RESEARCH ARTICLE OPEN ACCESS

Graph Theory: Graph Based Feature Extraction of Different Types of Graphs

Mr. Saptarshi Bhattacharyya [1], Mr. Abhijit Bera [2]
Research Associate [1], Guest Lecturer [1]Teacher [2]
West Bengal University of Technology (MAKAUT)
WB – India

ABSTRACT
In the domain of mathematics and computer science, graph theory is the study of graphs that concerns with the relationship among edges and vertices. It is a popular subject having its applications in computer science, information technology, biosciences, mathematics, and linguistics to name a few. Graph classification is an important data mining task for which many methods are already implemented. Yet in this paper graph based features extracted and various relevant features are selected using feature selection algorithms.
Keywords:- Graph, path, circuit, clustering, eccentricity, eigen value, trace, cello graph, protein graph.

I. INTRODUCTION

What is a Graph?

Graph: A graph is a pair of sets (V, E), where V is the set of vertices and E is the set of edges, connecting the pairs of vertices. Take a look at the following graph:

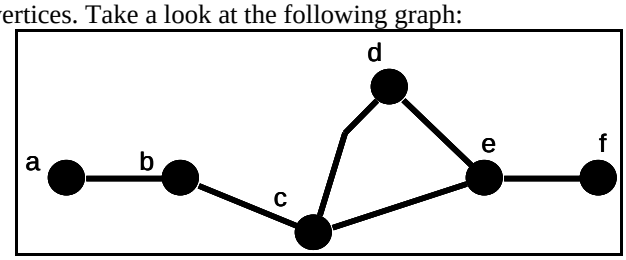


Fig. 1

Vertices: Vertices are also known as nodes, points and (in social networks) as actors, agents or players.

Edges: Edges are also known as lines and (in social networks) as ties or links. An edge e = (u,v) is defined by the unordered pair of vertices that serve as its end points.

As an example, the graph depicted in Figure 1 has vertex set V={a,b,c,d,e.f} and edge set E = {(a,b),(b,c),(c,d),(c,e),(d,e),(e,f)}.

Adjacency: Two vertices *u* and *v* are *adjacent* if there exists an edge (*u*,*v*) that connects them.

Incidence: An edge (u,v) is said to be *incident* upon nodes u and v.

Loop: A loop is a special type of edge that connects a vertex to itself. An edge that links a vertex to itself is known as a *self-loop* or *reflexive* tie.

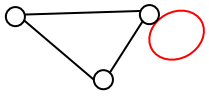
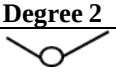


Fig. 2

Degree of a vertex: The degree of a vertex is the number of edges meeting at that vertex. It is possible for a vertex to have a degree of zero or larger.

Degree 0	Degree 1	Degree 2	Degree 3	Degree 4
				

Path: A path is a sequence of vertices using the edges. Usually we are interested in a path between two vertices. For example, a path from vertex A to vertex M is shown below. It is one of many possible paths in this graph.

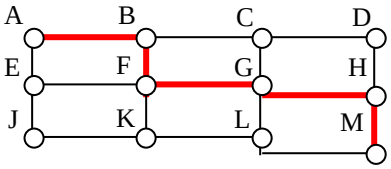


Fig. 3

Circuit. A circuit is a path that begins and ends at the same vertex. A circuit starting and ending at vertex A is shown below.

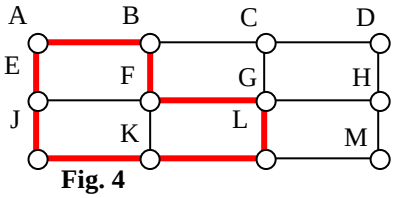


Fig. 4

Adjacency matrix: Every graph has associated with it an *adjacency matrix*, which is a binary *n*×*n* matrix A in which a_{ij} = 1 and a_{ji} = 1 if vertex vi is adjacent to vertex vj, and aij = 0 and aji = 0 otherwise. The natural graphical representation of an adjacency matrix is a table, such as shown in Figure 2.

	a	b	c	d	e	f
a	0	1	0	0	0	0

b	1	0	1	0	0	0
c	0	1	0	1	1	0
d	0	0	1	0	1	0
e	0	0	1	1	0	1
f	0	0	0	0	1	0

Fig. 5. Adjacency matrix for graph in Figure 1.

Subgraphs: A *subgraph* of a graph *G* is a graph whose points and lines are contained in *G*. A complete subgraph of *G* is a section of *G* that is complete (i.e., has density = 1).

Cliques: A clique is a maximal complete subgraph. A maximal complete subgraph is a subgraph of G that is complete and is maximal in the sense that no other node of G could be added to the subgraph without losing the completeness property. In Figure 1, the nodes {c,d,e} together with the lines connecting them form a clique. Cliques have been seen as a way to represent what social scientists have called primary groups.

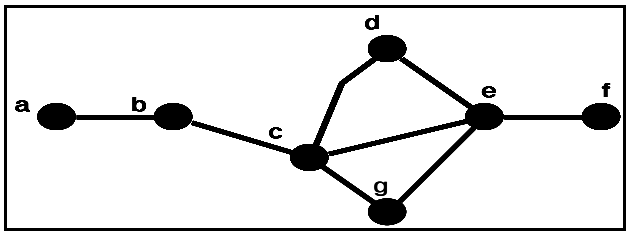


Fig. 6.

Component. A *component* of a graph is defined as a maximal subgraph in which a path exists from every node to every other (i.e., they are mutually reachable). The size of a component is defined as the number of nodes it contains. A connected graph has only one component.

Walk: A sequence of adjacent vertices $v_0,v_1,\ldots,v_n$ is known as a *walk*. In Figure 6, the sequence *a,b,c,b,c,g* is a walk. A walk can also be seen as a sequence of *incident* edges, where two edges are said to be incident if they share exactly one vertex.

Closed: A walk is closed if $v_o = v_n$.

Path: A walk in which no vertex occurs more than once is known as a *path*. In Figure 6, the sequence *a,b,c,d,e,f* is a path.

Trail: A walk in which no edge occurs more than once is known as a *trail*. In Figure 6, the sequence *a,b,c,e,d,c,g* is a trail but not a path. Every path is a trail, and every trail is a walk.

Cycle: A *cycle* can be defined as a closed path in which *n* >= 3. The sequence *c,e,d* in Figure 6 is a cycle.

Tree: A *tree* is a connected graph that contains no cycles. In a tree, every pair of points is connected by a unique path. That is, there is only one way to get from A to B.

Length: The length of a walk (and therefore a path or trail) is defined as the number of edges it contains. For example, in Figure 6, the path *a,b,c,d,e* has length 4.

Geodesic. A walk between two vertices whose length is as short as any other walk connecting the same pair of vertices is called a *geodesic*. Of course, all geodesics are paths. Geodesics are not necessarily unique. From vertex *a* to vertex *f* in Figure 6, there are two geodesics: *a,b,c,d,e,f* and *a,b,c,g,e,f*.

Distance. The *graph-theoretic distance* (usually shortened to just "distance") between two vertices is defined as the length of a geodesic that connects them. If we compute the distance between every pair of vertices, we can construct a distance matrix *D* such as depicted in Figure 7. The maximum distance in a graph defines the graph's *diameter*. As shown in Figure 7, the diameter of the graph in Figure 1 is 4. If the graph is not connected, then there exist pairs of vertices that are not mutually reachable so that the distance between them is not defined and the diameter of such a graph is also not defined.

	a	b	c	d	e	f	g
a	0	1	2	3	3	4	3
b	1	0	1	2	2	3	2
c	2	1	0	1	1	2	1
d	3	2	1	0	1	2	2
e	3	2	1	1	0	1	1
f	4	3	2	2	1	0	2
g	3	2	1	2	1	2	0

Fig. 7. Distance matrix of graph in Fig. 6

Eccentricity. The *eccentricity e(v)* of a point *v* in a connected graph G(V,E) is max d(u,v), for all $u \in V$. In other words, a point's eccentricity is equal to the distance from itself to the point farthest away. The eccentricity of node *b* in Figure 6 is 3.

Radius & Diameter. . The minimum eccentricity of all points in a graph is called the *radius* r(G) of the graph, while the maximum eccentricity is the *diameter* of the graph. In Figure 6, the radius is 2 and the diameter is 4.

Center. A vertex that is least distant from all other vertices (in the sense that its eccentricity equals the radius of the graph) is a member of the *center* of the graph and is called a *central point*. Every tree has a center consisting of either one point or two adjacent points.

Degree. The number of vertices adjacent to a given vertex is called the *degree* of the vertex and is denoted d(v). It can be obtained from the adjacency matrix of a graph by simply computing each row sum. For example, the degree of vertex *c* in Figure 6 is 4.

Average degree. The average degree, $\bar{d}$, of all vertices depicted in Figure 6 is 2.29. There is a direct relationship between the average degree, $\bar{d}$, of all vertices in a graph and the graph's density:

$$density = \frac{\bar{d}}{n-1}$$

Isolates & pendants. A vertex with degree 0 is known as an *isolate* (and constitutes a component of size 1), while a vertex with degree 1 is a *pendant*.

Degree variance. Holding average degree constant, there is a tendency for graphs that contain some nodes of high degree (and therefore high variance in degree) to have shorter distances than graphs with lower variance, with the high degree nodes serving as "shortcuts" across the network.

II. DIRECTED GRAPHS

Definition. A digraph *D(V,E)* consists of a set of nodes *V* and a set of ordered pairs of nodes *E* called *arcs* or directed lines. The arc *(u,v)* points from *u* to *v*.

Digraphs are usually represented visually like graphs, except that arrowheads are placed on lines to indicate direction (see Figure 5). When both arcs *(u,v)* and *(v,u)* are present in a digraph, they may be represented by a double-headed arrow (as in Figure 8a), or two separate arrows (as shown in Figure 8b).

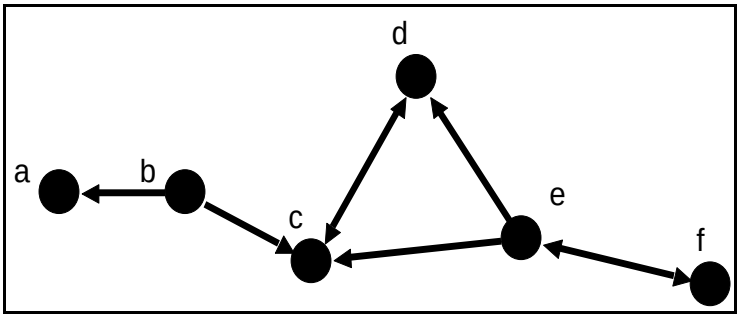


Fig. 8(a)

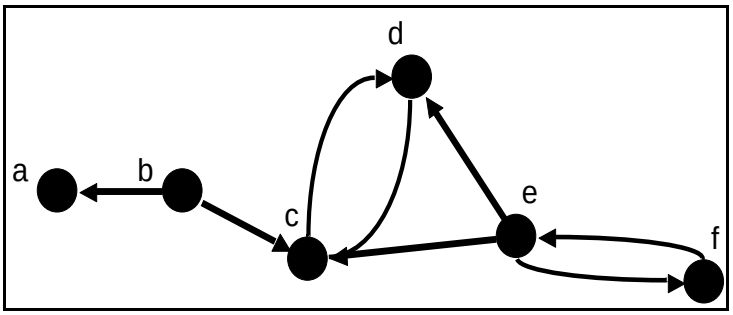


Fig. 8(b)

Directed Walk: In a digraph, a *walk* is a sequence of nodes $v_o,v_1,\ldots v_n$ in which each pair of nodes v_i, v_i+1 is linked by an arc (v_i,v_i+1). In other words, it is a traversal of the graph in which the flow of movement follows the direction of the arcs, like a car moving from place to place via one-way streets. A *path* in a digraph is a walk in which all points are distinct.

Semiwalk: A *semiwalk* is a sequence of nodes $v_o,v_1,\ldots v_n$ in which each pair of nodes v_i, v_i+1 is linked by either the arc (v_i,v_i+1) or the arc (v_i+1,v_i). In other words, in a semiwalk, the traversal need not respect the direction of arcs, like a car that freely goes the wrong way on one-way streets. By analogy, we can also define a *semipath*, *semitrail*, and *semicycle*.

Underlying graph: Another way to think of semi-walks is as walks on the *underlying graph*, where the underlying graph is the graph *G(V,E)* that is formed from the digraph *D(V,E')* such that $(u,v) \in E$ if and only if $(u,v) \in E'$ or $(v,u) \in E'$. Thus, the underlying graph of a digraph is basically the graph formed by ignoring directionality.

Strongly connected: A digraph is *strongly connected* if there exists a path (not a semipath) from every point to every other. Note that the path from *u* to *v* need not involve the same intermediaries as the path from *v* to *u*.

Unilaterally connected: A digraph is unilaterally connected if for every pair of points there is a path from one to the other (but not necessarily the other way around).

Weakly connected: A digraph is *weakly connected* if every pair of points is mutually reachable via a semipath (i.e., if the underlying graph is connected).

Strong component: A *strong component* of a digraph is a maximal strongly connected subgraph. In other words, it is a subgraph that is strongly connected and which is as large as possible (there is no node outside the subgraph that is strongly connected to all the nodes in the subgraph). A *weak component* is a maximal weakly connected subgraph.

Outdegree: The number of arcs originating from a node *v* (i.e., outgoing arcs) is called the *outdegree* of *v*, denoted *od(v)*.

Indegree: The number of arcs pointing to a node *v* (i.e., incoming arcs) is called the *indegree* of *v*, denoted *id(v)*. In a graph representing friendship feelings among a set of persons, *outdegree* can be seen as indicating gregariousness, while *indegree* corresponds to popularity. The average *outdegree* of a digraph is necessarily equal to the average *indegree*.

Directed adjacency: The *adjacency matrix* A of a digraph is an $n \times n$ matrix in which $a_{ij} = 1$ if $(v_i,v_j) \in E$ and $a_{ij} = 0$ otherwise. Unlike the adjacency matrix of an undirected graph, the adjacency matrix of a directed graph is not constrained to be symmetric, so that the top right half need not equal the bottom left half (i.e., $a_{ij} <> a_{ji}$). If a digraph is acyclic, then it is possible to order the points of D so that the adjacency matrix upper triangular (i.e., all positive entries are

above the main diagonal).

III. ATTRIBUTES FOR CLASSIFICATION OF GRAPHS

Brief description of graph attributes contained in feature vector for classification:

1. ***Average Degree:*** It is defined as the average value of degree of all nodes in the graph i.e $d(G)=\sum n_i d(u_i)/n$, where $d(u_i)$ denotes the degree of node u_i.

2. ***Average clustering coefficient:*** It is actually the ratio of the number of actual edges between neighbors of u to the number of possible edges between them, represented as $c(u)=\lambda(u)/\tau(u)$, where $\lambda(u)$ is the number of triangles of a node u and $\tau(u)=(d(u)^2-d(u))/2$;the number of triples a node u has. The clustering coefficient C(G) of a graph is the average clustering coefficient of nodes in the graph represented as $C(G)= 1/n(\sum n_i=1c(u_i))$

3. ***Average effective eccentricity:*** The eccentricity for a node assume for node u is defined by maximum distance of the shortest path from u to v denoted as $e(u)=\max\{d(u,v): v\in V\}$,where d(u,v) is the length of the shortest path from u to v .For effectiveness we consider the maximum length of the shortest path from node u. With the average of effective eccentricity over all nodes in the graph we get the effective eccentricity.

4. **Maximum effective eccentricity**: It provides maximum value of eccentricity over all nodes in the graph. It is actually the diameter of the graph.

   $diam(G) =\max\{e(u)|u\in V\}=\max\{d(u,v)|u,v\in V\}$.

   For maximum effective eccentricity it gives effective diameter

5. ***Minimum effective eccentricity:*** It provides minimum value of eccentricity over all nodes in the graph. It is actually the radius of the graph, So $rad(G)=\min\{e(u)|u\in V\}=\min\{d(u,v)|u,v\in V\}$.Foe maximum effective eccentricity it gives effective radius.

6. ***Average path length (Closeness centrality):*** the closeness centrality is defined by the reciprocal of the averaged total path length between node u and every other node that is reachable from node u, where u∈V. With the calculation of average of closeness centrality of all nodes considered as a global feature for a graph, i.e. $close(u)=(n-1)/\sum v\in V, v\neq d(u,v)$.

7. ***Percentage of total central point:*** It is computed by the ratio of total number of central points to the total number of points in the graph where each cenral point with respect to a node can be found where eccentricity of that node equal to effective radius of the graph means $effective\_rad(G)=e(u)$ where u∈V

8. ***Percentage of end points:*** If the end node is denoted as degree of one in graph then this type feature is calculated as a ratio of the number of end points to the total number of nodes in the entire graph.

9. ***Number of nodes:*** Simply it evaluates the total number of nodes the entire graph.

10. ***Number of edges:*** It counts the total number of edges.

11. ***Spectral radius:*** It is computed by the largest magnitude Eigen value of the adjacency matrix of the graph.if$|\lambda1|>|\lambda2|>\ldots.>|\lambda n|$ where λ1, λ2, λ3… λn are distinct Eigen values of the adjacency matrix A of the graph, and sorted by their magnitude. if ρ(G) represents the magnitude then $\rho(G)=|\lambda_1|$

12. ***Percentage of isolated points:*** If the isolated point of a node is denoted as degree of one in graph then this type feature is calculated as a ratio of the number of isolated points to the total number of nodes in the entire graph.

13. ***Second largest Eigenvalue:*** It is calculated by taking second largest Eigen value of the adjacency matrix A, i.e.,$|\lambda_2|$.

14. ***Trace:*** This feature is obtained by calculating the sum of all eigenvalue from a adjacency matrix A with respect to a graph, So that $Tr(A)= \sum n_i=1|\lambda_i|$.It is helpful for a graph having so many loops. The loop free graph has trace that is equal to 0.

15. ***Energy:*** It is evaluated by sum of all square of Eigen values of the adjacency matrix A with respect to a graph G. So that it can be expressed as $E(G)= \sum ni=1|\lambda i|^2$

16. ***Number of eigenvalue:*** If there are n Eigen values in an adjacency matrix A with respect to a graph G and among them s are distinct Eigen values. Every eigen values will be distinct is not always true.

17. ***Label entropy:*** If in a graph G has n different labels then label entropy can be measured as sum of the product of probability of a particular label and logarithm of probability of that label, if graph G has n different labels l1…..ln, then the label entropy is represented as $H(G)=-\sum n_i=_1 p(l_i)\log(l_i)$.

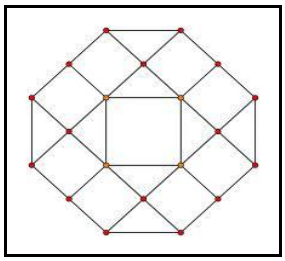
18. ***Neighborhood Impurity:*** Suppose L(u) is the label of a graph G and N(u) is the neighborhood of node u or N(u) contains the nodes which is adjacent to u, then degree of impurity will be zero when every node in neighborhood of u has same node label. So that, Impurity $Deg(u)=|L(v):v\in N(u),L(u)\neq L(v)|$

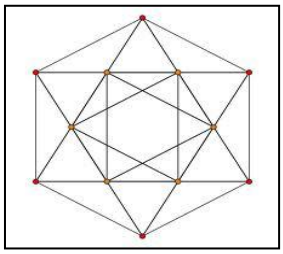
19. ***Link Impurity:*** When $L(u) \neq L(v)$ then edge connecting node u and node v or edge e(u,v) is said to be impure, For the entire graph G the impurity can be measured by $\{|(u,v) \in E : L(u) \neq L(v)|\}/m$, where m is the number of total edges in graph G.

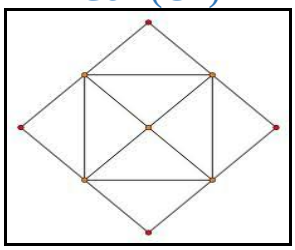
IV. GRAPH IMAGES FOR TEST

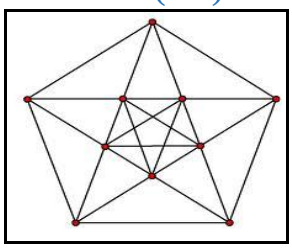
Images of the thirty one graphs (Protein and Cell graph) are given below.

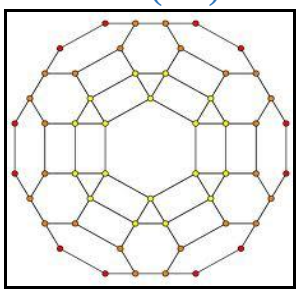
Cello Graph Images:

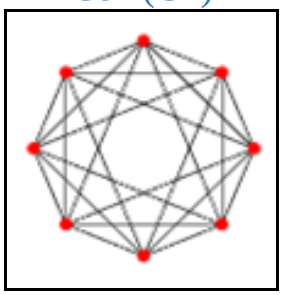

Cell (G1)

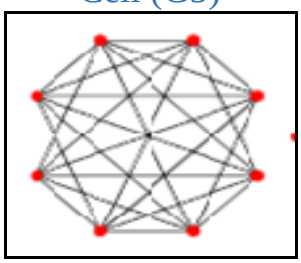

Cell (G2)

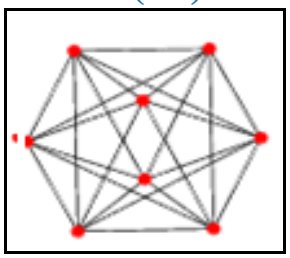

Cell (G3)

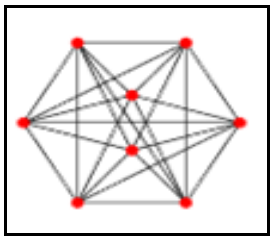

Cell (G4)

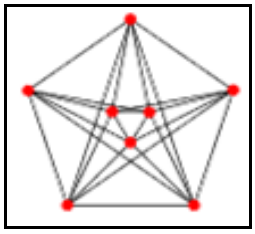

Cell (G5)

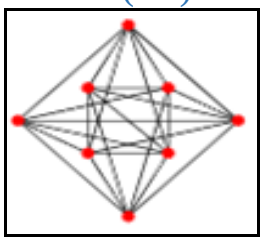

Cell (G6)

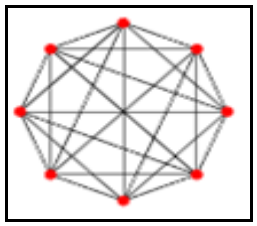

Cell (G7)

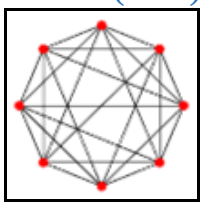

Cell (G8)

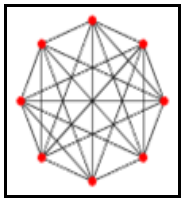

Cell (G9)

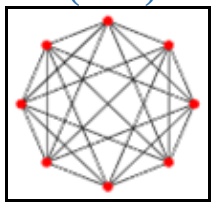

Cell (G10)


Cell (G11)

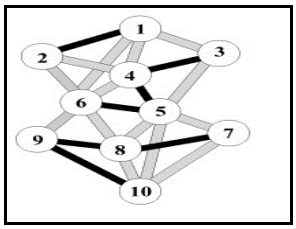

Cell (G12)


Cell (G13)

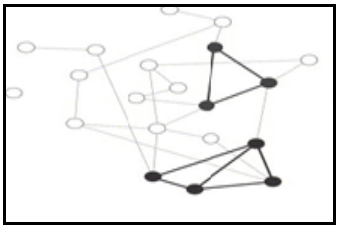

Cell (G14)

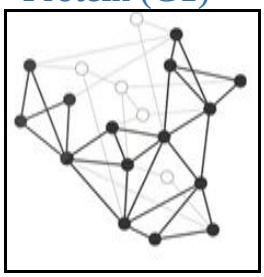

Cell (G15)

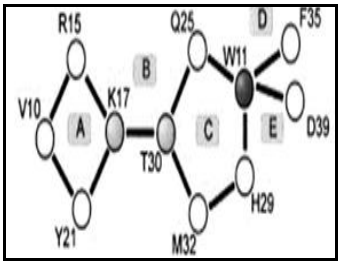
Protein graph images:



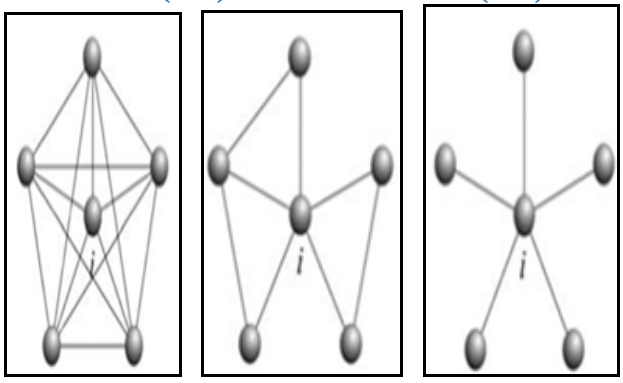
Protein (G1)


Protein (G2)

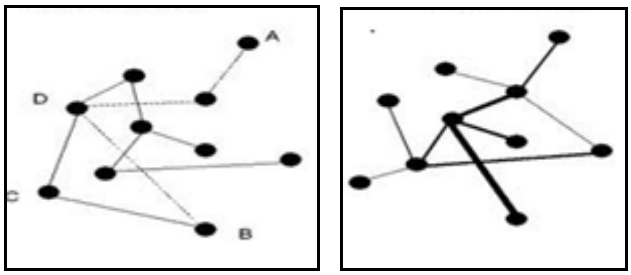

Protein (G3)



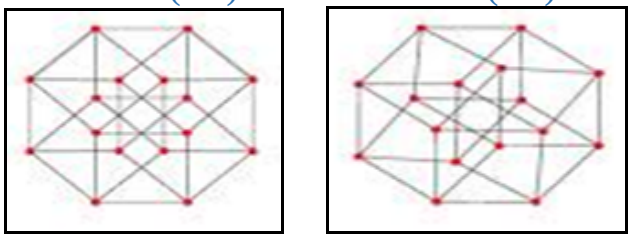
Protein (G4)



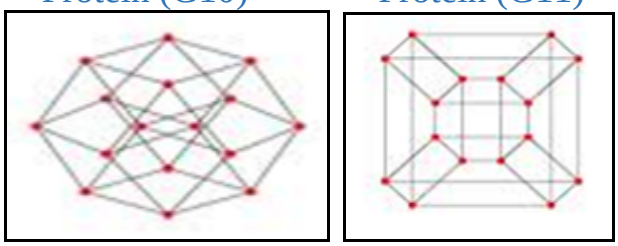
Protein (G5) Protein (G6) Protein (G7)



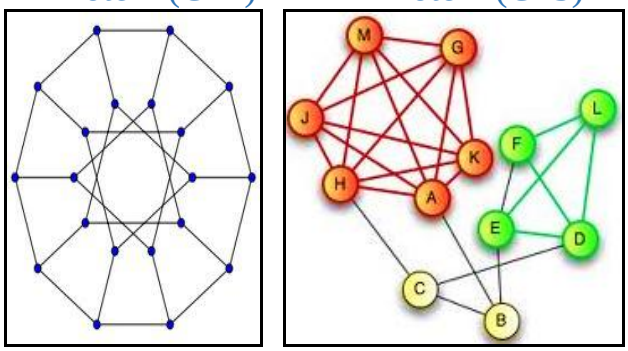
Protein (G8) Protein (G9)



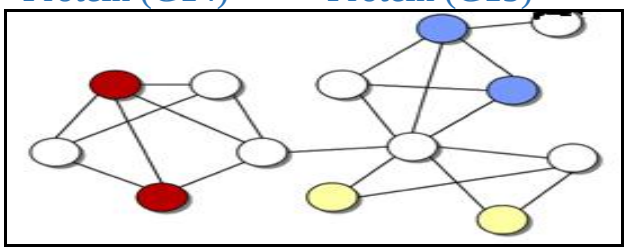
Protein (G10) Protein (G11)



Protein (G12) Protein (G13)



Protein (G14) Protein (G15)



Protein (G16)

V. FEATURED RESULT EXTRACTION OF GRAPHS BASED ON SOME CHARACTERISTICS:

Table 1 consumes 10 features and Table 2 consumes 9 features of the above mentioned 31 graphs i.e. 15 cello graphs and 16 protein graphs.

Graph	Average degree	Average Clustering Coefficient	Average Effective Eccentricity	Max. Effective Eccentricity	Min. Effective Eccentricity	Closeness centrality	% of Central points	% of isolated points	% of end points	Number of nodes
Cell (G1)	3.5	0.206667	4.3	5	3	0.3946	15	0	0	20
Cell (G2)	5	0.483333	2.5	3	2	0.6332	50	0	0	12
Cell (G3)	3.55556	0.696296	2.4444	3	2	0.6343	55.556	0	0	9
Cell (G4)	5	0.55	2	2	2	0.6964	100	0	0	10
Cell (G5)	3.73333	0.4344	5.0667	6	4	0.3468	26.667	0	0	30
Cell (G6)	6	0.8	2	2	2	0.875	100	0	0	8
Cell (G7)	3.42857	0	4	4	4	0.4643	100	0	0	9
Cell (G8)	9	0.554762	2.5	3	2	0.5999	50	0	0	9
Cell (G9)	6	0.8	2	2	2	0.875	100	0	0	8
Cell (G10)	6.58333	0.379861	2.7083	3	2	0.5259	29.167	0	0	8
Cell (G11)	6	0.8	2	2	2	0.875	100	0	0	8
Cell (G12)	3.07407	0.055556	9.2407	11	7	0.2064	11.11	0	0	9
Cell (G13)	6	0.8	2	2	2	0.875	100	0	0	8
Protein (G1)	4.2	0.6667	2.6	3	2	0.6193	40	0	0	10

Protein (G2)	2.7	0.1	4.75	7	0	0.3775	5	5	10	20
Protein (G3)	3.8	0.289524	3.85	5	3	0.4251	30	0	0	20
Protein (G4)	2.1818	0	4.8182	6	3	0.3773	9.091	0	18.18	11
Protein (G5)	5	1	1	1	1	1	100	0	0	6
Protein (G6)	2.67	0.8277	1.8333	2	1	0.7024	16.67	0	0	6
Protein (G7)	1.67	0	1.8333	2	1	0.6296	16.67	0	83	6
Protein (G8)	2	0.2166	4.8	6	3	0.3759	10	0	30	10
Protein (G9)	2	0	3.3	4	2	0.4586	10	0	60	10
Protein (G10)	3.875	0	4	4	4	0.4653	100	0	0	16
Protein (G11)	4	0	4	4	4	0.4688	100	0	0	16
Protein (G12)	3.875	0	4	4	4	0.4653	100	0	0	16
Protein (G13)	3	0	5	5	5	0.38	100	0	0	20
Protein (G14)	4.33	0.694	3.3333	4	2	0.4953	16.667	0	0	12
Protein (G15)	3.23	0.6465	4	5	3	0.4505	15.385	0	7.692	13

Table 1

Graph	Number of edges	Spectral radius	Second largest Eigenvalue	Trace	Energy	Number of Eigenvalue	Label Entropy	Neighbourhood Impurity	Link Impurity
Cell (G1)	35	3.8607	2.8159	32	70	20	5.1293	2	0.54286
Cell (G2)	30	5.2361	2.618	21	60	11	4.9069	2	0.4
Cell (G3)	16	4	2	14	32	8	4.25	2.6667	0.75
Cell (G4)	25	5.2361	2	18	50	9	4.6439	2	0.4
Cell (G5)	58	4.4581	3.4199	52	118	30	5.8074	2.6667	0.7143
Cell (G6)	24	6	2	12	48	8	4.585	0	0
Cell (G7)	24	3.4641	2	19	48	13	4.585	3.4286	1
Cell (G8)	20	9.1231	5.3223	56	216	23	6.7682	4	0.4444
Cell (G9)	24	6	2	12	48	8	4.585	0	0
Cell (G10)	20	6.7538	4.2345	50	158	24	6.3038	3.25	0.4937
Cell (G11)	24	6	2	12	48	8	4.396	0	0
Cell (G12)	36	3.6533	3.1789	82	166	54	6.375	0.9259	0.3012
Cell (G13)	24	6	2	12	48	8	4.585	0	0
Protein (G1)	21	4.4751	2.6747	16	42	10	4.3923	4	0.9524
Protein (G2)	27	3.3932	2.6887	28	54	20	4.7548	2.1	0.7778
Protein (G3)	38	4.2275	3.3864	33	76	20	5.2479	3.4	0.8947
Protein (G4)	12	2.4035	2.2739	13	24	11	3.5849	1.4545	0.6667
Protein (G5)	15	5	1	10	30	6	3.9068	0	0
Protein (G6)	8	2.9474	1.8208	8.2	16	6	3	2.3333	0.875
Protein (G7)	5	2.2361	2.2361	4.5	10	4	2.3219	1.6667	1

Protein (G8)	**10**	**2.4401**	**2.1171**	**12**	**20**	**10**	**3.3219**	**1.8**	**0.9**
Protein (G9)	**10**	**2.6131**	**1.4142**	**10**	**20**	**10**	**3.3219**	**1.6**	**0.8**
Protein (G10)	**31**	**3.9049**	**2.1742**	**24**	**62**	**15**	**4.9542**	**0.75**	**0.1935**
Protein (G11)	**32**	**4**	**2**	**24**	**64**	**12**	**5**	**0**	**0**
Protein (G12)	**31**	**3.9049**	**2.1742**	**24**	**62**	**16**	**4.9542**	**0.75**	**0.1935**
Protein (G13)	**30**	**3**	**2**	**32**	**60**	**16**	**4.9069**	**0**	**0**
Protein (G14)	**26**	**5.0911**	**3.233**	**20**	**52**	**11**	**4.7**	**2.667**	**0.6154**
Protein (G15)	**21**	**3.7117**	**3.1325**	**18**	**42**	**13**	**4.3923**	**2.6154**	**0.8095**

Table 2

VI. FUTURE SCOPE

By applying the concept of this characteristics of different graphs I am planning to design a relevant feature selection algorithm based on neural network

BIOGRAPHY

Saptarshi Bhattacharyya is a Research Assistant in the Department Of Computer Science, at University Of Calcutta. He has immense interest in research, and has contributed considerably in research endeavors. His major research interest areas are Biomedical Image Processing, Data Structure, Visual Cryptography, Data Hiding, Image Processing, Green Computing, various computer languages etc.

REFERENCES

[1] Kernels for Graph Classification by Hisashi Kashima and Akihiro Inokuchi

[2] IBM Research, Tokyo Research Laboratory 1623-14, Shimotsuruma, Yamato-shi,Kanagawa 242-8502, Japan fhkashima, Inokouchig@jp.ibm.com

[3] Saptarshi Bhattacharyya "Complexity Analysis of a Lossless Data Compression Algorithm using Fibonacci Sequence" International Journal of Information Technology (IJIT) – Volume 3 Issue 3, May - Jun 2017

[4] Journal-K.M Borgwardt and H.P Krigel Shortest path kernels on graphs. In 5th IEEE International Conference on data mining,pages 74-81,Wastington,DC,USA,2005.

[5] Managing And Mining Graph Data Edited by CHARU C. AGGARWAL , IBM T. J. Watson Research Center, Yorktown Heights