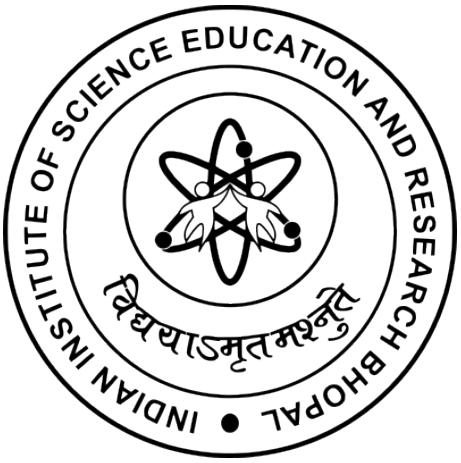
APPLICATIONS OF SPECTRAL GRAPH THEORY IN PROBING QUANTUM ENTANGLEMENT

PRIYANSHU KUMAR RAI



BS Project Endterm Evaluation

BS (Electrical Engineering and Computer Science)
Department of Electrical Engineering and Computer Science

submitted to
Indian Institute of Science Education and Research Bhopal

Supervisors

Dr. Ankur Raina
EECS Dept., IISER Bhopal

Dr. Sujoy Bhore
CSE Dept., IIT Bombay

April 2024

The graph with no vertices and
no edges is the ***null*** *graph,*
regarded by some authors as a
pointless concept.

— Chris Godsil, Gordon Royle [1]

ACKNOWLEDGMENTS

I would like to express my deepest appreciation to my advisors Dr. Ankur Raina and Dr. Sujoy Bhore for their constant feedback and guidance throughout the course of my project. This project would not have been possible without the expertise shared and the encouragement given by them.

I am also grateful to my labmates and batchmates who were always a source of new information and learning for me. Thanks should also go to the department for providing us with a competitive environment to grow and learn.

I'd like to recognize my friends who kept me in high spirits and motivated me at all times. I would be remiss in not mentioning my parents and brother for their encouragement and the trust they kept in me.

CONTENTS

LIST OF FIGURES

1 INTRODUCTION

1.1 WHAT IS SPECTRAL GRAPH THEORY?

Applications of spectral graph theory in probing quantum entanglement

Spectral graph theory is the study of the properties of a graph in relationship to the characteristic polynomial, eigenvalues, and eigenvectors of matrices associated with the graph, such as its adjacency matrix or Laplacian matrix. [10]

1.2 SOME IMPORTANT TERMINOLOGIES

1.2.1 *Definition of a Graph*

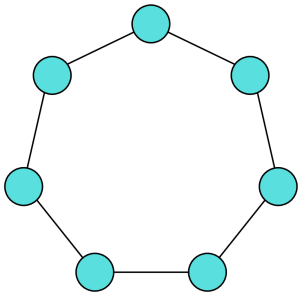


Figure 1.1: A two-regular graph with seven vertices.

A simple graph G is an ordered pair (V, E), consisting of a nonempty set V of vertices and a set E of edges, each edge a two-tuple of V with no edge having identical ends.

1.2.2 *Matrix Representation of a Graph*

Adjacency Matrix is a square matrix used to represent a graph. The adjacency matrix A has elements,

$$A_{ij} = \begin{cases} 1 & (i,j) \in E \\ 0 & o.w. \end{cases} \tag{1.1}$$

In case of no self-loops as mentioned above, in the case of a simple graph, $A_{ii} = 0 \; \forall \; i \in V$.

For the figure 1.1, the adjacency matrix can be represented as,

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

1.2.3 *Degree of a node*

The **degree** of a node in an undirected network is the number of edges connected to it. We denote the degree of node $i \in V$ by k_i.

We sometimes use D to represent the **degree matrix** which is a diagonal matrix with the values k_i where $i = j$.

1.2.4 *What makes a graph connected?*

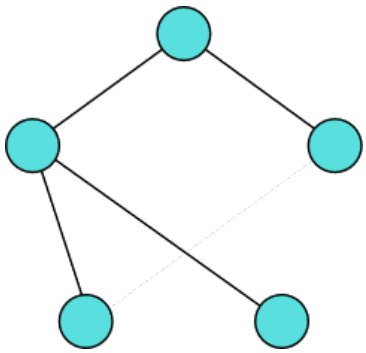


Figure 1.2: A simple tree with 5 nodes.

A connected graph G is one in which there exists a path between vertices a and b, $\forall a, b \in V$.

A **tree** T is a poorly connected graph as removing any one edge will render it disconnected. Figure 1.2 shows an example of such a tree.

A graph is **complete**, if every pairs of vertices are joined together by an edge, for n vertices, a complete graph is represented by K_n.

The **Graph Laplacian** for a simple undirected, unweighted network is an $n \times n$ symmetric matrix L with elements,

$$L_{ij} = \begin{cases} k_i, & i = j \\ -1, & i \neq j \\ 0, & \text{o.w.} \end{cases} \tag{1.2}$$

$$L_{ij} = k_i \delta_{ij} - A_{ij}$$

where A_{ij} is an element of the adjacency matrix and δ_{ij} is the Kronecker delta, which is 1 if $i = j$ and 0 otherwise.

Another way to find the laplacian matrix is,

$$L = D - A$$

where D is the diagonal matrix with the node degrees along its diagonal.

1.3 CONNECTING CLASSICAL GRAPHS WITH QUANTUM GRAPH STATES

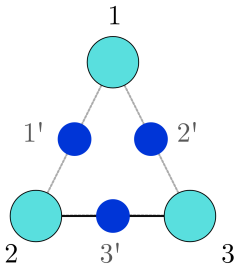
In this project we aim to find a classical metric using spectral graph theory and relate it with its quantum counterpart in order to find a good relation through which if we know one metric, we can guess the other with good certainty.

We will explore the metrics which we have come up with for now and also discuss the future metrics which can be valid for comparision.

2 EXPERIMENTS

2.1 ENTANGLEMENT VALUE

For every corresponding edge pair, we fill $log(n)$ if separating the system along the midpoints of the respective edges leads to two separated components unless when the edge is being compared to itself in which case we see if the edge is actually present in the graph state and thus assign it a value of $log(n)$ and the rest of the edge pairs are assigned a value of zero. We then sum all the elements on the primary diagonal and either of the upper or lower triangular parts. Here, the base of the logarithm is kept as two as was used originally by Von Neumann.



	1'	2'	3'
1'	0	1	0
2'	1	1	1
3'	0	1	0

Figure 2.1: (a) Graph representation and (b) its corresponding entanglement matrix representation.

The base of the logarithm, in this case, is base 2, and $\mathcal{S}$ is referred to as Von Neumann Entropy, which is,

$$\mathcal{S}(\rho A) = -\mathrm{Tr}[\rho_A \log_2 \rho_A] = -\mathrm{Tr}[\rho_B \log_2 \rho_B] = -\sum_i p_i \log p_i \quad (2.1)$$

2.2 PROPERTIES OF THE GRAPH LAPLACIAN

We consider only connected graph which means the number of connected components is one. Thus the number of zero eigenvalues of L is also one. [5]

As the laplacian is a PSD, the smallest eigenvalue is zero.

The second smallest eigenvalue $\alpha(G)$ is also known as the algebraic connectivity of a graph and is greater than 0 in our case. [6]

For a given value of n which is the number of vertices for the graphs generated, we find the laplacian and subsequently $\alpha(G)$. Our aim is

then to compare the entanglement values obtained with the second smallest eigenvalue of the respective graphs.

2.3 INVERSE DISTANCE DETERMINANT VALUE

We denote the shortest path between two nodes i and j with s_{ij} and store the result in a matrix S. We define a matrix T where,

$$T_{ij} = \begin{cases} \frac{1}{s_{ij}}, & \text{if } s_{ij} \neq 0 \\ 0, & \text{o.w.} \end{cases} \tag{2.2}$$

We record our observations with values of $n = 4, 5, 6, \ldots$. The idea here is that the closer every possible pair in a graph is, the more connected and therefore, the more entangled the graph might be.

3

EXPANDER GRAPHS

3.1 DEFINITION

An Expander Graph is a *sparsely populated graph* that is *well connected*.

- A *sparse graph* is a graph in which the total number of edges is few compared to the maximal number of edges.
- A graph G is connected if there exists a path between vertices a and b $\forall$ $a, b \in G$.

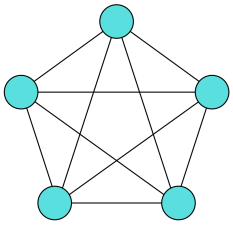


Figure 3.1: A k-regular graph (here $k = 4$) is a simple and good example of an expander graph.

3.2 CHEEGER CONSTANT

The **Cheeger constant** or the **edge expansion** of a finite graph G, [8]

$$c(G) = \min_{0<|S|\leq \frac{n}{2}} \frac{|\partial(S)|}{|S|} \tag{3.1}$$

where the boundary of S is δS. In simple terms, we have to find out all the possible subsets of V of sizes ranging from 1 to $n/2$ and calculate $\frac{|\partial(S)|}{|S|}$ for every such set. We finally keep the minimum value among those which becomes the cheeger constant for the graph G.

The larger the Cheeger constant, the better connected the graph is. $c(G) > 0$ iff G is a connected graph.

3.3 VERTEX EXPANSION

The vertex expansion or the vertex isoperimetric number $h_{\text{out}}(G)$ of a graph G is defined as,

$$h_{\text{out}}(G) = \min_{0\leq|S|\leq \frac{n}{2}} \frac{|\partial_{\text{out}}(S)|}{|S|}$$

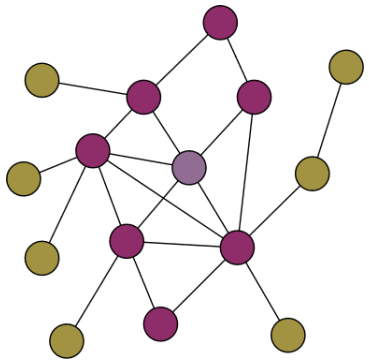


Figure 3.2: The purple shaded nodes are part of S. The dark purple nodes are part of $\partial_{in}(S)$ while the yellowish nodes are part of $\partial_{out}(S)$

Similarly, $h_{in}(G)$ is defined as,

$$h_{in}(G) = \min_{0 \leq |S| \leq \frac{n}{2}} \frac{|\partial_{in}(S)|}{|S|}$$

3.4 SPECTRAL EXPANSION

The spectral gap of G, $s(G)$ is defined as $s(G) = \lambda_1 - \lambda_2$, where λ_1 is the largest eigenvalue and λ_2 is the second largest eigenvalue of the adjacency matrix of G. [2]

Cheeger's and Buser's inequalities

$$\frac{s(G)}{2} \leq c(G) \leq \sqrt{2\lambda_1 s(G)}$$

If the graph is bipartite, the eigenvalues are symmetric about zero.

3.5 RAMANUJAN GRAPHS

A Ramanujan graph is a construction of spectral expander which can be used in error correcting codes to create asymptotically good codes. [7]

$$\lambda_2 \geq 2\sqrt{d-1} - o(1)$$

Here λ_2 is the largest eigenvalue in its absolute value.

3.6 MIXING LEMMA

In expander graphs (for which $\lambda << d$) the fraction of edges connecting two large sets of vertices approximately equals the product of the densities of these sets. This property is called *mixing*. Here density of a set S is defined as

$$\rho(S) := \frac{|S|}{N}$$

3.7 EXPANDER RANDOM WALK LEMMA

Let $G = (N, E)$ be a d-regular graph, and consider walks on G that start from a uniformly chosen vertex and take $l - 1$ additional random steps, where in each such step we uniformly select one out of the d edges incident at the current vertex and traverse it.

Let W be a subset of $[N]$ and $\rho = \frac{|W|}{N}$. Then the probability that such a random walk stays in W is at most,

$$\rho \cdot \left(\rho + (1 - \rho) \cdot \frac{\lambda(G)}{d} \right)^{l-1}$$

where $\lambda(G)$ is known as the **eigenvalue bound**.

3.8 FAMILIES OF EXPANDERS

3.8.1 *The Iterated Zig-Zag Construction*

The construction of a large expander graph proceeds in iterations, where in the i-th iteration the current graph G_i and the fixed graph G are combined, resulting in a larger graph G_{i+1} while keeping the expansion property of G_{i+1} at least as good as the expansion of G_i, while G_{i+1} maintains the degree of G.

3.8.2 *Margulis-Gabber-Galil*

For every natural number n, graph G_n with the vertex set $\mathbb{Z}_n \times \mathbb{Z}_n$, where $\mathbb{Z}_n = \mathbb{Z}/n\mathbb{Z}$. For every vertex $(x, y) \in \mathbb{Z}_n \times \mathbb{Z}_n$, its eight adjacent vertices are $(x \pm 2y, y), (x \pm (2y + 1), y), (x, y \pm 2x), (x, y \pm (2x + 1))$.

It follows that for all n, the graph G_n has second-largest eigenvalue $\lambda(G) \leq 5\sqrt{2}$.

3.8.3 *k-regular Graph*

A finite k-regular graph is said to be a (one-sided) ϵ-expander if one has $\lambda_2 \leq (1 - \epsilon)k$ and a two-sided ϵ-expander if one also has $\lambda_n \geq -(1 - \epsilon)k$.

3.8.4 *Chordal Cycle Graph*

A *chordal graph* is one in which all cycles of four or more vertices have a chord, which is an edge that is not part of the cycle but connects two vertices of the cycle.

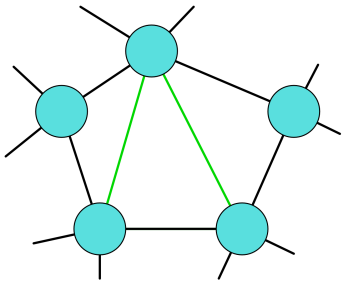


Figure 3.3: An example of a chordal graph.

For every node, mark a left node $(x-1)$, a right node $(x+1)$ and a chord node x^{p-2} and connect x with these three. Take modulus where necessary.

3.8.5 *Paley Graph*

P_q is $(q-l)/2$-regular, any two adjacent vertices have $(q-5)/4$ common neighbors and any two non-adjacent vertices have $(q-l)/4$ common neighbors. C is a Hadamard matrix. $B = A - I$.

$$A = \begin{bmatrix} 0 & + & - \\ - & 0 & + \\ + & - & 0 \end{bmatrix} \qquad B = \begin{bmatrix} - & + & - \\ - & - & + \\ + & - & - \end{bmatrix} \qquad C = \begin{bmatrix} + & + & + & + \\ + & - & + & - \\ + & - & - & + \\ + & + & - & - \end{bmatrix}$$

3.9 SOME INTERESTING PROPERTIES OF EXPANDERS

- The second largest eigenvalue, λ_2 is bound by $2\sqrt{d-1}$ for a d regular graph.
- If the graph is bipartite, the eigenvalues are symmetric about zero.
- $\lambda_1 = d$ so $s(G) = d - \lambda_2$.
- $\lambda_2(G) \leq \max_{G_1,G_2} \min\{\lambda_1(G_1), \lambda_2(G_1)\}$.

3.10 PEARSON CORRELATION COEFFICIENT VALUES FOR VARIOUS n AND METHODS USED

The correlation is obtained between the method used vs second largest eigenvalue.

n	Entanglement	Inverse Distance	Cheeger Constant
4	0.91874	0.94776	0.92042
5	-	0.90590	0.90685
6	-	0.55140	0.81209

4 ERROR CORRECTION

4.1 INTRODUCTION

- The message sent from Alice is not the same as that received by Bob.
- It is a mapping $f : \{0,1\}^k \to \{0,1\}^n$.
- The *original message* is encoded into *codewords*.
- The *minimum relative distance* δ should be large enough.
- The amount of information transmitted by the *rate* which is k/n.
- We'll deal in linear codes which are linear combinations over $GF(2)$.

4.2 EXPANDERS : ARE THEY GOOD ENOUGH?

- For a graph $G \in \mathcal{G}$ with n vertices, construct a code of length $dn/2$.
- Since C has minimum distance of at least δd, it is possible to correct $\delta d/2$ errors.

4.3 APPROACH FOR USING EXPANDER CODES FOR ERROR CORRECTION

For a d-regular graph $G(V,E) \in \mathcal{G}$ with $|V| = n$ and $|E| = nd/2$, we can construct a code of length $nd/2$ called $\mathcal{C}(G,C)$. The edges in G each together form a codeword in C. Due to n vertices, there are a total of $(1-r)dn$ constraints.

We know that n and k should be such that,

$$\begin{aligned} n+1 &\leq 2^{n-k} \\ n+1 &\leq \frac{2^n}{2^k} \\ 2^k(n+1) &\leq 2^n \\ 2^k &\leq \frac{2^n}{n+1} \\ k &\leq \log_2\left(\frac{2^n}{n+1}\right) \end{aligned}$$

Using the above inequality, we can approximate optimum value for k given n and d of the graph.

Example 1: 3-regular graph with 4 vertices. Here, code length will be $dn/2 = 3 \times 4/2 = 6$. Possible values of k are $1, 2,$ and 3. We take $k = 3$ and proceed with generating codes.

Example 2: 4-regular graph with 5 vertices. Here, code length will be $dn/2 = 4 \times 5/2 = 10$. Possible values of k are $1, 2, 3, 4, 5,$ and 6. We take $k = 6$ and proceed with generating codes.

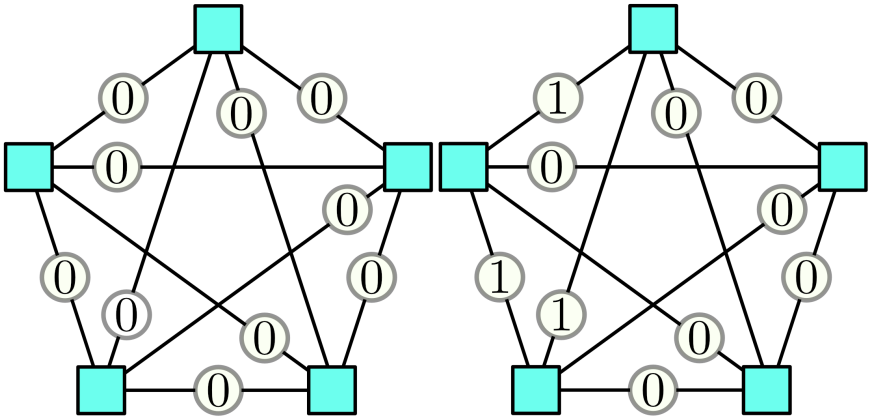


Figure 4.1: Two codewords which satisfy all the parity check constraints. The graph is 4-regular with 5 vertices.

In the above case, we have taken a 4-regular graph with 5 vertices. As observed, the *minimum code distance* is 3.

4.4 ALGORITHM FOR DECODING EXPANDER CODES

Linear time decoding is possible with Expander Graphs. [9]

Every edge induces codewords to both the vertices it has as its endpoints. The nodes of the graph become similar to parity check functions, which means that the sum of values at edges of a particular node is added and a *modulus* of 2 is applied. If the parity is zero, the bit is valid otherwise we perform adjustments. Flip the edges suggested by a vertex only within distance $\delta d/4$.

4.5 EDGE CASES

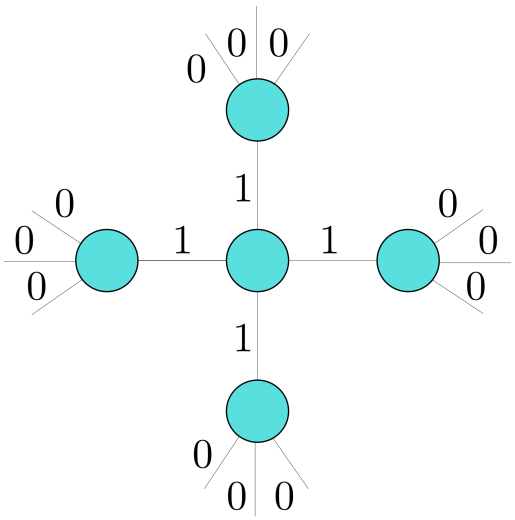


Figure 4.2: Edge case with each node having degree 4

In the above case, as observed, the edges about the central node all have values 1 however those at the neighboring edges contain majority as 0, thus the neghboring edges form codeword according to that vertex, but will appear as corrupt to the neighbors.

4.6 PARITY CHECK MATRICES FOR EXPANDER GRAPHS

For a graph with n vertices and k edges, we construct an $n \times k$ matrix where the entry corresponding to a certain row i and certain column j signifies whether the node i and edge j are connected. This is done by enemerating the edges in any order starting from 0 to $k-1$. After that for every node we fill the entries corresponding to each of the columns.

The Appendix displays all the graphs which satisfy the above constraint. The parity matrix for the graph with $n = 6$ and $si = 012$ is,

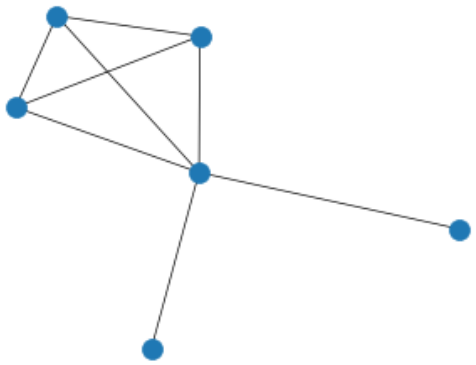


Figure 4.3: Graph with $n = 6$ and $si = 012$

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

5 OBSERVATIONS

5.1 ENTANGLEMENT VALUE VS SECOND SMALLEST EIGENVALUE

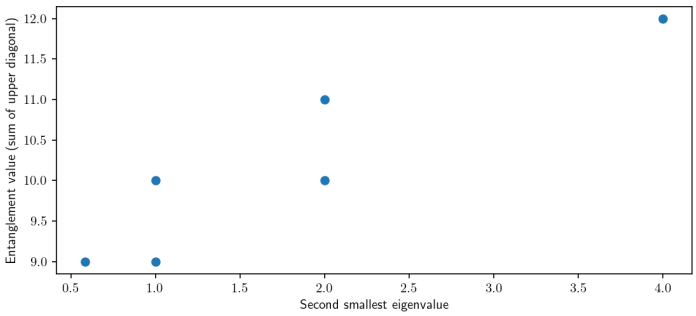


Figure 5.1: Entanglement value vs Second smallest eigenvalue for all the possible graphs with $n = 4$

For $n = 3$, there are only 2 graphs possible (2 and 3 edges) therefore no clear observation is possible. In case of $n = 5$, it is no longer possible to represent the graph on a plane in case of large number of connections hence, while seperating the system along the midpoints of the respective edges of interest, leads to a few points for which deciding whether the graph is actually getting divided into separate subsystems is difficult.

5.2 INVERSE DISTANCE DETERMINANT VALUE VS SECOND SMALLEST EIGENVALUE

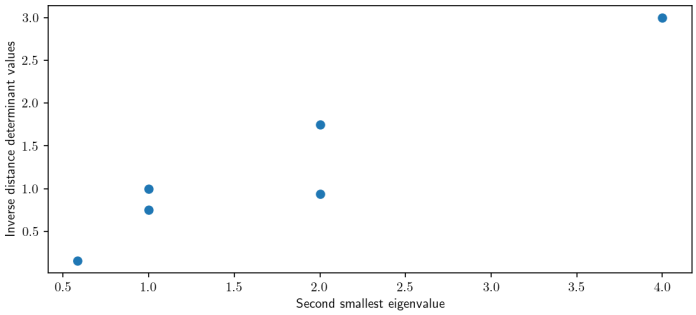


Figure 5.2: Inverse Distance Determinant Value vs Second smallest eigenvalue for graphs with $n = 4$

For $n = 3$, there are only 2 graphs possible (2 and 3 edges) therefore no clear observation is possible.

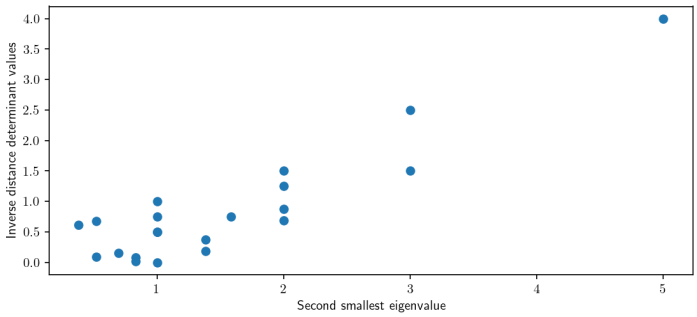


Figure 5.3: Inverse Distance Determinant Value vs Second smallest eigenvalue for graphs with $n = 5$

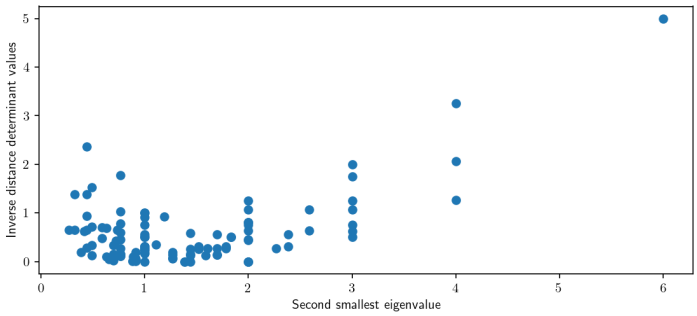


Figure 5.4: Inverse Distance Determinant Value vs Second smallest eigenvalue for graphs with $n = 6$

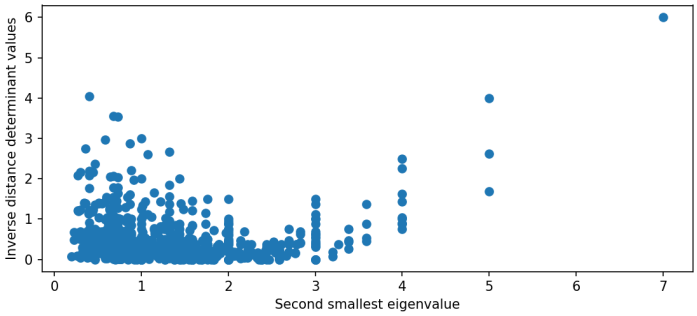


Figure 5.5: Inverse Distance Determinant Value vs Second smallest eigenvalue for graphs with $n = 7$

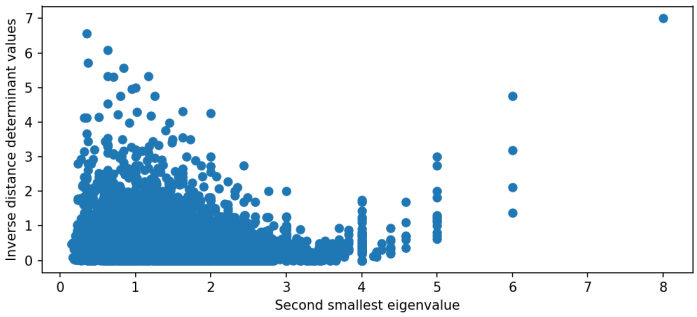


Figure 5.6: Inverse Distance Determinant Value vs Second smallest eigenvalue for graphs with $n = 8$

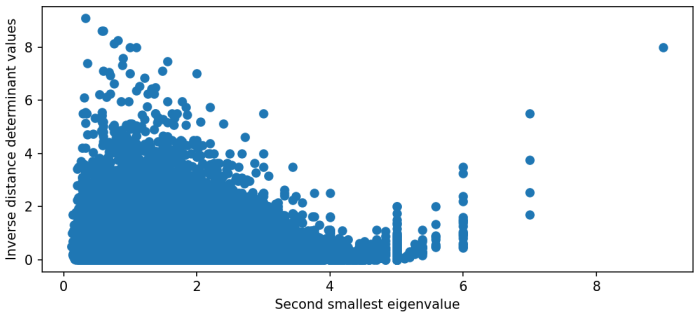


Figure 5.7: Inverse Distance Determinant Value vs Second smallest eigenvalue for graphs with $n = 9$

In the above figure 5.4, we notice a poor correlation between the two axes. Therefore, we need to find a better candidate to compare against the second smallest eigenvalue.

5.3 CHEEGER CONSTANT VALUE VS SECOND SMALLEST EIGENVALUE

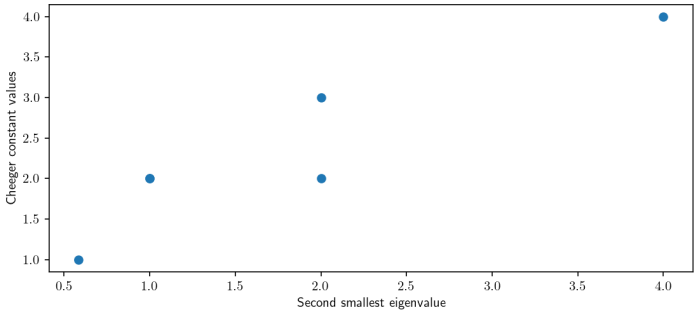


Figure 5.8: Cheeger constant value vs Second smallest eigenvalue for graphs with $n = 4$

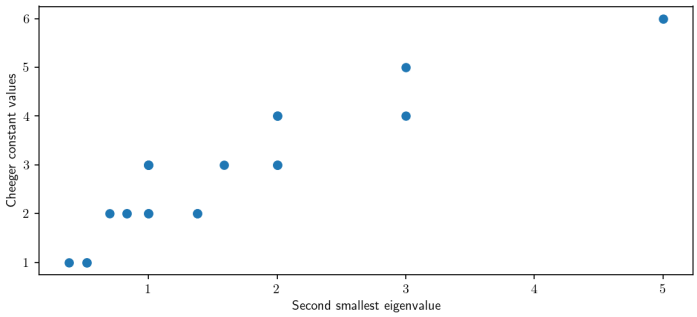


Figure 5.9: Cheeger constant value vs Second smallest eigenvalue for graphs with $n = 5$

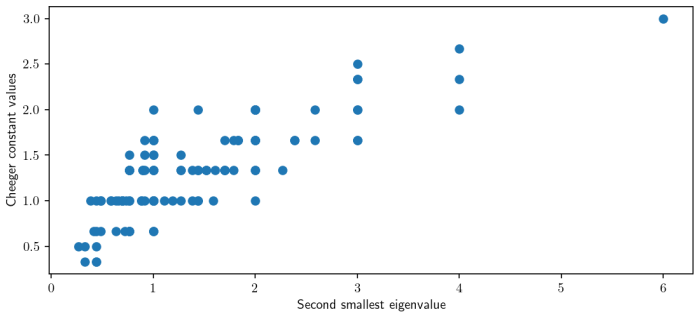


Figure 5.10: Cheeger constant value vs Second smallest eigenvalue for graphs with $n = 6$

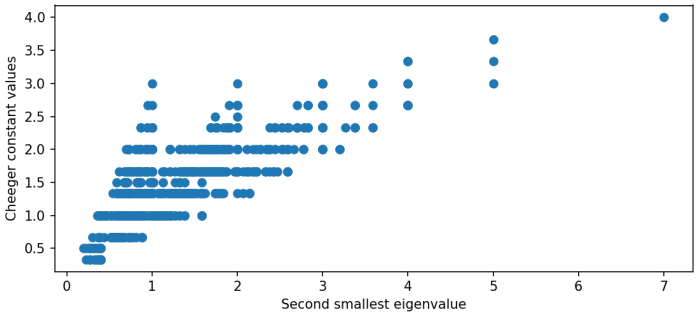


Figure 5.11: Cheeger constant value vs Second smallest eigenvalue for graphs with $n = 7$

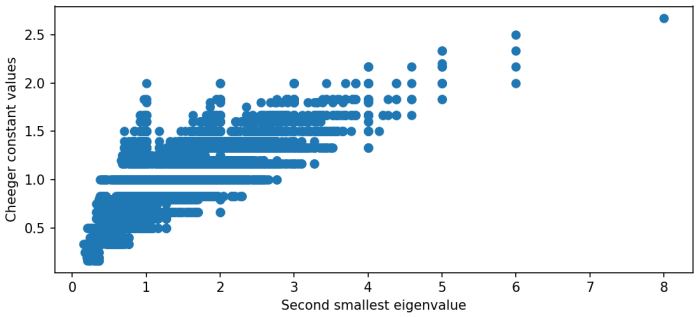


Figure 5.12: Cheeger constant value vs Second smallest eigenvalue for graphs with $n = 8$

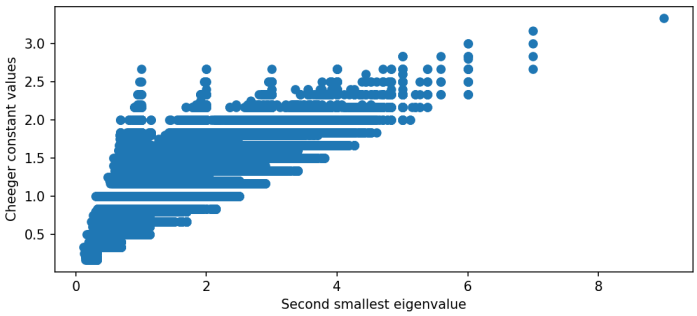


Figure 5.13: Cheeger constant value vs Second smallest eigenvalue for graphs with $n = 9$

5.4 DEFICIENCY OR NULLITY OF A MATRIX

We calculate the d value which is simply the minimum number of columns in a matrix which can perform row-wise modular addition to give a column with all zeros. The d value is to be calculated over the parity check matrices of a graph which satisfy the constraint that is $n < k$. The method for obtaining the parity check matrices has been explained in 4.6.

5.5 CORRELATION VALUES BETWEEN DIFFERENT PARAMTERS

Why correlation instead of covariance?

Covariance represents change while *correlation* represents connection. Coefficient of correlation lies between $(-1, 1)$ while covariance can range between $(-\infty, \infty)$. The correlation coefficient is calculated as follows:

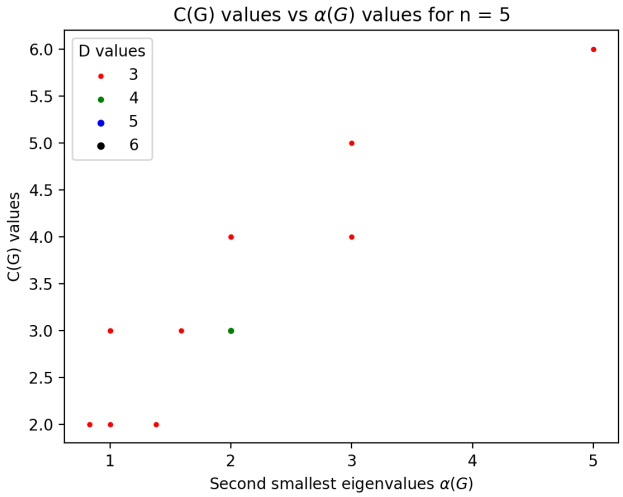
$$r = \frac{\sum(x - \overline{x})(y - \overline{y})}{\sqrt{\sum(x - \overline{x})^2}\sqrt{\sum(y - \overline{y})^2}}$$

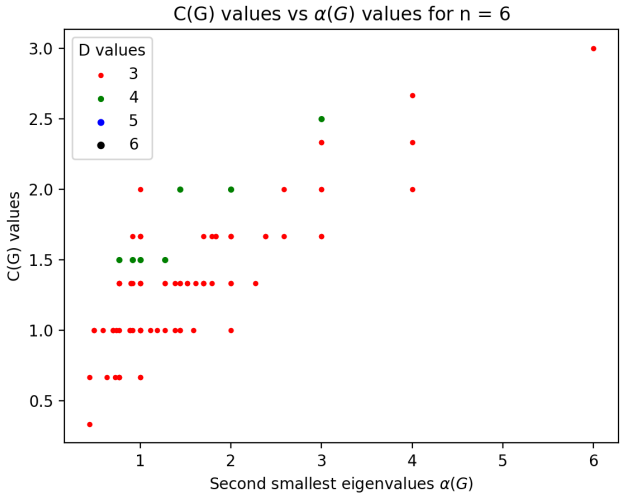
n	No. of filtered graphs	C(G) vs D vals	C(G) vs $\alpha(G)$	$\alpha(G)$ vs D vals
4	2	nan	1.00000	nan
5	13	0.75289	0.00003	0.98950
6	93	0.03303	0.00000	0.83271
7	809	0.87909	0.00000	0.00635
8	11005	0.00000	0.00000	0.00000

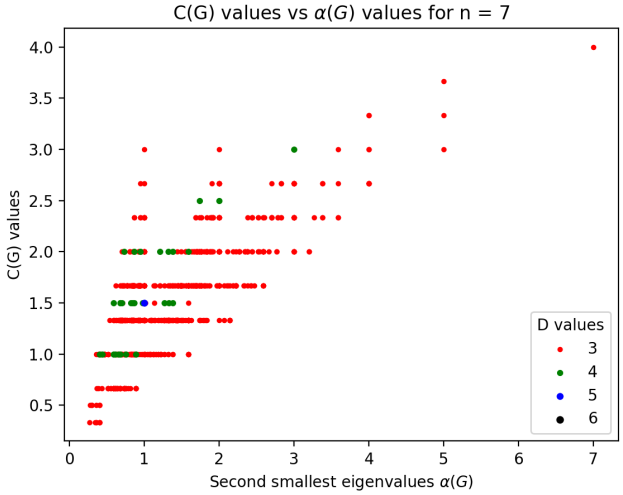
From the above table we can observe that the correlation value diminishes completely at $n = 8$ and further. There is no correlation found between C(G) and $\alpha(G)$. We also notice that in the column C(G) vs D vals, at $n = 6$, there is a sudden dip in the correlation value which increases further at $n = 7$.

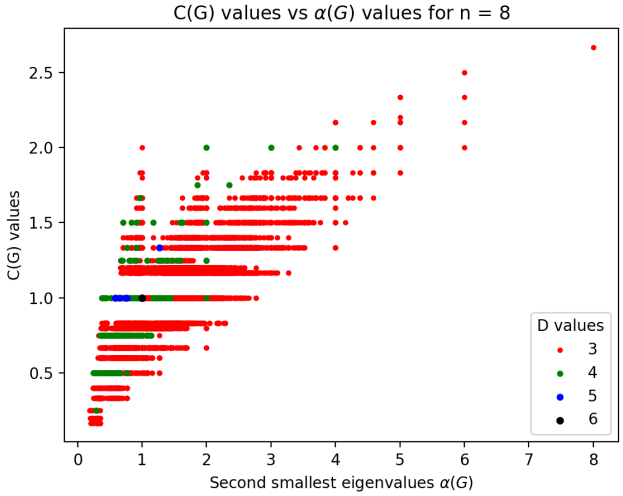
5.6 COMPARISON BETWEEN C(G), $\alpha(G)$ AND D VALS

We have made pairwise comparisons before between the above metrics. Now we compare these three metrics in a single plot to better understand the connection between them.




C(G) values vs α(G) values for n = 6
D values
3
4
5
6
C(G) values
Second smallest eigenvalues α(G)


C(G) values vs α(G) values for n = 7
D values
3
4
5
6
C(G) values
Second smallest eigenvalues α(G)


C(G) values vs α(G) values for n = 8
D values
3
4
5
6
C(G) values
Second smallest eigenvalues α(G)

From the above graphs, we can observe that there is no relation one way or the other between C(G) and D values and $\alpha(G)$ and D values whatsoever as the data points for higher D values are randomly scattered across the plot and do not show any kind of relation or clustering.

5.7 NEURAL NETWORK

The model has three dense layers of 64, 32 and, 1 output sizes each. The first two layers use the ReLU activation function. I have used the *adam* optimizer and Mean Square Error as the loss function. For the actual training and testing part, I proceeded with taking sets of graphs with different sets of vertices for both the training and testing. For the set with smaller n we rescale the matrix by first flattening and then filling with zeros to match the size. For disjoint sets we get a suboptimal performace with an accuracy of around 21% which is akin to random guessing.

A

APPENDIX

Apart from the various available literature, we perform the experiments via python code. As our approach requires graphs for a specific value of n which is the number of vertices, it is crucial to test our methods using all the possible number of graphs with n vertices.

To do this, we first start with generating all possible random matrices which are symmetric and create corresponding graphs for them. As node ordering doesn't matter, we end up with many duplicate graphs.

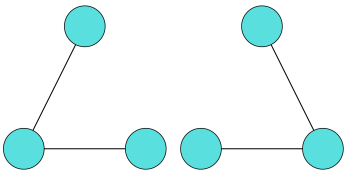


Figure A.1: A pair of isomorphic graphs.

Figure A.1 shows one such example where the adjacency matrix is different but the graph is the same. We have to remove such duplicates. Therefore we started with enumerating the degree of all the nodes and separating them in different classes based on the number of edges in the graph as well as the degrees of all the nodes. However this approach does not scale well above values of n such as 6, 7 and so on. The issue is the graphs should be a single connected component as well as they should not disregard duplicate configurations.

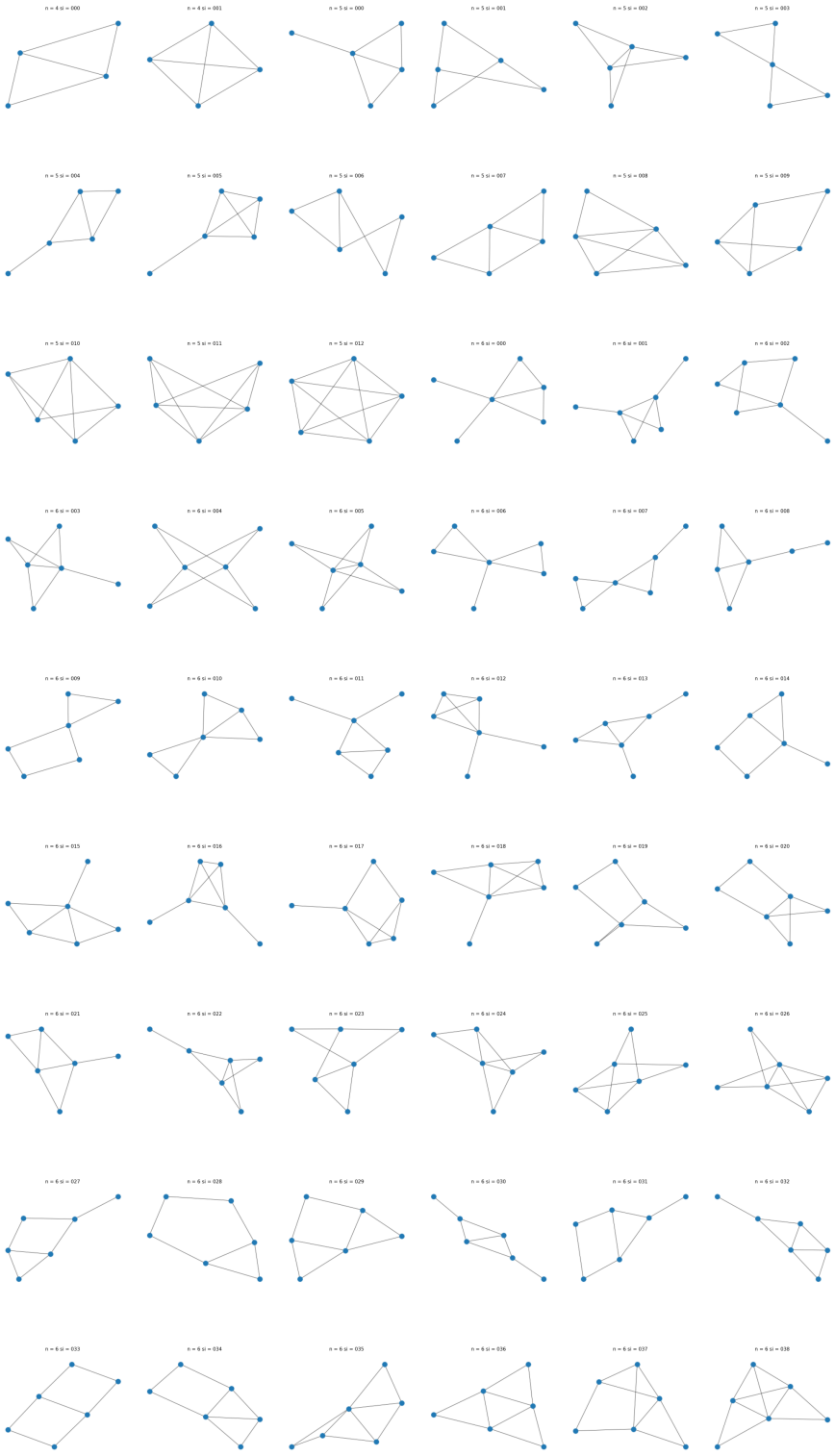
The set of all possible graphs for every value n which is for interest to us was therefore taken and processed as an internal python package for further use in the experiments [1].

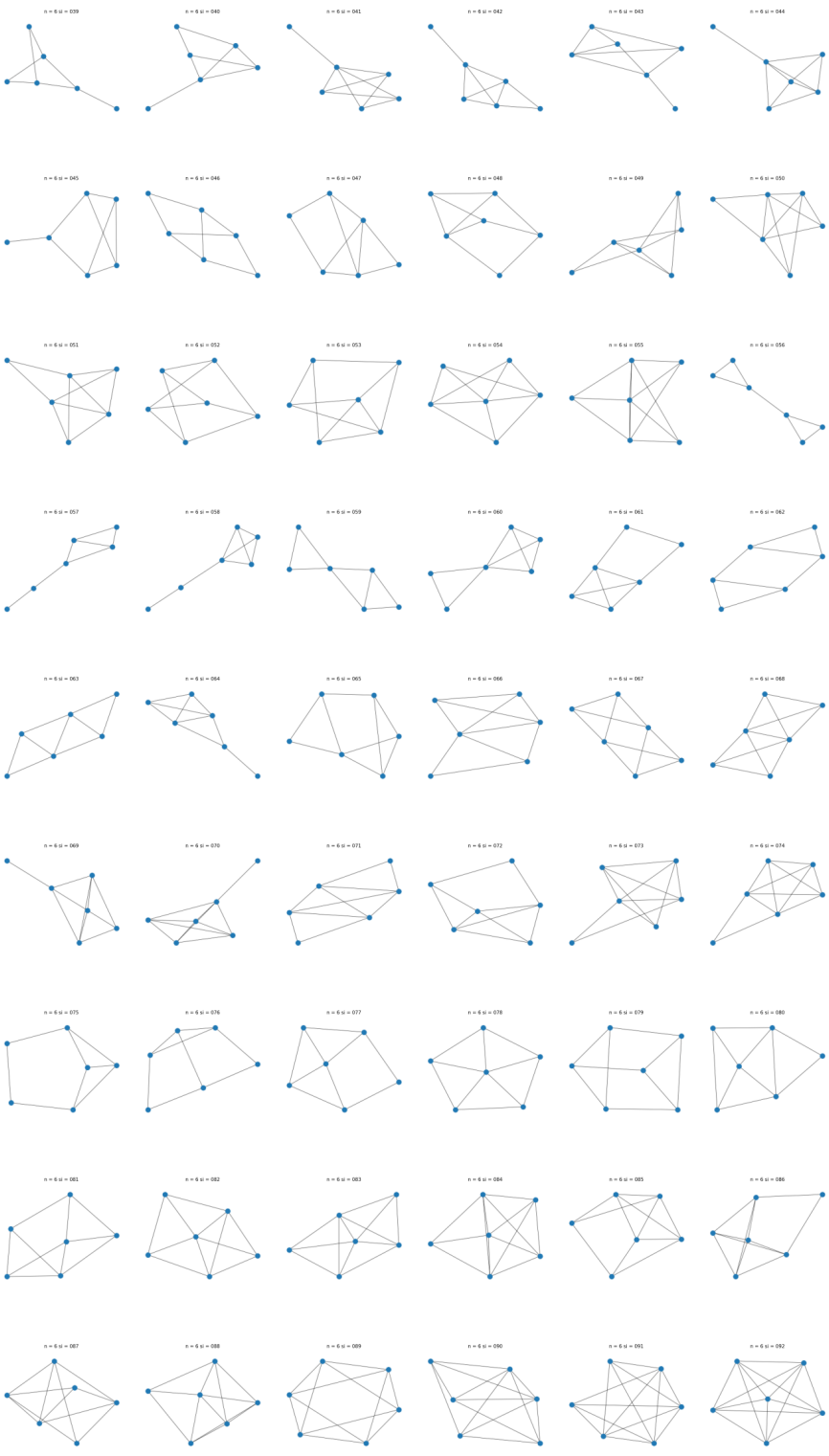
1 The data can be obtained in `g6` format from https://users.cecs.anu.edu.au/~bdm/data/graphs.html

Algorithm 1 Cheeger algorithm

```
1: function CHEEGER(G_1)
2:     ls_cv ← ∅
3:     for i ← 2 to ⌊n/2⌋ do
4:         for nodes in COMBINATIONS(G_1, i) do
5:             edge_set ← ∅
6:             for j in nodes do
7:                 edges_j ← {sorted(t) | t ∈ G_1.edges(j)}
8:                 edge_set ← edge_set ∪ edges_j
9:             end for
10:            num_dels ← 0
11:            num_s ← 0
12:            for e in edge_set do
13:                if e[0] in nodes and e[1] in nodes then
14:                    num_s ← num_s + 1
15:                else
16:                    num_dels ← num_dels + 1
17:                end if
18:            end for
19:            if num_s ≠ 0 then
20:                ls_cv.append(num_dels/num_s)
21:            else
22:                continue
23:            end if
24:        end for
25:        return min(ls_cv)
26:    end for
27: end function
```

Below are all the graphs which satisfy the $n < k$ constraint, that is, the number of nodes are less than edges in the graph.




n = 6 si = 039
n = 6 si = 040
n = 6 si = 041
n = 6 si = 042
n = 6 si = 043
n = 6 si = 044
n = 6 si = 045
n = 6 si = 046
n = 6 si = 047
n = 6 si = 048
n = 6 si = 049
n = 6 si = 050
n = 6 si = 051
n = 6 si = 052
n = 6 si = 053
n = 6 si = 054
n = 6 si = 055
n = 6 si = 056
n = 6 si = 057
n = 6 si = 058
n = 6 si = 059
n = 6 si = 060
n = 6 si = 061
n = 6 si = 062
n = 6 si = 063
n = 6 si = 064
n = 6 si = 065
n = 6 si = 066
n = 6 si = 067
n = 6 si = 068
n = 6 si = 069
n = 6 si = 070
n = 6 si = 071
n = 6 si = 072
n = 6 si = 073
n = 6 si = 074
n = 6 si = 075
n = 6 si = 076
n = 6 si = 077
n = 6 si = 078
n = 6 si = 079
n = 6 si = 080
n = 6 si = 081
n = 6 si = 082
n = 6 si = 083
n = 6 si = 084
n = 6 si = 085
n = 6 si = 086
n = 6 si = 087
n = 6 si = 088
n = 6 si = 089
n = 6 si = 090
n = 6 si = 091
n = 6 si = 092

A.1 PYTHON LIBRARIES USED

- The numpy library has been used for more memory efficient matrix based computations.
- The networkx [3] library implements several interfaces for effective working with graphs and large-scale networks.
- The matplotlib and plotly [4] libraries are used to create various plots for visualizing the connection between the data and observing the graphs.
- Tensorflow has been used in neural network training and testing.
- Scipy [11] has been used to compute metrics such as the correlation and at various points it has been helpful in knowing the relationship between the data points.

BIBLIOGRAPHY

[1] Chris Godsil and Gordon Royle. "Algebraic Graph Theory." In: vol. 207. Graduate Texts in Mathematics. New York, NY: Springer New York, 2001. ISBN: 978-0-387-95220-8 978-1-4613-0163-9. DOI: 10.1007/978-1-4613-0163-9. URL: http://link.springer.com/10.1007/978-1-4613-0163-9 (visited on 04/24/2024).

[2] Oded Goldreich. "Basic Facts about Expander Graphs." en. In: *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation: In Collaboration with Lidor Avigad, Mihir Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan, Dana Ron, Madhu Sudan, Luca Trevisan, Salil Vadhan, Avi Wigderson, David Zuckerman*. Ed. by Oded Goldreich. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2011, pp. 451–464. ISBN: 978-3-642-22670-0. DOI: 10.1007/978-3-642-22670-0_30. URL: https://doi.org/10.1007/978-3-642-22670-0_30 (visited on 01/23/2024).

[3] Aric Hagberg, Pieter J. Swart, and Daniel A. Schult. "Exploring network structure, dynamics, and function using NetworkX." In: (Jan. 2008). URL: https://www.osti.gov/biblio/960616.

[4] Plotly Technologies Inc. *Collaborative data science*. 2015. URL: https://plot.ly.

[5] H. Li. *Properties and Applications of Graph Laplacians*. 2022. URL: https://scholar.google.com/citations?view_op=view_citation&hl=en&user=25RFqZcAAAAJ&citation_for_view=25RFqZcAAAAJ:u-x6o8ySG0sC (visited on 03/06/2024).

[6] A. Marsden. "EIGENVALUES OF THE LAPLACIAN AND THEIR RELATIONSHIP TO THE CONNECTEDNESS." In: 2013. URL: https://math.uchicago.edu/~may/REU2013/REUPapers/Marsden.pdf (visited on 01/22/2024).

[7] *Ramanujan graph*. en. Page Version ID: 1193483172. Jan. 2024. URL: https://en.wikipedia.org/w/index.php?title=Ramanujan_graph&oldid=1193483172 (visited on 02/13/2024).

[8] Joel Siegel. "EXPANDER GRAPHS." In: 2014. URL: https://www.semanticscholar.org/paper/EXPANDER-GRAPHS-Siegel/a87f5a5be362465866051f47e060bed1f09d1219 (visited on 01/23/2024).

[9] Daniel A. Spielman. "Constructing Error-Correcting Codes from Expander Graphs." en. In: *Emerging Applications of Number Theory*. Ed. by Dennis A. Hejhal, Joel Friedman, Martin C. Gutzwiller, and Andrew M. Odlyzko. The IMA Volumes in Mathematics

and its Applications. New York, NY: Springer, 1999, pp. 591–600. ISBN: 978-1-4612-1544-8. DOI: 10.1007/978-1-4612-1544-8_25. URL: https://doi.org/10.1007/978-1-4612-1544-8_25 (visited on 03/06/2024).

[10] Daniel Spielman. *Spectral and Algebraic Graph Theory*. en. Incomplete Draft, dated December 4, 2019. Yale University: Yale University, 2019. URL: http://cs-www.cs.yale.edu/homes/spielman/sagt.

[11] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python." In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.