Development & Application of a Menu-Driven Tool for Network Analysis

Submitted by
Nitin Kumar
(Reg. no 71320)

Supervision
Prof. Andrew M. Lynn

Covered

Development of a Menu-driven Tool for Network Analysis

Application:
Pipeline for evaluating *Pairwise-Disconnectivity index*
Tested on :

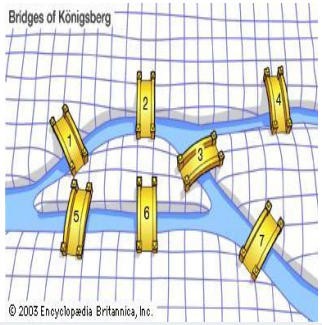
*****Mycobacterium tuberculosis H37Rv*****
Lactobacillus helveticus DPC 4571
Lactobacillus fructivorans KCTC 3543
Mycoplasma hyorhinis
Streptococcus infantarius

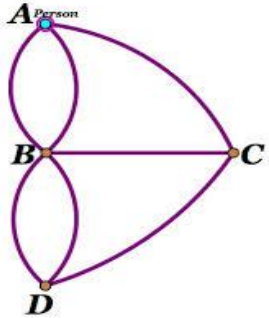
But Before we proceed

Some background about Graphs & Networks

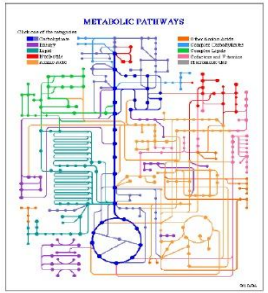
Graphs around us

- Most famous *konigsberg Bridge* problem[1]
- Social Networks
- Biological Network

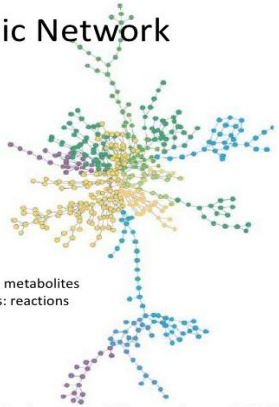




E. Coli Metabolic Network



Kegg, Wit, Biocyc, Bigg (UCSD)



Nodes: metabolites
Edges: reactions

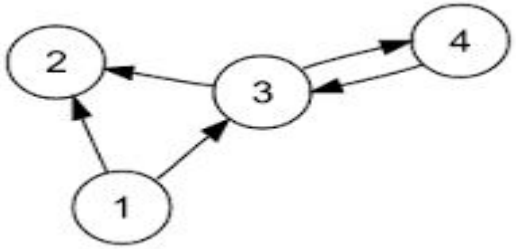
Guimera and Nunes Amaral 2005

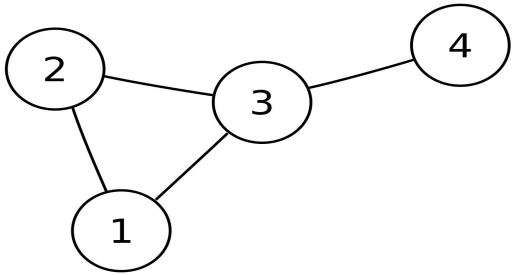
Ref : wersm.com/how-much-data-is-generated-every-minute-on-social-media/

Basics of Graph Theory

Graph is simply ***G = (V,E)*** consist of where V is set of vertices *(v1,v2,v3,.....,vn)* ,Vertices are often sometimes called as nodes in graph and E is the set of edges joining vertices *(e1,e2,e3,....,e4)*.[1]

Types of graph





Directed graph

Undirected Graph

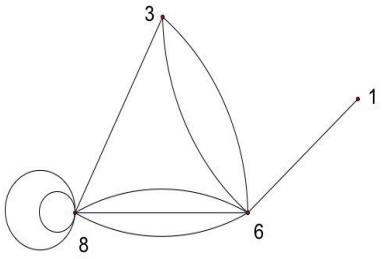
Some More Parameters

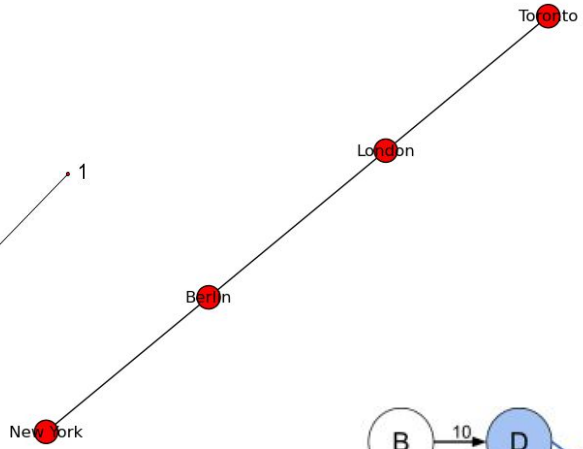
- Degree
- Degree distribution
- Path
- Average path length
- Clustering coefficient

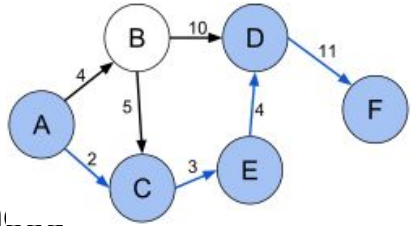
$$C_i = \frac{2\,n_i}{k_i\,(k_i - 1)}$$

Where *ni* is number of links between the vertices within its neighbourhood

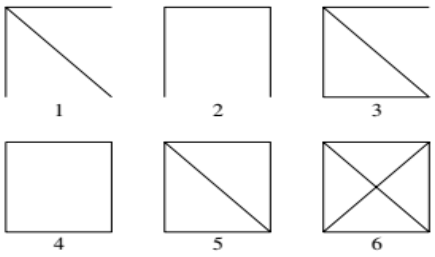
- Shortest path
- Neighbours







Some More Parameter(cont.)



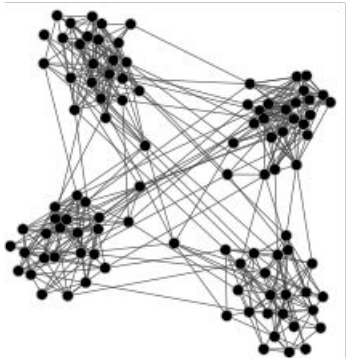
- Functional motifs

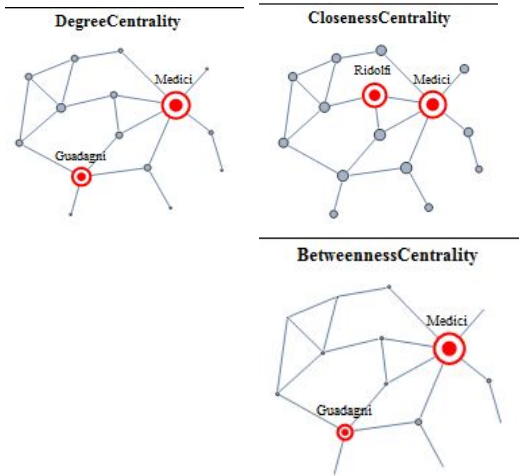
  Network motifs are subgraphs that repeat themselves

- Modularity

  Measure the strength of division of a network into modules (also called groups, clusters or communities).Biological networks, including exhibit a high degree of modularity.

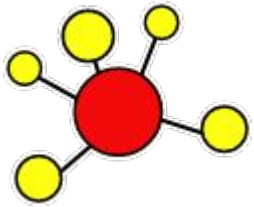
- Centrality





Packages, Database & Libraries Used

Python-igraph



Matplotlib

STRING db - Bioconductor

String Database

Pairwise Disconnectivity index

pairwise disconnectivity index of vertex v **is** ***Dis*(*v*)** **is the fraction of those initially connected pairs of vertices in a network which becomes disconnected if vertex *v* is removed from the network**

$$Dis(v)=(No-Nv)/No$$

Where *No* is number of ordered pairs in the network that are Connected by at least one directed path , *Nv* is the numbers of ordered pairs left connected after removal of the vertex *V* . *No* > 0 because there there exist at least one path between the pair of nodes .[2]

Where in the extreme case the removal of vertex v destroys all communication in a network resulting in Dis(v) = 1. In contrast, Dis(v) = 0 refers to a non-crucial vertex which is obviously not connected to any other vertex in a network .

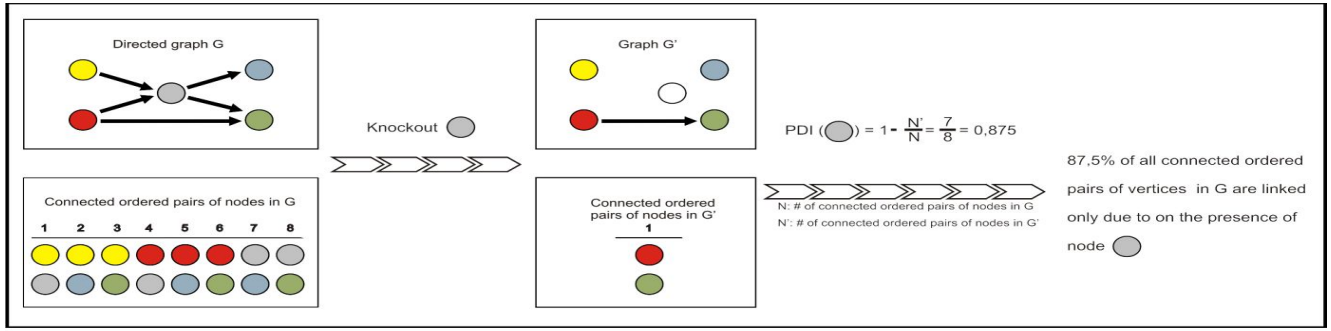


Figure 1. Estimating the pairwise disconnectivity index (PDI) of a vertex in an example network.

ref: diva.sybig.de/theory.php

Pairwise Disconnectivity index(cont).

- Measure how crucial an individual element is for sustaining the communication between connected pairs of vertices in a network Index measure of topological importance

- A measure of sensitivity of network In the presence (absence) of each individual element. It's been applied to the analysis of several regulatory networks from various organisms and interactions.

- The importance of an individual vertex or edge for the coherence of the network is determined by the particular position of the given element in the whole network.

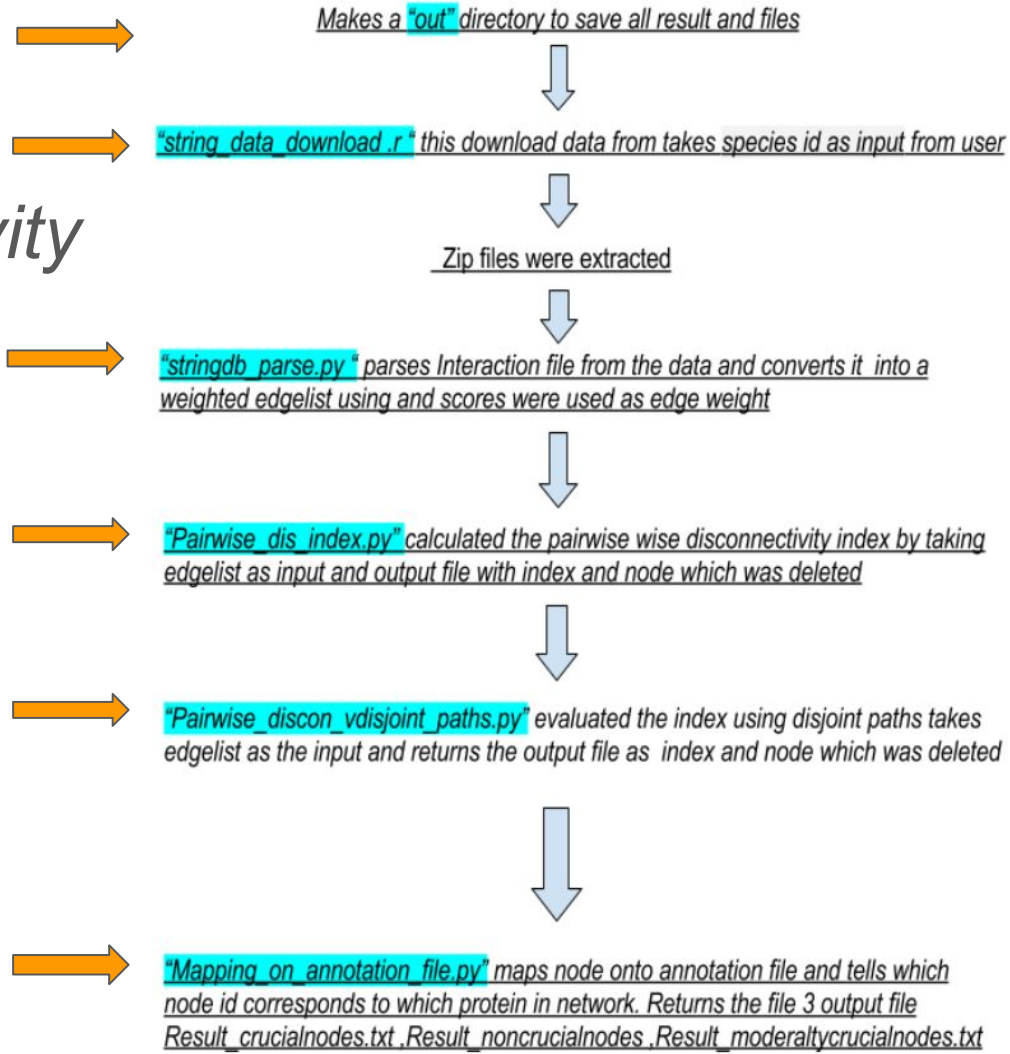
- Ability in systematically analyzing the role of every element, as well as groups of elements, in a regulatory or biological network.

- Can be evaluated for a Nodes or Edges

Method & workflow followed

Pipeline *Pairwise-Disconnectivity index*

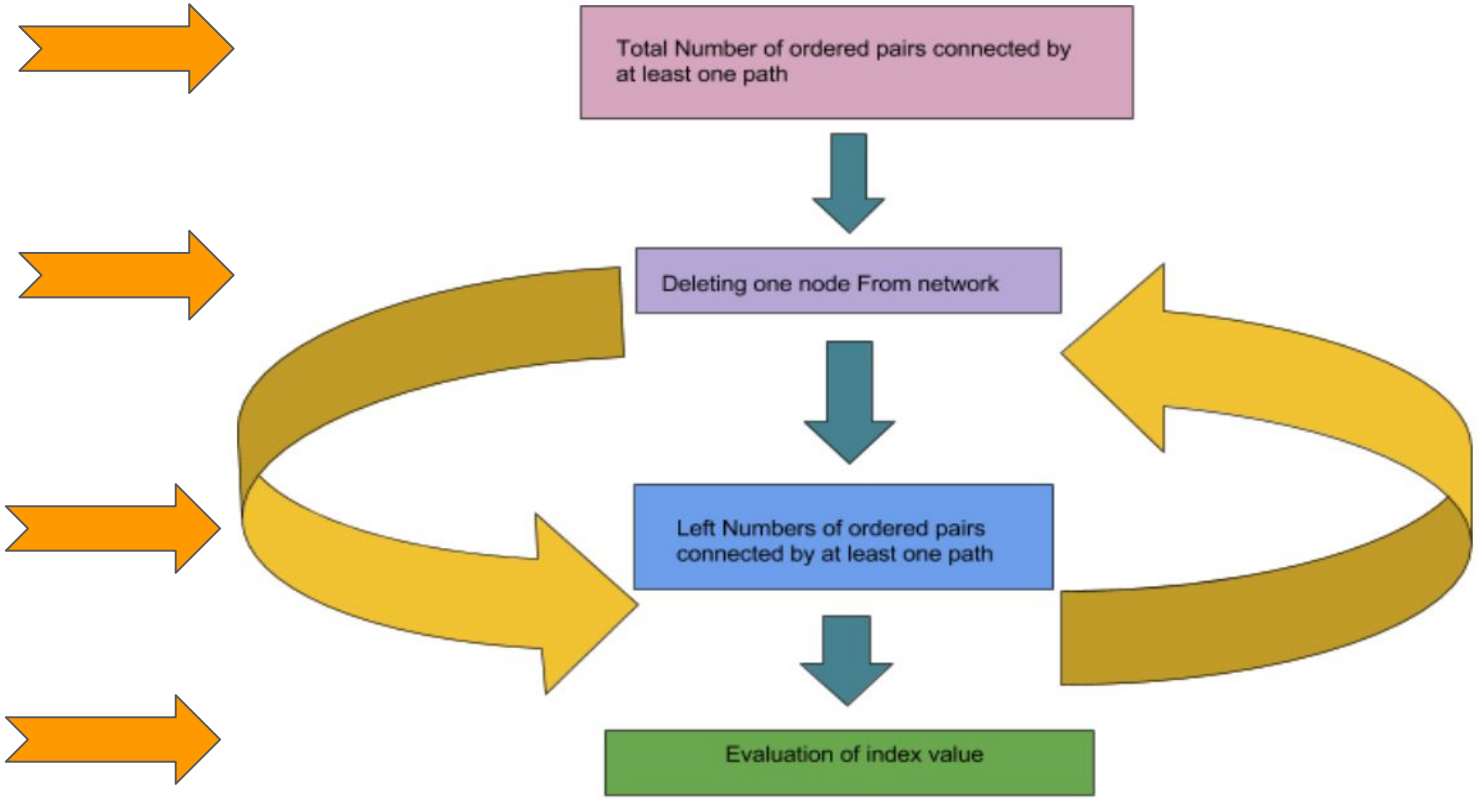
- *X.sh* bash script



Scripts used in pipeline

- String _data_download.r (Rscript for downloading data from string database for specified organism)
- Stringdb_parse.py (Parse the string data file and converts it into a weighted edgelist format)
- Pairwise_dis_index.py (Evaluates the index value)
- Pairwise_discon_vdisjoint_paths.py (Evaluates the index value using disjoint paths)
- mapping _on_annotation_file.py (maps the nodes onto the annotation file taken from string database)

** scripts can also be used as standalone by simply parsing inputs as system argument (for more refer to usage section mentioned in thesis)



Done iteratively for each of nodes in networks

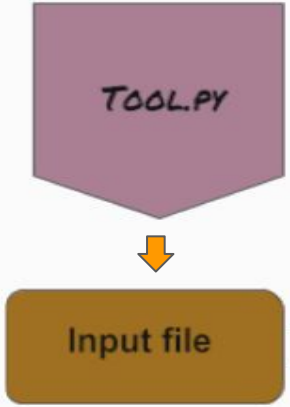
Tool for Network Analysis

- python script Developed using python-igraph library .
- Supports Network data in major formats ***"Graphml file"*** , ***"Adjacency matrix"*** , " ***Edgelist format*** " , " ***Weighted Edgelist*** " , " ***Lgl format*** " as in input file .
- Offers generation of Random network to benchmark your data and compare it against random graph.
- Offers evaluation many parameters for analysis also knocking out node addition of node can be easily done
- Parameters can be easily computed by simply pressing user input as integer
- Works with both python 2.7.12 and python 3

**for more refer to usage section given in thesis

Network Analysis Tool

- Tool.py



Parameter	user input
• Histogram of degree	1
• Centrality	
○ *Eigenvector centrality	2
○ *Betweenness centrality	3
• Average path length	4
• Degree distribution	5
• Clustering coefficient	6
• Shortest path between two pair of nodes	7
• Shortest path between all nodes	8
• Degree distribution power law	9
• Functional motifs	10
• Neighbour vertex	
*for two specified pair of vertex	11
*for all vertex of graph	12
• Modularity	13
• Connectivity	
Vertex *for given two pairs vertex.	14
■ *overall.	15
Edge *for given two pairs vertex	16
■ *overall	17
• No. of clusters	18
• Adjacency matrix of network	19
• Given node id its corresponding node attribute	20
• All node ids and their corresponding node attribute	21
• Edgelist.	22
• Diameter of graph	23
• Average path length	24
• Giant_component into a file	25
• Add vertex(single)	26
• Add vertices(many)	27
• Delete vertex(single)	28
• Delete vertices(many)	29
• Maximum degree nodes	30
• Minimum degree nodes.	31
• Deleting all node one by one saving in file to check how it affects	32

Inputs :-

19,22,25,26,27,28,29
will write a file containing graph

Graphml Files in temp directory

```
usage -format [filename]

format: adjecncy matrix -adj  edgelist -edgelist  graphml -graphml  lgl -lgl  random network - random


=========== parameters covered ============
Degree Distribution Histogram...........(1)
Centrality :
        *Eigenvector centrality.........(2)
        *Betweenness centrality.........(3)
Average path length.....................(4)
Degree distribution.....................(5)
Clustering coefficient..................(6)
Shortest path between two nodes.........(7)
Shortest path between all nodes.........(8)
Degree distribution power law...........(9)
Functional motifs.......................(10)
Neighbour vertex :
        * for two specified vertes.....(11)
        * for all vertex of graph......(12)
Modularity..............................(13)
connectivity :
        vertex *for given two vertex...(14)
               *overall................(15)
        Edge   *for given two vertex...(16)
               *overall................(17)
No. of clusters.........................(18)
Adjacency matrix........................(19)
given node id its EC no(for enzyme data file)..(20)
EC no for all node ids(for enzyme data file)...(21)
Edgelist................................(22)
Diameter................................(23)
Average path length.....................(24)
giant_component.........................(25)
add vertex(single)..............  ..   (26)
add vertices(many)..................   (27)
delete vertex(single)................ .(28)
delete vertices(many).... ........... .(29)
maximum degree nodes.... ........... . (30)
minimum degree nodes.... ...........  .(31)
Deleting all nodes saving in file .....(32)

Type the no of function wanted:
```

Example of a single menu-driven command

```
if user == 4 :
    print ('average path length', g.average_path_length(directed=False,unconn=True))

if user == 10 :
    print (' no. of functional motif ', g.motifs_randesu_no(size=3,cut_prob=None,))
```

- Average path length
- Number of Functional Motif

Example of a Single menu-driven command

- Deleting all nodes saving in file …(32)

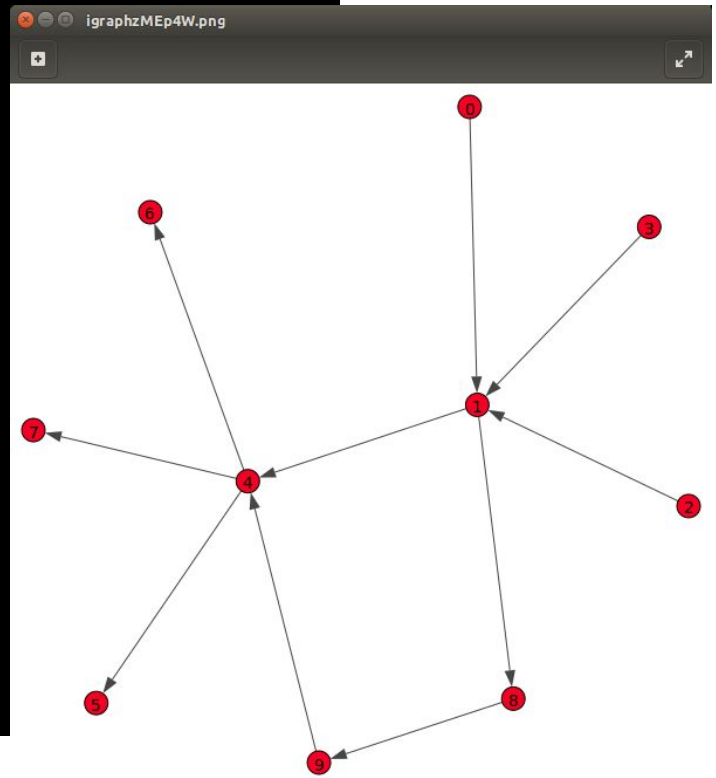
Done iteratively for all pair of nodes knocking out each

One by one and writing it out into a file.

```
if user == 32:
    num=0
    j=0
    while num == 0:
        g.delete_vertices(0)
        if len(g.vs) ==1:
            break
        g.write_edgelist(str(os.getcwd())+"/temp/deleted_node_"+str(j)+".txt")
        print ( 'files complete and saved after deleting node :- ' , str(j))
        j +=1
```

- Also for shortest path between all pairs of node is done similarly iterating over all pairs of nodes

```
$ python tool.py -edgelist ./out_83332/parsed_out.csv
```

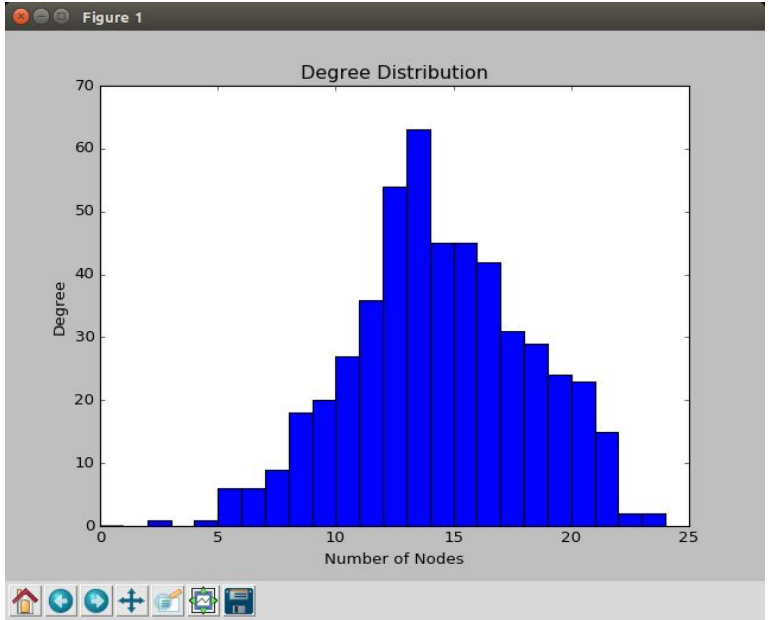


```
acer@acer:~/Desktop/project_dessertation/tool$ python tool.py --help
        usage -format [filename]

        format: adjecncy matrix -adj  edgelist -edgelist  graphml -graphml  lgl -lgl  random network - random


        =========== parameters covered ============
        Degree Distribution Histogram...........(1)
        Centrality :
                *Eigenvector centrality.........(2)
                *Betweenness centrality.........(3)
        Average path length.....................(4)
        Degree distribution.....................(5)
        Clustering coefficient..................(6)
        Shortest path between two nodes.........(7)
        Shortest path between all nodes.........(8)
        Degree distribution power law...........(9)
        Functional motifs.......................(10)
        Neighbour vertex :
                * for two specified vertes.....(11)
                * for all vertex of graph......(12)
        Modularity..............................(13)
        connectivity :
                vertex *for given two vertex...(14)
                       *overall................(15)
                Edge   *for given two vertex...(16)
                       *overall................(17)
        No. of clusters.........................(18)
        Adjacency matrix........................(19)
        given node id its EC no(for enzyme data file)..(20)
        EC no for all node ids(for enzyme data file)...(21)
        Edgelist................................(22)
        Diameter................................(23)
        Average path length.....................(24)
        giant_component.........................(25)
        add vertex(single)..............  ..   (26)
        add vertices(many)..................   (27)
        delete vertex(single)................ .(28)
        delete vertices(many).... ........... .(29)
        maximum degree nodes.... ........... . (30)
        minimum degree nodes.... ...........  .(31)
        Deleting all nodes saving in file .....(32)

Type the no of function wanted: 1
```



```
acer@acer:~/Desktop/project_dessertation/tool/final scripts$ python tool.py -random
        usage -format [filename]

        format: adjacencncy matrix -adj  edgelist -edgelist  graphml -graphml  lgl -lgl  random network - random

Enter the number of nodes 500

        =========== parameters covered ============
        Histogram.................................(1)
        Centrality :
                *Eigenvector centrality.........(2)
                *Betweenness centrality.........(3)
        Average path length.......................(4)
        Degree distribution.......................(5)
        Clustering coefficient...................(6)
        Shortest path between two nodes.........(7)
        Shortest path between all nodes.........(8)
        Degree distribution power law...........(9)
        Functional motifs........................(10)
        Neighbour vertex :
                * for two specified vertes.....(11)
                * for all vertex of graph......(12)
        Modularity...............................(13)
        connectivity :
                vertex *for given two vertex...(14)
                       *overall................(15)
                Edge   *for given two vertex...(16)
                       *overall................(17)
        No. of clusters..........................(18)
        Adjacency matrix.........................(19)
        given node id its EC no(for enzyme data file)..(20)
        EC no for all node ids(for enzyme data file)...(21)
        Edgelist.................................(22)
        Diameter.................................(23)
        Average path length.....................(24)
        giant_component.........................(25)
        add vertex(single)..............  ..   (26)
        add vertices(many)..................   (27)
        delete vertex(single)................ .(28)
        delete vertices(many).... ........... .(29)
        maximum degree nodes.... ........... . (30)
        minimum degree nodes.... ...........  .(31)
        Deleting all nodes saving in file .....(32)

Type the no of function wanted: 4
```

```
('average path length', 6.9352785571422 9)
```

```
Type the no of function wanted: 7
Enter the name(which is integer here) of the node from which the shortest path is to be calculated : 1
Enter the name(which is integer here) of the node to which the shortest path is to be calculated : 3
('shortest path', [[1, 29, 14, 6, 19, 22, 24, 3], [1, 29, 14, 6, 19, 27, 24, 3], [1, 29, 14, 6, 19, 32, 24, 3]])
```

```
Type the no of function wanted: 10
(' no. of functional motif ', 30166)
```

```
Type the no of function wanted: 6
('clustering coefficient', 0.5995809380540482)
```

Tool.py

```
Type the no of function wanted: 23
Diameter is  17
```

```
Type the no of function wanted: 26
 single vertex to be added attribute:  50
None
new added vertice is saved in graphml file
```

```
Type the no of function wanted: 5
degree distribution (degree, vertex id) file is saved degree_distribution.txt
```

Network Analysis Tool (cntd.)

Tool currently contains 32 functions

Wrapper is Developed around the igraph

Easy to use simply by pressing user input returns the output

Input file can be passed as system argument in all major formats

Graphs , plots and output file are saved automatically

Applying across species

Pipeline and tool was used to analyse on dataset of following organism

Species ID	ORGANISM
83332	*Mycobacterium tuberculosis H37Rv*
405566	*Lactobacillus helveticus DPC 4571*
941770	*Lactobacillus fructivorans KCTC 3543*
1050720	*Agrobacterium tumefaciens F2*
1129369	*Mycoplasma hyorhinis*
1069533	*Streptococcus infantarius*

Results

The study was carried out and specie Id. 83332 , *Mycobacterium tuberculosis H37Rv*

Total number of nodes in network : 3967

Crucial nodes	Index value 1.0	1422
Moderate nodes	Index value 0 > 1.0	712
Non crucial nodes	Index value 0.0	409

After mapping it on annotation file

Crucial Nodes index value 1.0

1.0	83332.Rv1140	83332.Rv1140	hypothetical	MT1173
1.0	83332.Rv1144	83332.Rv1144	short-chain	Rv1144
1.0	83332.Rv1248c	83332.Rv1248c	alpha-ketoglutarate	kgd
1.0	83332.Rv1279	83332.Rv1279	dehydrogenase	Rv1279
1.0	83332.Rv1350	83332.Rv1350	3-ketoacyl-ACP	fabG2
1.0	83332.Rv1484	83332.Rv1484	enoyl-ACP	inhA
1.0	83332.Rv1533	83332.Rv1533	hypothetical	Rv1533
1.0	83332.Rv1715	83332.Rv1715	3-hydroxybutyryl-CoA	fadB3

Figure : Truncated view of crucial node result (columns showing index value ,deleted protein and corresponding annotated protein respectively)

Moderate Nodes valued 0 < 1.0

0.964912280702	83332.Rv0551c	83332.Rv0551c	acyl-CoA	fadD8
0.916666666667	83332.Rv0554	83332.Rv0554	bromide	bpoC
0.5	83332.Rv0562	83332.Rv0562	polyprenyl	grcC1
0.833333333333	83332.Rv0636	83332.Rv0636	(3R)-hydroxyacyl-ACP	hadB
0.75	83332.Rv0904c	83332.Rv0904c	acetyl-CoA	accD3
0.857142857143	83332.Rv1013	83332.Rv1013	long-chain-fatty-acid--CoA	pks16
0.5	83332.Rv1070c	83332.Rv1070c	enoyl-CoA	echA8
0.8	83332.Rv1106c	83332.Rv1106c	cholesterol	Rv1106c
0.833333333333	83332.Rv1193	83332.Rv1193	acyl-CoA	fadD36
0.5	83332.Rv1427c	83332.Rv1427c	acyl-CoA	fadD12
0.6	83332.Rv1472	83332.Rv1472	enoyl-CoA	echA12
0.5	83332.Rv1483	83332.Rv1483	3-oxoacyl-ACP	fabG
0.666666666667	83332.Rv1521	83332.Rv1521	acyl-CoA	fadD25
0.5	83332.Rv1532c	83332.Rv1532c	hypothetical	MT1583
0.857142857143	83332.Rv1661	83332.Rv1661	polyketide	pks7

Figure : Truncated view of moderate valued nodes result (columns showing index value ,deleted protein and corresponding annotated protein respectively)

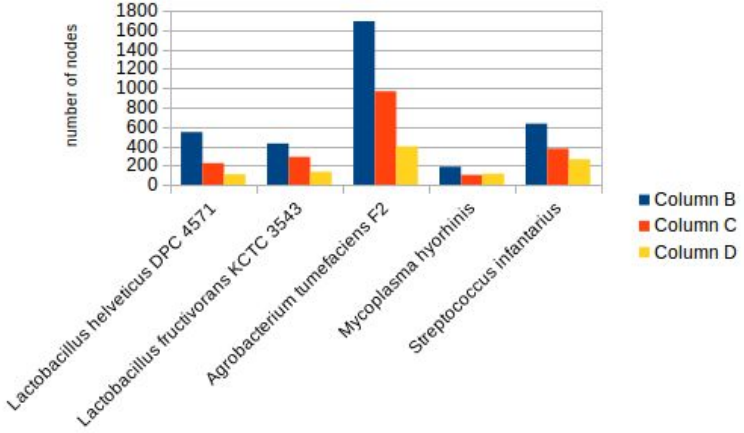
Non-crucial Nodes index 0.0

0.0	83332.Rv0658c	83332.Rv0658c	hypothetical	MT0687
0.0	83332.Rv0672	83332.Rv0672	acyl-CoA	fadE8
0.0	83332.Rv0675	83332.Rv0675	enoyl-CoA	echA5
0.0	83332.Rv0764c	83332.Rv0764c	cytochrome	cyp51
0.0	83332.Rv0766c	83332.Rv0766c	cytochrome	cyp123
0.0	83332.Rv0768	83332.Rv0768	aldehyde	aldA
0.0	83332.Rv0860	83332.Rv0860	fatty	fadB
0.0	83332.Rv0905	83332.Rv0905	enoyl-CoA	echA6
0.0	83332.Rv0906	83332.Rv0906	hypothetical	Rv0906
0.0	83332.Rv0914c	83332.Rv0914c	acetyl-CoA	MT0939
0.0	83332.Rv0945	83332.Rv0945	short-chain	Rv0945
0.0	83332.Rv0971c	83332.Rv0971c	enoyl-CoA	echA7
0.0	83332.Rv0972c	83332.Rv0972c	acyl-CoA	fadE12

Figure : Truncated view of non-crucial nodes result (columns showing index value ,deleted protein and corresponding annotated protein respectively)

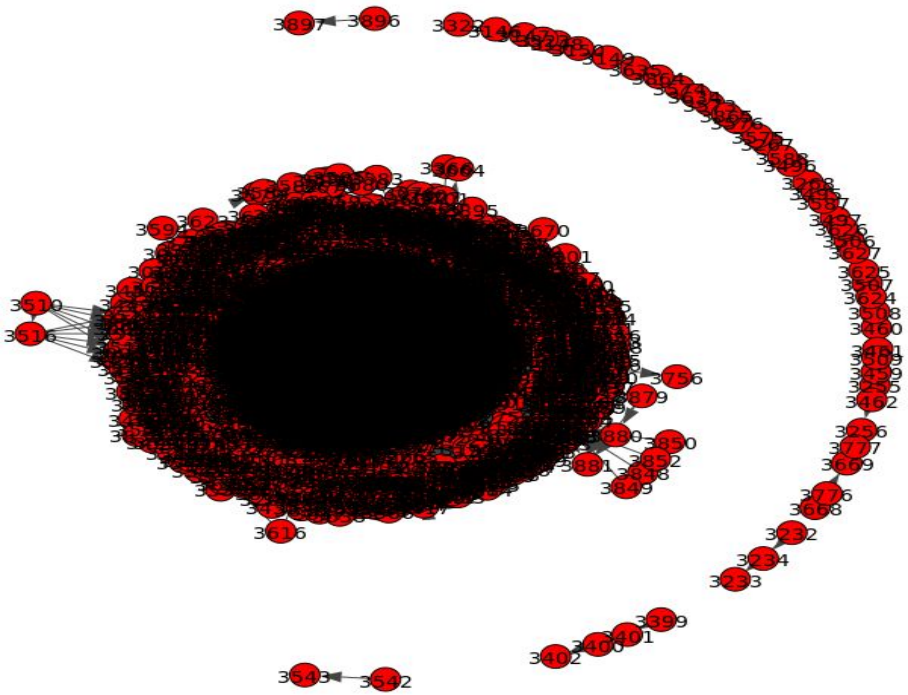
After applying on other datasets

	Lactobacillus helveticus DPC 4571	*Lactobacillus fructivorans KCTC 3543*	*Agrobacterium tumefaciens F2*	*Mycoplasma hyorhinis*	*Streptococcus infantarius*
Crucial nodes	545	427	1690	186	632
Moderate nodes	225	289	967	103	373
Non-crucial nodes	109	134	396	115	264



Further Downstream analysis using Tool.py

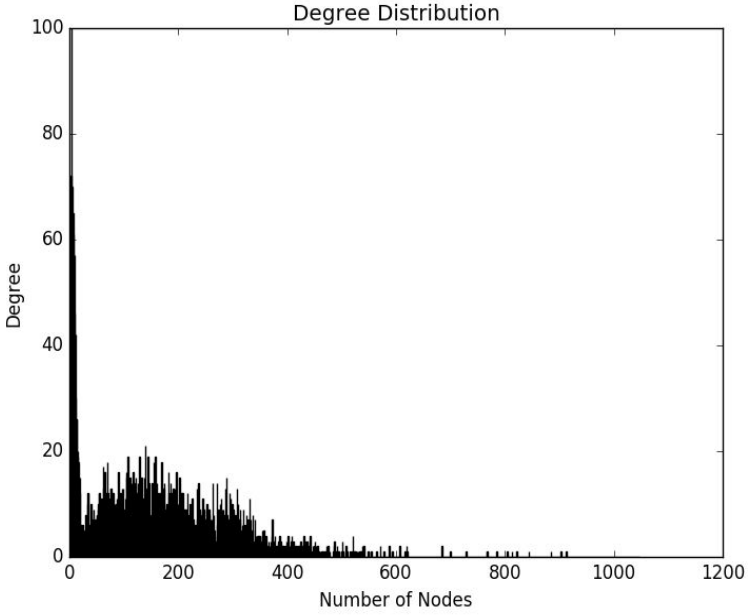
Mycobacterium tuberculosis **H37Rv**



visualisation and plot

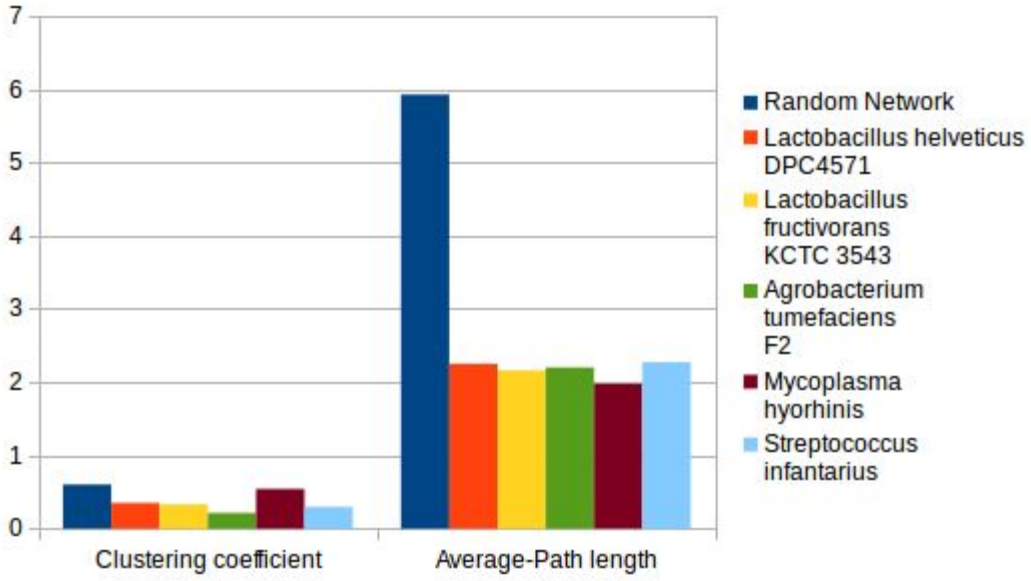
Mycobacterium tuberculosis H37Rv

- Clustering coefficient : 0.24846675631890996
- Average path length : 2.326801269355572
- Number of functional motif : 69397903
- Diameter : 12
- Modularity : 0.00219002235829123
- Number of of clusters of the graph', 3967



For other species

Parameter	*Lactobacillus helveticus DPC4571*	*Lactobacillus fructivorans KCTC 3543*	*Agrobacterium tumefaciens F2*	*Mycoplasma hyorhinis*	*Streptococcus infantarius*
Clustering coefficient	0.350852	0.330410	0.213433	0.54137795	0.2984035
Average-Path length	2.250659	2.160105	2.200877	1.97746338	2.2720507
Number Of functional motif	139075 03	9254957	126324842	4035184	13539239

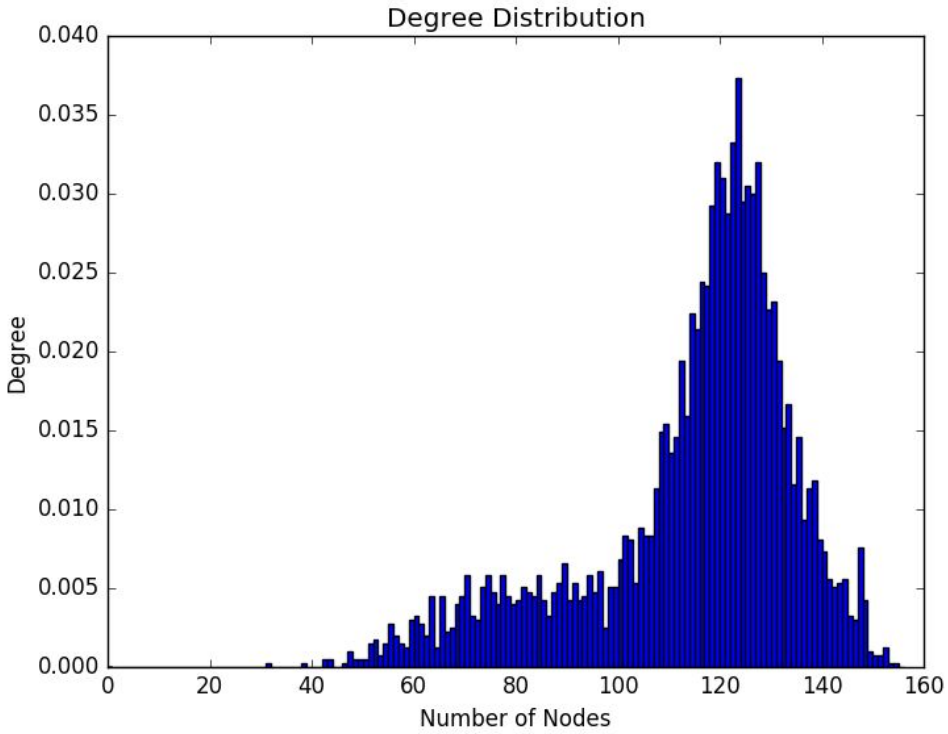

7
6
5
4
3
2
1
0
Clustering coefficient
Average-Path length
Random Network
Lactobacillus helveticus DPC4571
Lactobacillus fructivorans KCTC 3543
Agrobacterium tumefaciens F2
Mycoplasma hyorhinis
Streptococcus infantarius

Random Network

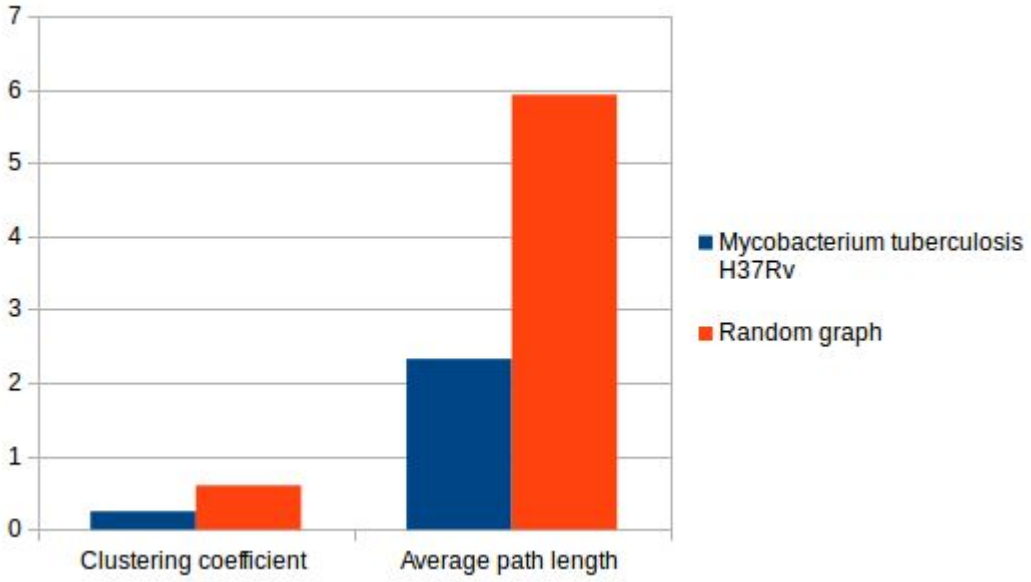
Nodes : 3967

Clustering coefficient : 0.602889

Average path length : 5.926425



Random graph & *Mycobacterium tuberculosis H37Rv*



Conclusion

Study was carried out for evaluation of important protein in the different organism **showing large number of nodes connected together** in a cluster adding onto robustness of the network

Networks behaved as **Barabási–Albert (BA) model** showing **high clustering coefficient and shorter average path length when compared to random network** of same number of nodes .[3]

In all the organism chosen as dataset showed some common property as high clustering coeff. And shorter average path length revealing that most of the nodes are clustered together and knocking out large number of nodes will disrupt the network

A giant component which exists within the network **containing high number of nodes CLUSTERED TOGETHER** . Also important nodes out of the network were identified using the disconnectivity index and further analysed through Tool.py using python igraph library for the data downloaded from string database using pipeline.

Index value returns the important nodes within the network and act as a good topological measure

Recommendations and Future possibilities

- **These nodes Can act as drug target**
- **Can tell about biology of organism ,**
- **System level study can be predicted**
- **Treated as vaccine candidates**
- **Also search for sample targets decreased from thousands to few hundred only**
- **Currently only 32 functions are there in tool more can be added**

Reference

[1]Harary, F. (1969). Graph theory. Reading, Mass: Addison-Wesley Pub. Co.

[2] Potapov, Anatolij P., Björn Goemann, and Edgar Wingender. "The pairwise disconnectivity index as a new metric for the topological analysis of regulatory networks." BMC bioinformatics 9.1 (2008): 227.

[3] Barabási, Albert-László, and Réka Albert. "Emergence of scaling in random networks." science 286.5439 (1999): 509-512

[4] Hunter, John D. "Matplotlib: A 2D graphics environment." Computing In Science &Engineering 9.3 (2007): 90-95.

[5] Csardi G, Nepusz T: The igraph software package for complex network researchInterJournal, Complex Systems 1695. 2006. http://igraph.org Corresponding BibTeX entry:

[6]Mason, Oliver, and Mark Verwoerd. "Graph theory and networks in biology." IETsystems biology 1.2 (2007): 89-119.

[7] de Silva, Eric, and Michael PH Stumpf. "Complex networks and simple models in biology." Journal of the Royal Society Interface 2.5 (2005): 419-430.

Thank you !!