modularity. The main point to be noted about permanence is that it quantifies a vertex's propensity to remain in its assigned community and the extent to which it is "pulled" by the neighboring communities.

There are two concepts behind the permanence metric. They are:

1. A vertex should have more number of internal connections than the number of connections to any of the external neighboring communities.

Most optimization metrics consider the total number of external neighbors of the vertex. However, in our earlier experiment [Chakraborty et al. 2014; Chakraborty 2015; Chakraborty et al. 2016b], we empirically demonstrated that a group of vertices are likely to be placed together so long as the number of internal connections is larger than the number of connections to any one single external community. In other words, a vertex that has connections to some external communities experiences a separate "pull" from each of these external communities. In formulating permanence, we consider the maximum pull, which is proportional to the maximum number of connections to an external community.

2. Within the substructure of a community, the internal neighbors of the vertex should be highly connected among each other.

Most optimization metrics only consider the internal connections of a vertex within its own community. However, how strongly a vertex is connected also depends on whether its internal neighbors are connected with each other. To measure this connectedness of a vertex, we compute the clustering coefficient of the vertex with respect to its internal neighbors. For a vertex *v* belonging to community *c*, it is measured by the ratio between the actual number of edges among the neighbors (which also belong to *c*) of *v* and the total number of possible edges among the neighbors [Holland and Leinhardt 1971]. The higher this internal clustering coefficient, the more tightly the vertex is connected to its community

Combining these two criteria permanence of a vertex *v* is formulated as follows:

$$Perm(v) = \left[\frac{I(v)}{E_{max}(v)} \times \frac{1}{D(v)}\right] - [1 - c_{in}(v)]$$

where *I* (*v*) is the number of internal (in its own community) neighbors of *v*, *Emax* (*v*) is the maximum number of connections of *v* to any one of the external communities, *D*(*v*) is the degree of *v* and *cin*(*v*) is the clustering coefficient among the internal neighbors of *v*.

There is some unique strength of permanence in the field of community detection. These are-

- ✓ Strengthening community structure
- ✓ Determining initial selector for message spreading.
- ✓ Can detect position from center of a community as farness centrality.

Strengthening community structure

The value of permanence of a vertex signifies its propensity to remain in its own community. Therefore, vertices having low permanence in a community are loosely connected to the community. We explore whether we can strengthen the community structure by deleting vertices with low permanence. Note that when a vertex is deleted from its community, it would also affect the permanence value of the remaining vertices.

Determining initial selector for message spreading

Since vertices with higher permanence form the core of the community, we posit that initiator selection based on permanence would help in faster dissemination of the message. Note that the message spreading algorithms are based on only the local view of the vertices; therefore, global methods such as those described in Kempe et al. [2003] will not be applicable under this formulation. We observe that the permanence-based initiator selection from ground-truth communities requires minimum timestamps to spread the message compared to the degree-based selection.

Heterogeneity and Core-Periphery Organization of Community Structure

Within a community, the extent of involvement and activity may not be same for all members permanence can capture this *heterogeneity*. The permanence of a node *v* belonging to a community *c* indicates the extent to which the node belongs to the community. The next investigation reveals the manner in which the permanence value of vertices decreases from the core. A smooth decrease in value would indicate that the nodes in a community are arranged in layers with each layer of vertices roughly having similar permanence.

Limitations of permanence:

This is not defined for weighted and directed graph.

2.2 Community Detection algorithms:

There are so many community detection algorithms which have been cited before. Here two of them are described. They are:

- Generalized Louvain method
- MaxPerm Algorithm

Louvain is defined upon the modularity optimization and Maxperm is based on the permanence value.

Generalized Louvain method

This is a community detection algorithm based on modularity optimization. It has two stages.
Stage 1:

- ✓ Initially each node is assigned in new community.
- ✓ Then, each node is removed from its own community and places it in another community.
- ✓ If none of these modularity changes are positive, we keep the node in its current community.
- ✓ If some of the modularity changes are positive, we move the node into the community for which the modularity change is most positive.

Stage 2:

- ✓ Now each community is considered as a node.
- ✓ Internal connections of a community are considered as self-loop of that corresponding node.
- ✓ Sum of weight of external connections between the communities is considered as edges between the corresponding nodes.
- ✓ Then, stage 1 is repeated until there is no positive change in modularity.

There is a limitation of this algorithm. That is resolution limit. It can't work for larger community. It merges the smaller communities and so those small communities can never be found as a result of modularity optimization

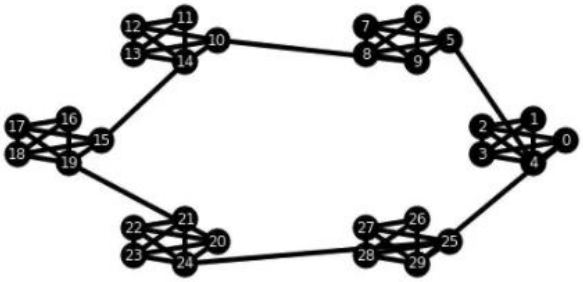


Figure 3 After first stage

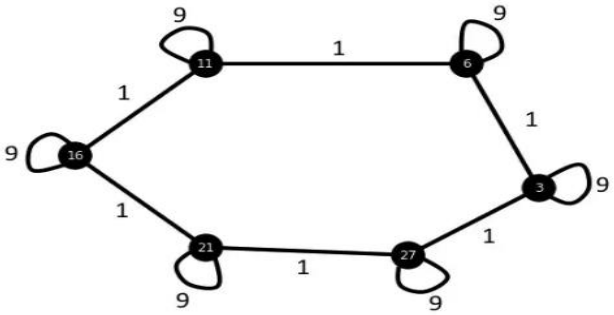
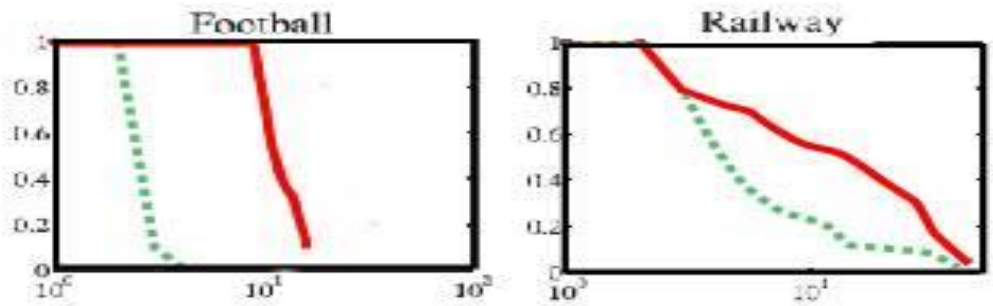


Figure 5 After second stage

Maxperm Algorithm

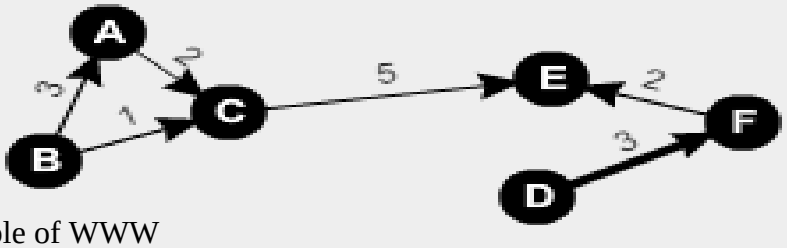
- ✓ Community detection algorithm based on permanence optimization.
- ✓ Initially nodes are assigned in seed community.
- ✓ Then, each node and its neighbor's permanence are calculated.
- ✓ Then each node is placed into other communities and again permanence of the node and its neighbors is calculated.
- ✓ If both increases then we keep the changed state otherwise previous state is preserved.

The maxperm algorithm can mitigate the problem of modularity optimization which was resolution limit. But this algorithm has some limitations also. It can't detect community in large networks.



Our Proposal

- Some graph are weighted and directed. As we have seen that permanence just works for unweighted and undirected graph so we want to modify it to apply in this field.
- For example, World Wide Web. Where one site can give reference to another website by hyperlink but that targeted website may or may not redirect to previous website. This causes the graph to be directed graph where websites are nodes and references are links between them and number of links between the website defines the weight of the edge.
- We will extend the definition of permanence such that it can be applied on weighted and directed graph.



Example of WWW

Future Work

- We have implemented MaxPerm and Louvain algorithm for our future use in the modified permenance.
- In the future we will try to implement the MaxPerm based on modified permanence.
- We will analyze the result of Modified MaxPerm algorithm along with Louvain method.
- Necessary dataset like Railway Network[4], Co-authorship Network[4] has been collected for our work.

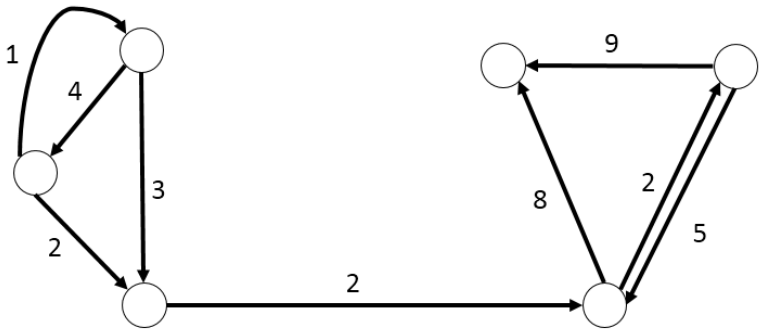
Chapter 3

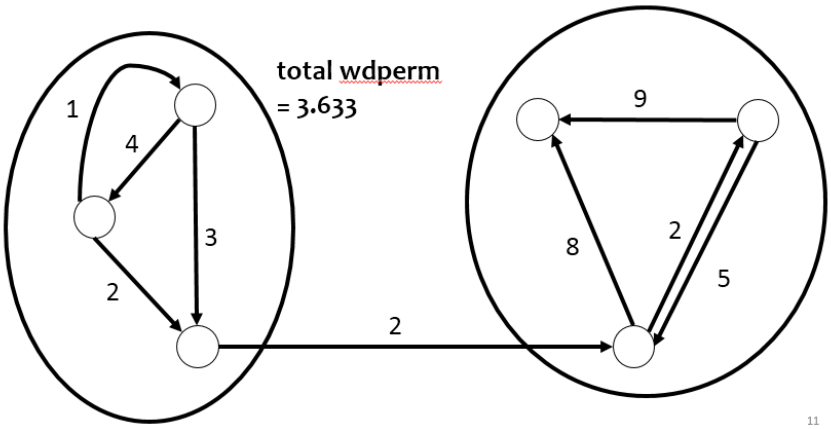
Implementation

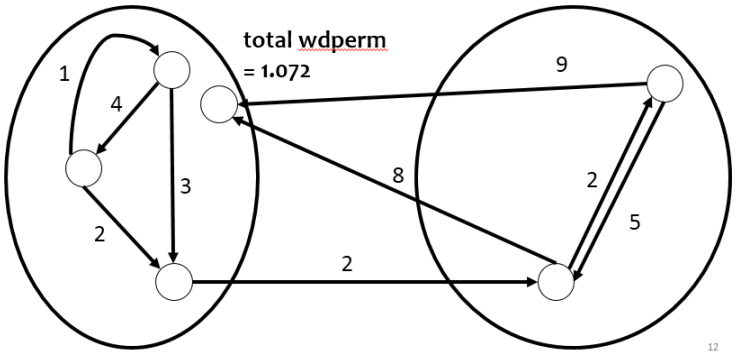
3.1 Enhanced Metric

We have considered the weight and direction of the nodes of a network. For this in the place of internal connection numbers we have taken the total weight of the in and outgoing vertices weight. In the place of maximum number of external connections, we considered the maximum weight of the edges to external communities. And we have modified the clustering coefficient also. We have defined it as the ratio of total weight of edges between neighboring vertices and the total number of possible connections can be made. And before the denominator was n(n-1)/2 but this time it is n(n-1) as in the directed graph edges can be on both sides

Enhanced permanence in working:







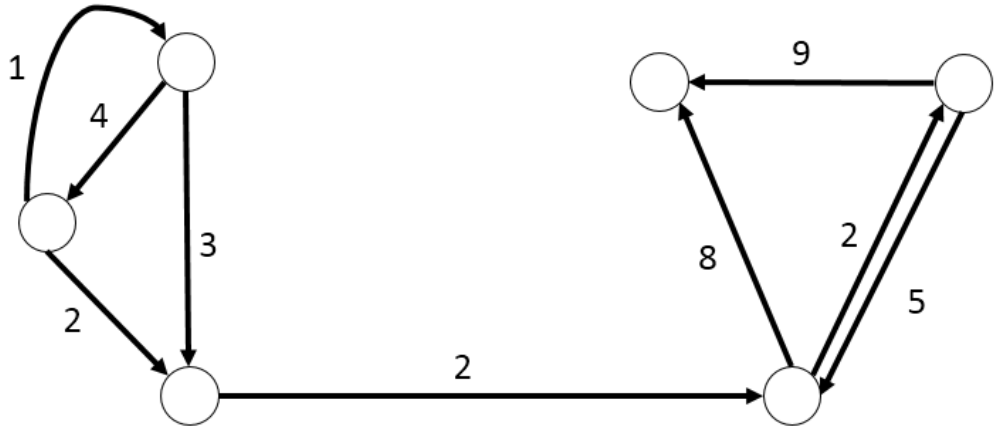
Equation for calculating weighted and directed permanence of a vertex

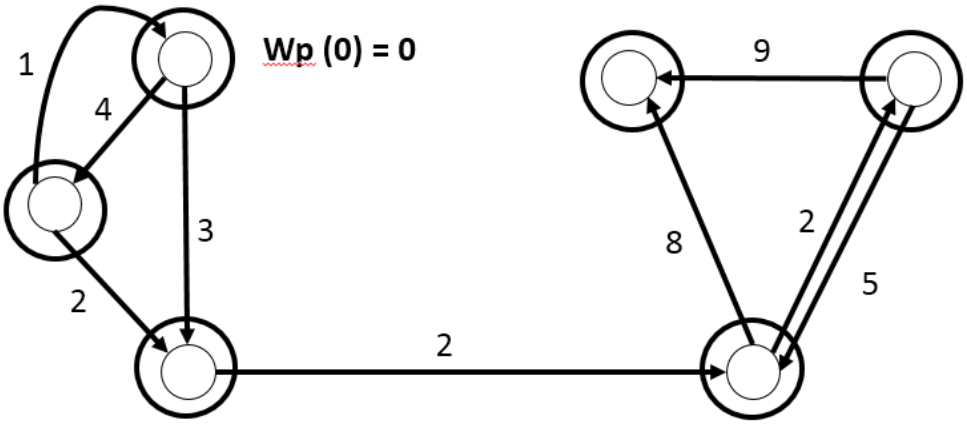
$$\boldsymbol{WDPerm(V)} = \left[\frac{\boldsymbol{IWin(V)+IWout(V)}}{\boldsymbol{Max(E_{in}(V)+E_{out}(V))}} \times \frac{\mathbf{1}}{\boldsymbol{Win(V)+Wout(V)}}\right] + [\boldsymbol{wc_{in}(V)}]$$

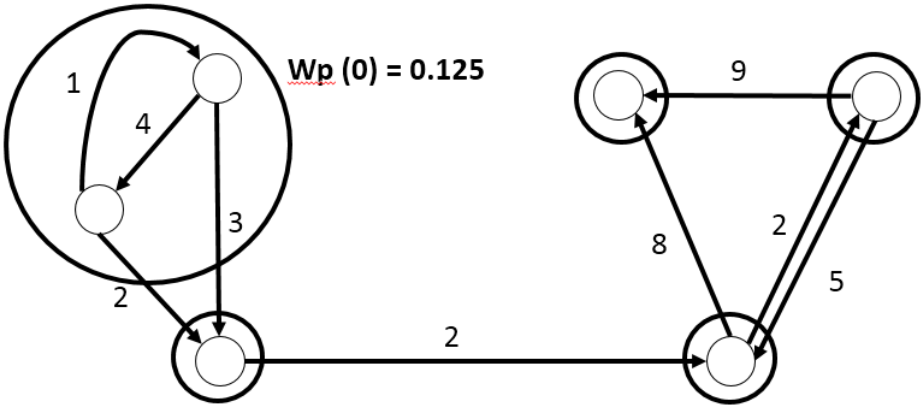
$$\boldsymbol{wc_{in}(V)} = \frac{\boldsymbol{Total\ weight\ of\ neighbouring\ nodes}}{\boldsymbol{n(n-1)}}$$

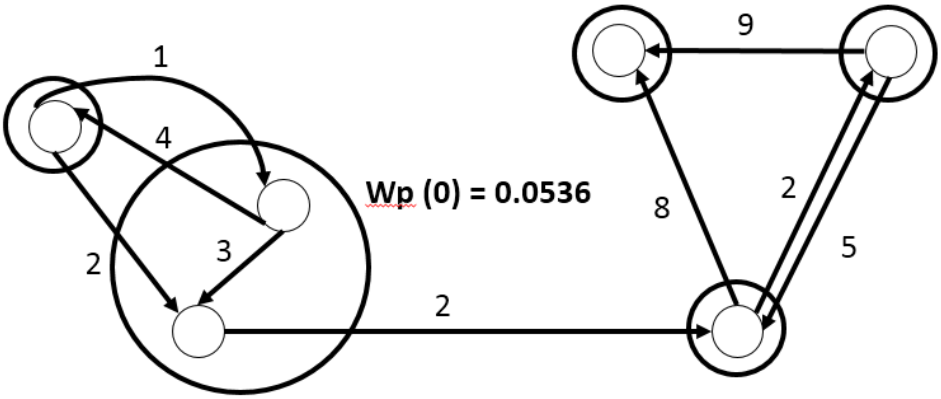
- Here, $IWin(V) + IWout(V)$ denotes the total internal weight of vertex V
- $Max(E_{in}(V) + E_{out}(V))$ denotes the maximum weight of external connections to any external communities
- $Win(V) + Wout(V)$ is the total weight of in and outgoing edges of v and
- $wc_{in}(v)$ is the weighted clustering coefficient.

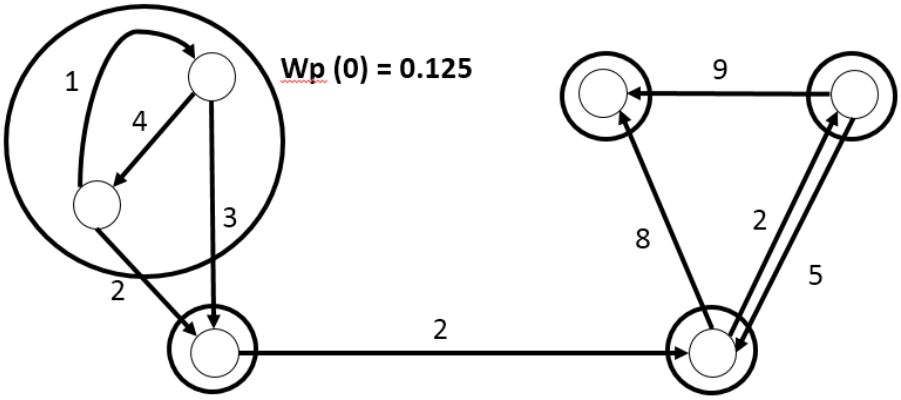
3.2 Enhanced Algorithm

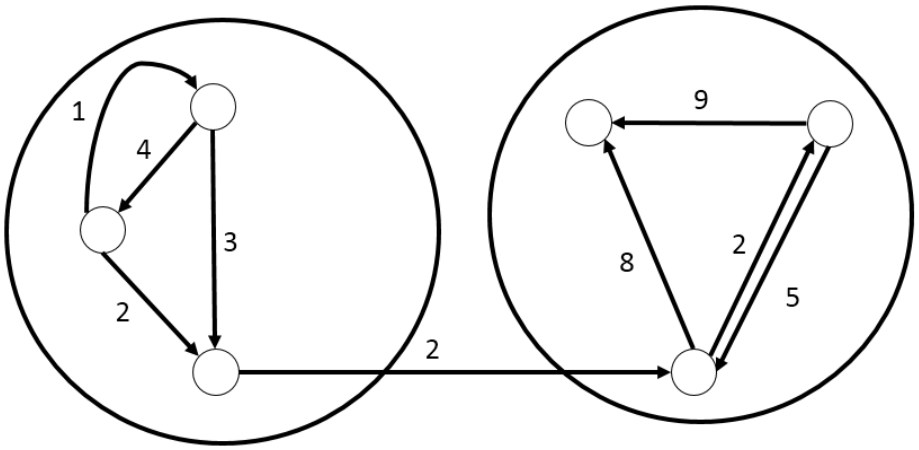







1
4
2
3
2
9
8
2
5
Wp (0) = 0.0536


1
4
3
2
2
9
8
2
5
Wp (0) = 0.125


1
4
3
2
2
9
8
2
5

Code implementation in C:

```
#include<bits/stdc++.h>
#define maxN 20

using namespace std;


int n;

int *array;
int input[maxN+5][maxN+5];

int degree[maxN+2];

int indegree[maxN+2];
int outdegree[maxN+2];
int index[maxN+2];
int community[maxN][maxN];
int tempComm[maxN][maxN];
int comsize[maxN];
int tempsize[maxN];
bool flag[maxN+2];
int comm=0;

int serach(int vertex)
{
    for(int i=0; i<comm; i++)
    {
        for(int j=0; j<comsize[i]; j++)
        {
            if(community[i][j]==vertex)
            {
                return i;

            }
        }

    }


}

double permanence(int vertex)
{
```

```
    double iv=0;
    double emaxv=0;
    double cinv;
    int com;

    com=serach(vertex);

    for(int i=0; i<comsize[com]; i++)
    {
        if(input[vertex][community[com][i]]>=1)
        {
            iv+=input[vertex][community[com][i]];
        }

        if(input[community[com][i]][vertex]>=1)
        {
            iv+=input[community[com][i]][vertex];
        }
    }


    for(int i=0; i<comm ; i++)
    {
        if(i==com) continue;

        double temp=0;
        for(int j=0; j<comsize[i]; j++)
        {
            if(input[vertex][community[i][j]]>=1)
            {
                temp+=input[vertex][community[i][j]];
            }

            if(input[community[i][j]][vertex]>=1)
            {
                temp+=input[community[i][j]][vertex];
            }
        }

        if(temp>emaxv)
            emaxv=temp;

    }

//    else
    //{
        double num=0;
        double denom;
```

```
        double neigh=0;

        for(int i=0; i<comsize[com]; i++)
        {
            if(community[com][i]==vertex) continue;

            if(input[vertex][community[com][i]]>=1 ||
input[community[com][i]][vertex]>=1)
                neigh+=1;

        }



        for(int i=0; i<comsize[com]-1; i++)
        {
            if(community[com][i]==vertex) continue;

            for(int j=i+1; j<comsize[com]; j++)
            {
                if(community[com][j]==vertex) continue;


                if( ( input[vertex][community[com][i]]>=1 ||
input[community[com][i]][vertex]>=1 ) && (
input[vertex][community[com][j]]>=1 ||
input[community[com][j]][vertex]>=1 ) ){

                if(input[ community[com][i] ][ community[com][j]
]>=1)
                {
                    num+=input[ community[com][i] ][
community[com][j] ];
                }


                if(input[ community[com][j] ][ community[com][i]
]>=1)
                {
                    num+=input[ community[com][j] ][
community[com][i] ];
                }

                }

            }
        }
```

```
        denom= neigh*(neigh-1) ;

        if(denom<1) cinv=0;
        else
            cinv=(double)num/(double)denom;

    double perm;
    if(emaxv==0.0) perm= ( (double)iv/ (
indegree[vertex]+outdegree[vertex] ) )  + cinv;
    else perm= ( (double)iv/(emaxv * (
indegree[vertex]+outdegree[vertex] ) ) ) /*- (1-*/ + cinv;
    return perm;

}

int cmp(const void *a, const void *b)
{
    int ia = *(int *)a;
    int ib = *(int *)b;
    return array[ia] > array[ib] ? -1 : array[ia] < array[ib];
}


void seedComm()
{


    comm=n;

    for(int i=0;i<n;i++)
    {
            community[i][0]=i;
            comsize[i]=1;
    }

}

double detectCommnity()
{

    double sum=0;
    int itern=0;
    double old_sum=-1;
    double curpne,curp;
    int maxItr=2;

    while (sum != old_sum && itern < maxItr)
```

```
{
    itern+=1;
    old_sum=sum;
    sum=0;
    for(int i=0; i<n; i++)
    {

        curp=permanence(i);

        curpne=0;

        for(int j=0; j<n; j++)
        {
            if(input[i][j]>=1 || input[j][i]>=1)
            {
                curpne+=permanence(j);
            }
        }

        int com=serach(i),oldcom;
        int com1,oldcom1;


        for(int j=0; j<comsize[com]; j++)
        {
            if(community[com][j]==i)
            {
                com1=j;
                break;
            }
        }


        oldcom=com;
        oldcom1=com1;


        for(int j=0; j<comm; j++)
        {
            if(j==oldcom){
               // printf("\nCom\n");
                continue;
            }


        bool glaf=false;

        for(int kk=0;kk<comsize[j];kk++)
```

```
            {
                if(input[i][community[j][kk]]>0 ||
input[community[j][kk]][i]>0)
                {
                    glaf=true;
                    break;
                }
            }

                for(int x=0; x<comm; x++)
                {
                    for(int z=0; z<comsize[x]; z++)
                    {
                        tempComm[x][z]=community[x][z];
                    }

                    tempsize[x]=comsize[x];
                }


                for(int k=com1; k<comsize[com]; k++)
                {
                    community[com][k]=community[com][k+1];
                }

                comsize[com]-=1;


                community[j][comsize[j]]=i;
                comsize[j]+=1;



                double np=permanence(i);
                double npne=0;

                for(int k=0; k<n; k++)
                {
                    if(input[i][k]>=1 || input[k][i]>=1 )
                        npne+=permanence(k);
                }


                if (curp< np && curpne < npne)
                {

                    curp=np;
```

```
                    com=j;
                    com1=comsize[j]-1;
                }
                else
                {
                    //
memcpy(community,tempComm,sizeof(community));


                    for(int x=0; x<comm; x++)
                    {
                        for(int z=0; z<tempsize[x]; z++)
                        {
                            community[x][z]=tempComm[x][z];
                        }

                        comsize[x]=tempsize[x];
                    }



                }

            }

            sum+=curp;

        }

    }


    return sum/n;


}




int main()
{
    FILE *fp=fopen("ds1.txt","r");

    fscanf(fp,"%d",&n);

    int i,j,w;
    memset(indegree,0,sizeof(indegree));
```

```
    memset(outdegree,0,sizeof(outdegree));

    memset(flag,0,sizeof(flag));

    while(fscanf(fp,"%d %d %d",&i,&j,&w)!=EOF){

            input[i][j]=w;

            if(input[i][j]>=1){
                outdegree[i]+=w;
                indegree[j]+=w;
            }

    }
        seedComm();

    printf("Initial Community:\n\n");

    for(int i=0; i<comm; i++)
    {
        for(int j=0; j<comsize[i]; j++)
        {
            printf("%d ",community[i][j]);
        }
        printf("\n");
    }
    double result=detectCommnity();


    printf("Detected Community:\n\n");

    for(int i=0; i<comm; i++)
    {
        for(int j=0; j<comsize[i]; j++)
        {
            printf("%d ",community[i][j]);
        }
        printf("\n");
    }



    printf("\n\nTotal Permanence %lf\n\n\n",result);

    return 0;
}
```

References

Barthelemy, S. F. (2007). Resolution limit in community detection. *Proceedings of the National Academy of Sciences.*

Benjamin H Good, Y.-A. d. (2004). Performance of modularity maximization in practical contexts. *Physical Review*.

Fortunato, A. L. (2012). Consensus clustering in complex networks.

Fortunato, S. (2010). Community detection in graphs. *Physics reports*.

Jure Leskovec, K. J. (2009). Community structure in large networks: Natural cluster sizes. *Internet Mathematics*.

Newman, E. A. (2008). Community structure in directed networks. *Physical review letters*.

Newman, M. E. (2004). Analysis of weighted networks. *Physical review E*.

Newman, M. E. (2006). Modularity and community structure in networks. *Proceedings of the national academy of sciences*.

Newman., M. E. (2013). Community detection and graph partitioning. *Europhysics Letters*.

Pasquale De Meo, E. F. (2013). Enhancing community detection using a network weighting strategy. *Information Sciences.*

Peter J Mucha, T. R. (2010). Community structure in time-dependent,Community structure in time-dependent,. *Science*.

Tanmoy Chakraborty, S. S. (2016). Permanence and community structure in complex networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*.

Usha Nandini Raghavan, R. A. (2007). Near Linear time algorithm to detect community structures in large-scale networks.

Vincent D Blondel, J.-L. G. (2008). Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*.

Zongqing Lu, Y. W. (2013). Community detection in weighted networks: Algorithms and applications. In Pervasive Computing and Communications (PerCom).

Yong-Yeol Ahn, James P. Bagrow, and Sune Lehmann. 2010. Link communities reveal multiscale complexityin networks. *Nature* 466, (August 2010), 761–764.A. Arenas, A. Fern´andez, and S. G´omez. 2008. Analysis of the structure of complex networks at differentresolution levels. *New Journal of Physics* 10, 5 (2008), 053039.

David A. Bader, Henning Meyerhenke, Peter Sanders, and Dorothea Wagner (Eds.). 2013. Graph partitioningand graph clustering. In *Proceedings of the10th DIMACS Implementation Challenge Workshop.*Contemporary Mathematics, vol. 588. American Mathematical Society.

Jeffrey Baumes, Mark Goldberg, and Malik Magdon-Ismail. 2005. Efficient identification of overlapping communities. In *Proceedings of the 2005 IEEE International Conference on Intelligence and Security Informatics (ISI'05)*. Springer-Verlag, Berlin, 27–36.

Jonathan W. Berry, Bruce Hendrickson, Randall A. LaViolette, and Cynthia A. Phillips. 2011. Tolerating the community detection resolution limit with edge weighting. *Physical Review E* 83, 5 (May 2011), 056119.

Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics* 2008 (2008), P10008.

Tanmoy Chakrabort, Sandipan Sikdar, Vihar Tammana, Niloy Ganguly, and Animesh Mukherjee. 2013. Computer science fields as ground-truth communities: Their impact, rise and fall. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining(ASONAM'13)*. ACM, New York, NY, 426–433.

Tanmoy Chakraborty. 2015. Leveraging disjoint communities for detecting overlapping community structure. *Journal of Statistical Mechanics: Theory and Experiment* 2015, 5 (2015), P05017.

Tanmoy Chakraborty, Ayushi Dalmia, Animesh Mukherjee, and Niloy Ganguly. 2016a. Metrics for community analysis: A survey. *CoRR* abs/1604.03512 (2016).

Tanmoy Chakraborty, Suhansanu Kumar, Niloy Ganguly, Animesh Mukherjee, and Sanjukta Bhowmick. 2016b. GenPerm: A unified method for detecting non-overlapping and overlapping communities. *CoRR* abs/1604.03454 (2016).

Tanmoy Chakraborty, Sriram Srinivasan, Niloy Ganguly, Sanjukta Bhowmick, and Animesh Mukherjee. 2013. Constant communities in complex networks. *Scientific Reports* 3, (May 2013). DOI:10.1038/srep01825

Tanmoy Chakraborty, Sriram Srinivasan, Niloy Ganguly, Animesh Mukherjee, and Sanjukta Bhowmick. 2014. On the permanence of vertices in network communities. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'14)*. ACM, New York, NY, 1396–1405. DOI:http://dx.doi.org/10.1145/2623330.2623707.