ON THE ALGEBRAIC CONNECTIVITY OF GRAPHS

by

Timothy Salamon

Submitted in partial fulfillment of the requirements
for the degree of Master of Science

at

Dalhousie University
Halifax, Nova Scotia
April 2022

Table of Contents

List of Tables . . . . **iv**

List of Figures . . . . **vi**

Abstract . . . . **ix**

Acknowledgements . . . . **x**

Chapter 1 Introduction **1**

Chapter 2 Regular Graphs with Tree Subgraphs **6**

2.1 A Generalized Upper Bound 6

2.2 A Two Term Asymptotic Estimate 19

2.3 A Second Upper Bound 24

Chapter 3 Algebraic Connectivity of Necklace Graphs **32**

3.1 General Necklace Graphs 33

3.2 Regular Necklace Graphs 38

Chapter 4 Algebraic Connectivity of Path-Like Graphs **47**

4.1 Hourglass Graphs 47

4.2 Path-Like Graphs With One Middle Block 52

Chapter 5 Algorithms on Empty Graphs **56**

5.1 Random Edge Addition 56

5.2 The Achlioptas Process 58

5.3 The Edge-Augmentation Algorithm 59

5.3.1 Random Tree Augmentation 61

5.4 Brute Force Search 62

5.5 Comparison 63

Chapter 6 Augmentation from Graph Families **66**

6.1 Starting from Random Graphs 66

6.2 Starting from Random Regular Graphs 69

6.3 Starting from Complete Bipartite Graphs 72

Chapter 7 Edge Addition and Deletion **76**

7.1 Using the Fiedler Vector 76

7.2 Using Brute Force 78

7.3 Combining the Fiedler Vector and Brute Force 80

7.3.1 An Improved Algebraic Connectivity 82

Chapter 8 Conclusion **84**

Bibliography **89**

Appendix A Matlab M Files **92**

A.1 Random Edge Addition 92

A.2 The Achlioptas Process 93

A.3 The Edge-Augmentation Algorithm 96

A.4 Brute Force Search 97

List of Tables

2.1 Upper bound values for the algebraic connectivity of d-regular graphs containing root-connected perfect trees of height K as subgraphs. 15

2.2 Comparison between theoretical and computed algebraic connectivity in large random 10-regular graphs. 16

2.3 Minimum and maximum algebraic connectivity of quartic graphs based on girth and order. 18

2.4 Numerical values and asymptotic estimates for the algebraic connectivity of cubic graphs formed by the T_K graph. 23

2.5 Upper bound values for the algebraic connectivity of cubic graphs having level binary trees with a root of degree 3 as subgraphs. . 29

2.6 Minimum and maximum algebraic connectivity of cubic graphs based on girth and order. 31

3.1 First term asymptotic estimates valid for the algebraic connectivity of d-regular necklace graphs containing c copies of $K_{d+1}\backslash\{u,v\}$, the complete graph K_{d+1} with the edge $\{u,v\}$ being removed. . 46

3.2 Numerical values for the algebraic connectivity of d-regular necklace graphs containing c copies of $K_{d+1}\backslash\{u,v\}$, the complete graph K_{d+1} with the edge $\{u,v\}$ being removed, obtained from the smallest root of equation (3.13) with $k = 1$. 46

4.1 Numerical values of the algebraic connectivity of odd degree d-regular hourglass graphs and path-like graphs with one middle block. 55

5.1 Comparison between average and maximum final algebraic connectivity returned by the four algorithms on empty graphs. . . 64

6.1 Comparison between initial and final algebraic connectivity amongst random regular graphs. 72

6.2 Comparison between initial and final algebraic connectivity amongst complete bipartite graphs. 75

8.1 Summary of algorithms increasing algebraic connectivity from an originally empty graph. 88

List of Figures

2.1 The graph T_3, consisting of two perfect binary trees of height 3 joined by an edge connecting their roots. 7

2.2 A cubic graph with diameter $D = 3$ containing the subgraph T_2 with labelled roots r_1 and r_2. 8

2.3 The graph $T_{3,3}$ consists of two root-connected tertiary trees of height 3. 9

2.4 The cubic graph formed by the T_3 graph. 19

2.5 Behaviour of the function $1 - 2z - 2^{J+1}z^{2J+2} + 2^{J+1}z^{2J+1}$ for $J = 2, \ldots, 5$. Note the root at $z = \frac{\sqrt{2}}{2}$ and that the other roots approach both $\frac{1}{2}$ and 1 as $J \to \infty$. 23

2.6 A root-connected binary tree with leaves on levels 2 and 3. . . 24

2.7 A level binary tree with a root of degree 3. 25

3.1 The necklace graph $N_{K_2,4}$ composed of 4 copies of the complete graph K_2 is a cycle of length 8. 32

3.2 The necklace graph $N_{D,5}$ composed of 5 copies of the diamond graph D. 39

3.3 Intersection points between the cubic function $\lambda(\lambda - 5)^2$ and the expression $-2(5 - 2)(\cos(\frac{2\pi k}{6}) - 1)$ with $n = 5$, $c = 6$, and $k = 1, 2, 3$. The orange line corresponds to $k = 1$ and the first intersection point has coordinates (0.1263,3). 45

4.1 The smallest hourglass graph H_{K_3}. 48

4.2 A 3-regular hourglass graph. 49

4.3 The 5-regular hourglass graph obtained by following the given steps. 50

4.4 The cubic path-like graph with 3 blocks obtained by following the steps. 53

5.1 Algebraic connectivity plot and degree histogram for the random edge addition algorithm on graphs with 50 vertices and 250 final edges. 57

5.2 Algebraic connectivity plot and degree histogram of the Achlioptas process algorithm with 60 potential edges per iteration on graphs with 50 vertices and 250 final edges. 58

5.3 Algebraic connectivity plot and degree histogram of the standard edge-augmentation algorithm on graphs with 50 vertices and 250 final edges. 60

5.4 Algebraic connectivity plot and degree histogram of the brute force search algorithm with 50 vertices and 250 final edges. . . 63

5.5 Comparison between the algebraic connectivity plots from the standard edge-augmentation algorithm, the Achlioptas process, random edge addition, and brute force search. 65

6.1 Algebraic connectivity plot and degree histogram of the standard edge-augmentation algorithm starting from a random graph. 67

6.2 Algebraic connectivity plot and degree histogram of the standard edge-augmentation algorithm followed by random edge addition. 68

6.3 Comparison between the standard edge-augmentation algorithm, random edge addition followed by augmentation, and augmentation followed by random edge addition for graphs with 50 vertices and 250 total edges. 69

6.4 Algebraic connectivity plot and degree histogram of the standard edge-augmentation algorithm starting from a random 5-regular graph. 70

6.5 Algebraic connectivity plots of the edge-augmentation algorithm starting from a random 3-regular to a random 9-regular graph. 71

6.6 Algebraic connectivity plot and degree histogram of the standard edge-augmentation algorithm starting from the complete bipartite graph $K_{2,48}$. 73

6.7 Algebraic connectivity plots of the edge-augmentation algorithm starting on the complete bipartite graphs $K_{1,49}$ to $K_{5,45}$. . . . 74

7.1 Algebraic connectivity plot and degree histogram of the algorithm adding and deleting edges according to the Fiedler vector. 77

7.2 Algebraic connectivity plot and degree histogram of the brute force addition and deletion algorithm. 79

7.3 Algebraic connectivity plot and degree histogram of the Fiedler vector addition and brute force deletion algorithm on graphs with 50 vertices and 250 edges. 80

7.4 Algebraic connectivity plot and degree histogram of the brute force addition and Fiedler vector deletion algorithm on graphs with 50 vertices and 250 edges. 81

7.5 Comparison between the edge-augmentation algorithm, the Fiedler vector addition and brute force deletion algorithm, and the brute force addition and Fiedler vector deletion algorithm for graphs with 50 vertices and 250 total edges. 83

8.1 The graph obtained by the edge-augmentation algorithm with 50 vertices and 250 edges. 87

Abstract

Algebraic connectivity, or the second smallest eigenvalue of the Laplacian matrix, is a well-studied parameter in spectral graph theory. In this thesis, we present new upper bounds and asymptotic estimates for the algebraic connectivity of regular graphs. They include a generalization of an upper bound given by Kolokolnikov as well as an upper bound valid for specific cubic graphs. Furthermore, we introduce two new graph families, which we call the necklace graphs and the hourglass graphs. We proceed to determine the complete spectrum of general necklace graphs in terms of a matrix involving roots of unity. We then consider a class of regular necklace graphs and derive a first term asymptotic estimate applicable to their algebraic connectivity. We also numerically investigate which algorithm returns graphs with the highest algebraic connectivity amongst those with fixed order and size.

Acknowledgements

I would like to express my gratitude to my co-supervisors Dr. Jeannette Janssen and Dr. Theodore Kolokolnikov. Your guidance and support made this thesis possible. I would also like to thank Dr. Jason Brown and Dr. David Iron for reading this thesis. Finally, I would like to thank my mother Jarmila Kurucova. I would not be who I am and where I am today without you.

Chapter 1

Introduction

Let G be a simple undirected graph with vertices $v_1, \ldots, v_n$ and edge set $E(G)$. Its *adjacency matrix* is defined as the $n \times n$ symmetric matrix A with entry $a_{i,j}$ given by

$$\begin{cases} a_{i,j} = 1 \text{ if there is an edge between vertex } v_i \text{ and } v_j \\ a_{i,j} = 0 \text{ otherwise.} \end{cases}$$

The *degree matrix* of G is defined as the $n \times n$ diagonal matrix D with entry $d_{i,j}$ given by

$$\begin{cases} d_{i,j} = deg(v_i) \text{ if } i = j \\ d_{i,j} = 0 \text{ otherwise,} \end{cases}$$

where $deg(v_i)$ denotes the degree of vertex v_i. The *diameter* of G is the maximum distance between any pair of vertices. The *girth* of G is the length of its shortest cycle and is defined to be infinity if the graph is acyclic.

The *Laplacian matrix* of G, denoted by L, is defined as

$$L = D - A.$$

As A is symmetric and D is diagonal, L is a symmetric matrix. Furthermore, as its quadratic form, given by $x^T L x = \sum_{(i,j) \in E(G)} (x_i - x_j)^2$, is non-negative for all $x = (x_1, \ldots, x_n)^T$, L is positive-semidefinite [1]. In fact, if G is connected, its Laplacian matrix L has an eigenvalue of 0 with corresponding eigenvector $(1, \ldots, 1)^T$ [2]. Since L is symmetric, all of its eigenvalues are real and all of its eigenvectors corresponding to different eigenvalues are orthogonal. Notably, this implies that all other eigenvectors must be orthogonal to $(1, \ldots, 1)^T$. In turn, the sum of their entries must be zero.

The Laplacian matrix arises naturally in relation to the diffusion process in graphs [3]. That is, for a vector $x = (x_1, \ldots, x_n)^T$ denoting weight x_i on vertex v_i, the quantity $a_{i,j}(x_i - x_j)$ represents the information flow from vertex v_i to vertex v_j

across their adjacent edge. The i^{th} entry of Lx is then given by

$$(Lx)_i = \sum_{v_j \in N(v_i)} (x_i - x_j), \tag{1.1}$$

where the sum is taken over all neighbours of vertex v_i. For an eigenvalue λ of the Laplacian such that

$$Lx = \lambda x, \tag{1.2}$$

we can then construct a system of equations of the form

$$\lambda x_i = \sum_{v_j \in N(v_i)} (x_i - x_j) \tag{1.3}$$

for each weight x_i. In this case, the weight vector x is also an eigenvector of L. Eigenvectors can thus be represented by an assignment of weights to the vertices of the graph.

The *algebraic connectivity* of G, symbolized by $\lambda_2(G)$, is the second smallest eigenvalue of L. Since its introduction by Miroslav Fiedler in 1973, it has been used as a tool to analyze the structure of graphs. That is, graphs with high algebraic connectivity also have higher vertex and edge connectivity [4].

The algebraic connectivity of a graph on n vertices can also be defined using the *Rayleigh quotient* [5]. That is, for a vector $x = (x_1, \ldots, x_n)^T$, the algebraic connectivity of G is given by

$$\lambda_2(G) = \min_{x \in \mathbb{R}^n, x \perp (1,1,\ldots,1)^T} \frac{\sum_{i,j \in E(G)} (x_i - x_j)^2}{\sum_{j=1}^{n} x_j^2}.$$

The Rayleigh quotient normalizes the quadratic form of the Laplacian matrix and returns the algebraic connectivity when x is its corresponding eigenvector. These eigenvectors are referred to as *Fiedler vectors* and can be used to partition graphs based on the sign of each of their entries [6]. In particular, we note the following lemma by de Abreu [5].

Lemma 1.0.1 *Let G be a connected graph on n vertices weighted by the Fiedler vector $(x_1, \ldots, x_n)^T$. If $x_i \neq 0$ for $1 \leq i \leq n$, then the set of all edges (i, j) for which $x_i \cdot x_j < 0$ forms an edge cut of G with exactly two components.*

As the multiplicity of the zero eigenvalue of the Laplacian matrix is equal to the number of connected components in a graph, its algebraic connectivity is greater than zero if and only if it is connected [2]. Hence, the algebraic connectivity of a disconnected graph is zero and this number serves as a lower bound for this parameter.

Bounds to algebraic connectivity are a subject of continual research, see [7–10]. This parameter has also been related to the matching number [11] and to the domination number [12]. Among them, vertex connectivity provides a well-known upper bound. That is, $\lambda_2(G) \leq \kappa(G)$, where $\kappa(G)$ denotes the minimum number of vertices needing to be removed in order to disconnect the graph [5].

Algebraic connectivity is also related to the *Cheeger constant*. This constant is defined as $h(G) = \min \frac{|E(X,Y)|}{\min\{|X|,|Y|\}}$ where the minimum is taken over all partitions of the vertex set of G into non-empty sets X and Y and where $E(X,Y)$ are the edges between X and Y [13]. The relationship between algebraic connectivity and the Cheeger constant is then defined through the Cheeger Inequalities [14]. Namely,

$$\frac{h^2(G)}{2\Delta(G)} \leq \lambda_2(G) \leq 2h(G)$$

where $\Delta(G)$ denotes the maximum degree of G.

The Cheeger constant is studied in the context of *expander graphs*. These are sparse graphs with high edge, vertex, or spectral expansion properties and with several applications in network design, complexity theory, and theoretical computer science [15]. *Ramanujan graphs*, as well as many other regular graphs, are known to make excellent expanders [16]. For this reason, their algebraic connectivity is often of interest.

In regular graphs, a clear relationship exists between the eigenvalues of the Laplacian and the eigenvalues of the adjacency matrix. For a d-regular graph G, it is given by $\lambda_2(G) = d - \theta_2(G)$, where $\theta_2(G)$ denotes the second highest eigenvalue of its adjacency matrix [17]. The difference between the two highest eigenvalues of the adjacency matrix of a graph is called its *spectral gap* [18]. As the highest eigenvalue of a d-regular graph is d, its algebraic connectivity is equal to its spectral gap.

Still, the value of $\theta_2(G)$ for large regular graphs might not be easily obtainable. In turn, determining their algebraic connectivity becomes difficult. Hence, most of the research has focused on finding upper bounds for this parameter [19, 20]. In particular, Kolokolnikov found an upper bound for the algebraic connectivity of a cubic graph containing two perfect binary trees joined by an edge connecting their roots as a subgraph [21]. In Chapter 2, we refer to these two binary trees as being *root-connected*.

This thesis is divided into two parts. In the first part, we establish new upper bounds and asymptotic estimates for the algebraic connectivity of regular graphs. We begin by generalizing Kolokolnikov's work on cubic graphs to hold for d-regular graphs with specific tree subgraphs. We show that this bound is tight for some n by providing an example of a 4-regular graph that attains it. We also explore the relationship between algebraic connectivity and girth, showing that d-regular graphs with the same order and higher girth also have higher algebraic connectivity.

We then proceed to derive a two term asymptotic estimate for an eigenvalue of the Laplacian matrix of specific cubic graphs. These graphs are obtained by a specific configuration of the leaves of their root-connected binary tree subgraph. Numerically, we show that this asymptotic estimate applies to the graph's algebraic connectivity.

We then consider cubic graphs with root-connected binary trees with leaves differing by one level as subgraphs. An upper bound for their algebraic connectivity, involving the smallest root of a function on a given interval, is obtained. This bound is also shown to be achievable for some n.

In Chapter 3, we introduce a new graph family which we call the *necklace graphs*. We derive the complete spectrum of their Laplacian matrices based on the union with multiplicity of the eigenvalues of matrices involving roots of unity. We also show that their Fiedler vectors, and thus their algebraic connectivity, are related to the first root of unity. We then focus our attention to a particular family of regular necklace graphs. Their spectrum is established, as well as a first term asymptotic estimate for one of their eigenvalues. Once again, we numerically show that the estimate applies to their algebraic connectivity.

In Chapter 4, we build upon the work on cubic graphs done by Guiduli [22, 23] and improved by Abdi, Ghorbani, and Imrich [24,25] following a conjecture by Babai. Their work showed that these graphs have minimal algebraic connectivity amongst all cubic graphs of the same order. In this chapter, we generalize the structure of the cubic graphs to d-regular graphs with odd degree. To do so, we first define a second graph family which we call the *hourglass graphs*. Supported by Guiduli's work, we conjecture that these graphs also have minimal algebraic connectivity amongst regular graphs of the same order and degree.

The second part of this thesis begins in Chapter 5. The goal of this half of the thesis is to determine which algorithm constructs graphs with the highest algebraic connectivity amongst all graphs with fixed order and size. Each algorithm's performance is compared by the resulting graph's algebraic connectivity as well as by its computational complexity. The final graph's structure in terms of degree distribution is also of interest. All algorithms were written in MATLAB.

We begin by exploring four algorithms on initially empty graphs of order 50. One edge is added at each iteration until a total of 250 edges is reached. Two algorithms involve randomness, one involves the Fiedler vector, and the last one involves brute force. The computer code for these algorithms is included in Appendix A. The following lemma by Fiedler [4] also guarantees that adding edges to a graph with a fixed number of vertices can only increase its algebraic connectivity.

Lemma 1.0.2 *Let G_1 be a subgraph of G_2 with the same set of vertices. Then $\lambda_2(G_1) \leq \lambda_2(G_2)$.*

In Chapter 6, we explore how the best performing algorithm of the previous chapter behaves when starting on different graph families. These include random graphs, random regular graphs, and complete bipartite graphs. The last two are known to be algebraic connectivity maximizers under some conditions [21]. General trends as well as specific examples are addressed.

The notion of edge deletion in the algorithms is investigated in Chapter 7. In the two outlined methods, particular factors are chosen in order to determine both the edges which increase algebraic connectivity the most as well as those that do so the least. Naturally, the ones that increase it the most are added while the ones that do so the least are deleted. The combination of these methods is also considered. The algebraic connectivity of the graph obtained by each algorithm on originally empty graphs is included in the form of an ordered table at the end of this thesis.

Chapter 2

Regular Graphs with Tree Subgraphs

2.1 A Generalized Upper Bound

As every connected 2-regular graph is a cycle, finding an expression for the algebraic connectivity of this graph family would be enough to conclude this case. In fact, such an expression is known. For a cycle C_n on n vertices, its algebraic connectivity is given by $\lambda_2(C_n) = 2(1 - \cos(\frac{2\pi}{n}))$ [5]. However, regular graphs with higher degree cannot be represented by a single graph family. For example, both the complete graph K_{d+1} and the complete bipartite graph $K_{d,d}$ are d-regular graphs. Yet, many of their other properties, such as their order, size, and girth are different. For this reason, we begin by considering upper bounds for the algebraic connectivity of cubic, or 3-regular, graphs. The following bound, based on a cubic graph's diameter, was given by Nilli in 2004 [26]:

Lemma 2.1.1 *Let G be a cubic graph with diameter D. Then an upper bound for its algebraic connectivity is $\lambda_2(G) \leq 3 - 2\sqrt{2}\cos(2\pi/D)$.*

Now, consider a graph in which two perfect binary trees of height K are joined by an edge connecting their roots. For clarity, we define the roots r_1 and r_2 to be at level 1 and the height K as one more than the distance from any leaf to its root. One such graph, with $K = 3$, is shown in Figure 2.1. We denote this graph family by T_K and label its two roots by r_1 and r_2. Based on this structure, the following bound was obtained by Kolokolnikov in 2014 [21]:

Lemma 2.1.2 *Suppose that a cubic graph G has T_K as a subgraph. Then an upper bound for its algebraic connectivity is $\lambda_2(G) \leq 3 - 2\sqrt{2}\cos(\pi/K)$.*

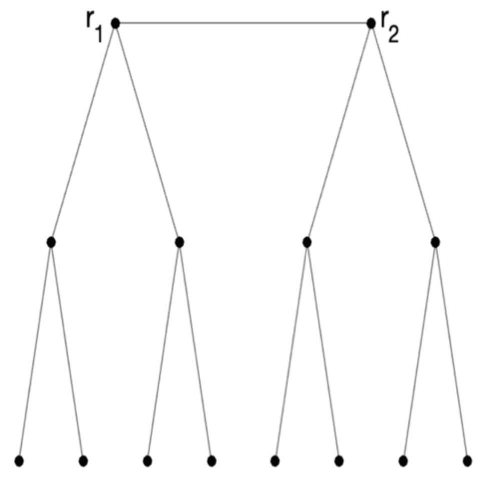


Figure 2.1: The graph T_3, consisting of two perfect binary trees of height 3 joined by an edge connecting their roots.

The next proposition relates the structure of T_K with a cubic graph's diameter.

Proposition 2.1.3 *Let G be a cubic graph containing a T_K subgraph. Then the diameter of G is at least K.*

Proof. Let T_K be a subgraph of G. Then, there is a path from each leaf vertex of the binary tree with root r_1 to r_2, the root of the second tree. Note that the distance between these two vertices is K and that no shorter path is possible. If such a path existed, then T_K would contain a vertex with degree 4 which is impossible since G is a cubic graph. Thus, if T_K is a spanning subgraph of G, then D, the diameter of G, is K. Yet, if T_K is not a spanning subgraph, then G contains additional vertices not in the vertex set of T_K that may increase its diameter. Hence, the diameter of G must be at least K. ■

Provided that a cubic graph has T_K as a subgraph, this lemma shows that Kolokolnikov's bound from Lemma 2.1.2 is an improvement on the one given by Nilli in Lemma 2.1.1 when $K \leq D \leq 2K$. The case $D = K$ comes directly from the fact that $\cos(\frac{2\pi}{K}) < \cos(\frac{\pi}{K})$ for any $K \geq 2$. We illustrate the case $D > K$ with an example. Consider the cubic graph in Figure 2.2, with diameter $D = 3$ and a clear T_2 subgraph. Using MATLAB, we found that its algebraic connectivity is equal to 0.7639. With $D = 3$, Nilli's bound results in 4.4142 while Kolokolnikov's results in 3. Yet, note that

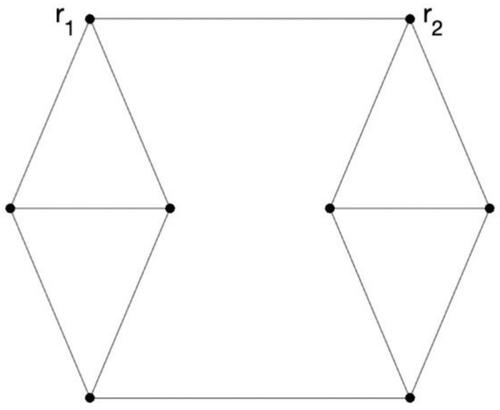


Figure 2.2: A cubic graph with diameter $D = 3$ containing the subgraph T_2 with labelled roots r_1 and r_2.

Nilli's bound is better for cubic graphs when $D > 2K$. For example, the cubic graph in Figure 3.2 has an algebraic connectivity of 0.1904, a diameter of 7, and contains a T_2 subgraph. Nilli's bound results in 1.2365 while Kolokolnikov's results in 3. Note that the bounds are equal if $D = 2K$. Thus, Kolokolnikov's bound is only better when $K \leq D \leq 2K$. We now provide a necessary condition for a graph to contain T_K as a subgraph.

Lemma 2.1.4 *Let G be a cubic graph. If its girth is at least $2K$, then G contains T_K as a subgraph.*

Proof. Let G be a cubic graph with a girth of at least $2K$ for some $K \geq 2$. Choose any edge and label its endpoints by r_1 and r_2. As G is cubic, these two vertices are each adjacent to two other vertices. Note that r_1 and r_2 must be adjacent to different vertices since the graph would otherwise have a girth of 3, which is less than $2K$. If $K = 2$, we are done since the graph contains T_2. If $K > 2$, then, by the same argument, these four new vertices must each be adjacent to two other vertices. This results in a graph with a T_3 subgraph. The process terminates with a graph containing a T_K subgraph once the value of K is reached. ■

We now generalize Kolokolnikov's bound to hold for any d-regular graph. To do so, we first define $T_{s,K}$ as a graph in which two perfect s-ary trees of height K are joined by an edge connecting their roots. In the previously considered case of cubic

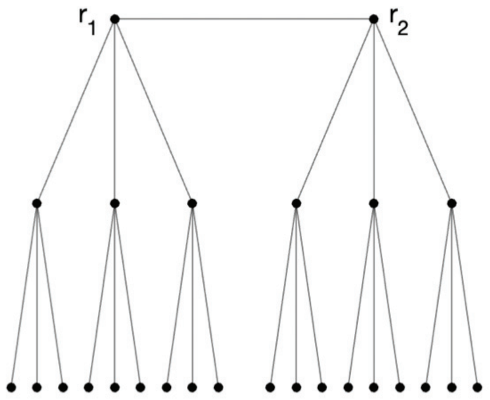


Figure 2.3: The graph $T_{3,3}$ consists of two root-connected tertiary trees of height 3.

graphs, this corresponds to $T_{2,K}$. Thus, we have $s = d - 1$ for $d \geq 3$. Also note that $T_{s,K}$ has order $\frac{2(s^K-1)}{s-1}$ as it is composed of two perfect s-ary trees, each with $\frac{s^K-1}{s-1}$ vertices. The $T_{3,3}$ graph, composed of two tertiary trees of height 3, is shown in Figure 2.3. We now proceed to state and prove the main theorem of this section.

Theorem 2.1.5 *Suppose that a d-regular graph G contains $T_{s,K}$ as a subgraph. Then $\lambda_2(G) \leq d - 2\sqrt{d-1}\cos(\pi/K)$.*

Proof. Recall that an eigenvector can be represented as an assignment of weights to the vertices of G. Furthermore, recall that all other eigenvectors must be orthogonal to the all-ones vector $(1, \ldots, 1)^T$. We then assign weights, chosen later, to the vertices as follows: for vertices at level k on the left tree, assign weight x_k; for those at level k on the right tree, assign weight $-x_k$; for all other vertices, assign a weight of zero. This is a valid choice for the eigenvector $x = (x_1, \ldots, x_n)^T$ since the sum of its entries is zero and so this vector is orthogonal to the all-ones vector.

We now derive the equations needed for λx_i based on equation (1.3) and the structure of its $T_{s,K}$ subgraph. We begin with r_1, the left root vertex of $T_{s,K}$. Note that this vertex is adjacent to the right root vertex r_2 and to $d-1$ vertices at level 2 of the left tree. By our choice of weights, the vertex r_1 has weight x_1, the vertex r_2 has weight $-x_1$, and the vertices at level 2 of the left tree have weight x_2. Using these values, the equation for the root vertex r_1 is then

$$\lambda x_1 = (x_1 - (-x_1)) + (d-1)(x_1 - x_2). \tag{2.1}$$

Note that r_2, the right root vertex of $T_{s,K}$, is adjacent to r_1 and to $(d-1)$ vertices at level 2 of the right tree, each with a weight of $-x_2$. The equation for this vertex is then

$$\lambda(-x_1) = (-x_1 - x_1) + (d-1)(-x_1 - (-x_2)).$$

As this equation is proportional to the one in (2.1), it may be omitted. In fact, it can be easily checked that the equations for all vertices of the right tree are proportional to those for the left tree by a factor of -1. We thus only derive the remaining equations for the vertices of the left tree.

We now consider the vertices at level 2 of the left tree, each with weight x_2. For such a vertex, note that it is adjacent to its parent root vertex at level 1 with weight x_1 and to $d-1$ vertices at level 3 with weight x_3. The equation for this vertex is then

$$\lambda x_2 = (x_2 - x_1) + (d-1)(x_2 - x_3). \tag{2.2}$$

Note that equation (2.2) holds for all vertices at level 2 of the left tree of $T_{s,K}$. Furthermore, for $2 \leq k \leq K-1$, it can be easily checked that each vertex at level k is adjacent to one vertex at level $k-1$ and to $d-1$ vertices at level $k+1$. Equation (2.2) thus generalizes to

$$\lambda x_k = (x_k - x_{k-1}) + (d-1)(x_k - x_{k+1}),\ k = 2, \ldots, K-1. \tag{2.3}$$

The remaining equation is obtained by considering the leaf vertices at level K. This time, however, the vertices to which the leaf vertex is adjacent to are undetermined. Still, each leaf vertex of $T_{s,K}$ must have d edges, one connecting it to its parent with weight x_{K-1} while the other $d-1$ connect it to either another leaf with weight $\pm x_K$ or to a vertex outside $T_{s,K}$ whose weight is zero. These options give rise to three distinct cases, which we consider separately.

Case 1 (The leaf is joined to a leaf of the same tree): Without loss of generality, assume that both leaves have weight x_K. Then, as both leaves have the same weight, we obtain that $x_K - x_K = 0$.

Case 2 (The leaf is joined to a leaf of the other tree): In this case, we may assume that the first leaf has weight x_K and the other has weight $-x_K$. Then, similarly to the first case, we obtain that $x_K - (-x_K) = 2x_K$.

Case 3 (The leaf is joined to a vertex outside $T_{s,K}$): Assume that the leaf has weight

x_K. Then, as the second vertex lies outside $T_{s,K}$, it has a weight of zero. Thus, as in the other two cases, we obtain that $x_K - 0 = x_K$.

As all three cases are possible, we note that the entries in (Lx) corresponding to the leaf vertices are maximized when each leaf is only made adjacent to its parent and to leaf vertices of the other tree. We refer to the graph obtained by this leaf configuration of $T_{s,K}$ by H, with corresponding matrix M and eigenvalue μ. Thus, the leaf configuration considered in case 2 results in the equation

$$\mu x_K = (x_K - x_{K-1}) + (d-1)(x_K - (-x_K)). \tag{2.4}$$

Note that as G and H both contain $T_{s,K}$ as a subgraph, the right-hand side of equations (2.1) and (2.3) will also apply for μx_1 and μx_k, respectively. We can thus substitute μ for λ in those equations. Together with (2.4), these equations can then be combined to create an eigenvalue problem for the algebraic connectivity of the cubic graph G. That is, since G has higher order than H and all of its vertices not in $T_{s,K}$ have a weight of zero, $\lambda_2(G)$ will be bounded above by any eigenvalue μ of the eigenvalue problem

$$\begin{cases} \mu x_1 = (x_1 - (-x_1)) + (d-1)(x_1 - x_2); & (2.5) \\ \mu x_k = (x_k - x_{k-1}) + (d-1)(x_k - x_{k+1}), \ k = 2, \dots, K-1; & (2.6) \\ \mu x_K = (x_K - x_{K-1}) + (d-1)(x_K - (-x_K)) & (2.7) \end{cases}$$

corresponding to the K by K tridiagonal matrix

$$M = \begin{bmatrix} d+1 & -(d-1) & & & & \\ -1 & d & -(d-1) & & & \\ & -1 & \ddots & \ddots & & \\ & & \ddots & d & -(d-1) & \\ & & & -1 & d & -(d-1) \\ & & & & -1 & 2d-1 \end{bmatrix}. \tag{2.8}$$

The matrix in (2.8) was obtained by considering equations (2.5), (2.6), and (2.7) as a system of equations. Namely, the entries in the k^{th} row of M correspond to the coefficients of the weights in the k^{th} equation of the system. Now, let z be a variable to be determined and consider the test vector $\mathbf{z} = (1, z, z^2, \dots, z^{K-1})^T$. By considering the equation $M\mathbf{z} = \mu\mathbf{z}$, we obtain that

$$\begin{bmatrix} d+1 & -(d-1) & & & & \\ -1 & d & -(d-1) & & & \\ & -1 & \ddots & \ddots & & \\ & & \ddots & d & -(d-1) & \\ & & & -1 & d & -(d-1) \\ & & & & -1 & 2d-1 \end{bmatrix} \begin{pmatrix} 1 \\ z \\ z^2 \\ \vdots \\ z^{K-1} \end{pmatrix} = \mu \begin{pmatrix} 1 \\ z \\ z^2 \\ \vdots \\ z^{K-1} \end{pmatrix}$$

which, after matrix multiplication, results in

$$\begin{pmatrix} (d+1)-(d-1)z \\ -1+dz-(d-1)z^2 \\ -z+dz^2-(d-1)z^3 \\ \vdots \\ -z^{K-3}+dz^{K-2}-(d-1)z^{K-1} \\ -z^{K-2}+(2d-1)z^{K-1} \end{pmatrix} = \begin{pmatrix} \mu \\ \mu z \\ \mu z^2 \\ \vdots \\ \mu z^{K-2} \\ \mu z^{K-1} \end{pmatrix}. \tag{2.9}$$

Note that equation (2.9) can also be written as

$$\begin{pmatrix} (d+1)-(d-1)z \\ (d-(d-1)z-\frac{1}{z})z \\ (d-(d-1)z-\frac{1}{z})z^2 \\ \vdots \\ (d-(d-1)z-\frac{1}{z})z^{K-2} \\ -z^{K-2}+(2d-1)z^{K-1} \end{pmatrix} = \begin{pmatrix} \mu \\ \mu z \\ \mu z^2 \\ \vdots \\ \mu z^{K-2} \\ \mu z^{K-1} \end{pmatrix} \tag{2.10}$$

by factoring out z from the second to the second-to-last entry of the vector on the left-hand side of the equation. As these entries were originally obtained from equation (2.6), that equation is satisfied for any $k = 2, \ldots, K-1$ whenever

$$\mu = d - (d-1)z - \frac{1}{z}. \tag{2.11}$$

We now derive our ansatz for the eigenvector of M. To do so, we first let $\mu^* = d-(d-1)w-\frac{1}{w}$ for some w. Then, setting $\mu = \mu^*$ in (2.11), we obtain

$$d-(d-1)z-\frac{1}{z} = d-(d-1)w-\frac{1}{w}. \tag{2.12}$$

By cancelling out d on both sides and multiplying both sides by -1, equation (2.12) simplifies to

$$(d-1)z + \frac{1}{z} = (d-1)w + \frac{1}{w}. \tag{2.13}$$

Multiplying both sides by wz and simplifying, equation (2.13) results in

$$(dz - z)w^2 - (dz^2 - z^2 + 1)w + z = 0 \tag{2.14}$$

whose solutions are

$$w_+ = z \text{ and } w_- = \frac{1}{(d-1)z}. \tag{2.15}$$

Our ansatz for the eigenvector is then

$$x_k = Az^k + B\left(\frac{1}{(d-1)z}\right)^k \tag{2.16}$$

for $k = 1, \ldots, K$ and non-zero A and B. Note that, for each k, this ansatz is a combination of the solutions for w found in (2.15). Furthermore, note that equation (2.6) may be simplified to the equation

$$\mu x_k = dx_k - (d-1)x_{k+1} - x_{k-1},\ k = 2, \ldots, K-1. \tag{2.17}$$

To find the value of z, we begin by rewriting equations (2.5) and (2.7) in terms of equation (2.6). That is, we write equation (2.5) as

$$\mu x_1 = (dx_1 - (d-1)x_2 - x_0) + x_0 + x_1$$

and equation (2.7) as

$$\mu x_K = (dx_K - (d-1)x_{K+1} - x_{K-1}) - (d-1)x_K - (d-1)x_{K+1}$$

By applying equation (2.17), these two equations reduce to

$$x_0 + x_1 = 0 \text{ and } x_K + x_{K+1} = 0 \tag{2.18}$$

By (2.16), we obtain the following four equations:

$$x_0 = A + B,$$

$$x_1 = Az + B\left(\frac{1}{(d-1)z}\right),$$

$$x_K = Az^K + B\left(\frac{1}{(d-1)z}\right)^K,$$

and

$$x_{K+1} = Az^{K+1} + B\left(\frac{1}{(d-1)z}\right)^{K+1}.$$

The equations in (2.18) now become

$$A + B + Az + B\left(\frac{1}{(d-1)z}\right) = 0$$

and

$$Az^K + B\left(\frac{1}{(d-1)z}\right)^K + Az^{K+1} + B\left(\frac{1}{(d-1)z}\right)^{K+1} = 0.$$

These can be rewritten as the following matrix equation

$$\begin{bmatrix} 1+z & 1+\frac{1}{(d-1)z} \\ z^K + z^{K+1} & \left(\frac{1}{(d-1)z}\right)^K + \left(\frac{1}{(d-1)z}\right)^{K+1} \end{bmatrix} \begin{pmatrix} A \\ B \end{pmatrix} = 0 \tag{2.19}$$

where $\begin{pmatrix} A \\ B \end{pmatrix}$ belongs to the kernel of the matrix. As a non-zero vector exists for which equation (2.19) equals zero, the 2×2 matrix must be singular with a determinant of zero. After taking its determinant and simplifying, we obtain the polynomial

$$z^{-2K} = (d-1)^K$$

so that $z = \sqrt{\frac{1}{d-1}}e^{\frac{2\pi ik}{2K}}$. The choice $k = 0$ corresponds to $z = \sqrt{\frac{1}{d-1}}$ and, substituting this value of z into (2.16), results in $x_k = 0$ for all k. To see this, note that as none of the vertices in $T_{v,K}$ were assigned a weight of x_0, this weight must be zero by our choice of weights. The equation $x_0 = A+B$ then yields $A = -B$, which, when applied to equation (2.16), results in $x_k = 0$ for all k. Although its weights sum to zero, this vector is not allowed since the zero vector is by convention not an eigenvector. The remaining choices are obtained by plugging in the value of z into equation (2.11). After some simplification, this yields

$$\mu = d - \sqrt{d-1}e^{\frac{2\pi ik}{2K}} - \sqrt{d-1}e^{\frac{-2\pi ik}{2K}}, k = 1, \ldots, K \tag{2.20}$$

or equivalently, by using the complex definition of cosine $\cos(\theta) = \frac{e^{i\theta}+e^{-i\theta}}{2}$ for any θ,

$$\mu = d - 2\sqrt{d-1}\cos\left(\frac{\pi k}{K}\right), k = 1, \ldots, K. \tag{2.21}$$

The smallest eigenvalue amongst those in (2.21) corresponds to the choice $k = 1$ and to the bound of the theorem. ■

Table 2.1 presents the upper bound values for $d = 3, \ldots, 9$ and $K = 2, \ldots, 20$. As can be seen from the table, the value of the upper bound decreases as K increases and d is held fixed. This is to be expected as, for large values of K, $\cos(\pi/K)$ tends to 1 and the values of the bound converge to $d - 2\sqrt{d-1}$. For fixed K, higher d values have higher upper bounds since those graphs are better connected.

Numerical Values of the Generalized Bound							
K \ d	3	4	5	6	7	8	9
2	3	4	5	6	7	8	9
3	1.5858	2.2679	3	3.7639	4.5505	5.3543	6.1716
4	1	1.5505	2.1716	2.8377	3.5359	4.2583	5
5	0.7118	1.1975	1.7639	2.3820	3.0366	3.7191	4.4235
6	0.5505	1	1.5359	2.1270	2.7574	3.4174	4.1010
7	0.4517	0.8790	1.3961	1.9707	2.5862	3.2325	3.9034
8	0.3869	0.7996	1.3045	1.8683	2.4739	3.1113	3.7738
9	0.3422	0.7448	1.2412	1.7976	2.3965	3.0276	3.6843
10	0.3100	0.7054	1.1958	1.7468	2.3408	2.9675	3.6200
15	0.2334	0.6116	1.0874	1.6256	2.2081	2.8241	3.4668
20	0.2064	0.5786	1.0493	1.5829	2.1613	2.7736	3.4128
$d - 2\sqrt{d-1}$	0.1716	0.5359	1	1.5279	2.1010	2.7085	3.3432

Table 2.1: Upper bound values for the algebraic connectivity of d-regular graphs containing root-connected perfect trees of height K as subgraphs.

The general bound of Theorem 2.1.5, as well as Nilli's and Kolokolnikov's bounds on cubic graphs, are supported by McKay's research on the eigenvalues of the adjacency matrix [27]. In 1981, he found that for a fixed degree d, the second largest eigenvalue of the adjacency matrix of a d-regular graph tends to $2\sqrt{d-1}$ as the number of vertices approaches infinity. Recall that, for a d-regular graph, $\lambda_2(G) = d - \theta_2(G)$ where $\theta_2(G)$ is the second largest eigenvalue of the adjacency matrix. Then, as expected, we obtain $\lambda_2(G) = d - 2\sqrt{d-1}$.

Using MATLAB, we can test McKay's result on random 10-regular graphs with a varying number of vertices in terms of percent error. The results are shown in Table 2.2, where algebraic connectivity is abbreviated as AC.

Based on Table 2.2, it is clear that this result is accurate for large regular graphs. As seen in the last row, the error between the theoretical and computed algebraic connectivity is less than one percent. However, judging from the first two rows, it is not optimal for smaller graphs. The percent error there is greater than ten percent. Still, this is to be expected as this bound is asymptotic. A jump in percent error between graphs containing 100 and 500 vertices is also noteworthy.

Algebraic Connectivity of Random 10-Regular Graphs			
Number of Vertices	Computed AC	Theoretical AC	Percent Error
50	4.6495	4	16.2375
100	4.4836	4	12.09
500	4.1004	4	2.51
1000	4.0862	4	2.155
5000	4.0212	4	0.53

Table 2.2: Comparison between theoretical and computed algebraic connectivity in large random 10-regular graphs.

Graphs whose algebraic connectivity manages to attain the bound in Theorem 2.1.5 are also of interest. In 2014, Kolokolnikov found that it is reached by four cubic graphs, each containing a T_K subgraph for some value of K [21]. Namely, the complete bipartite graph $K_{3,3}$ attains the bound when $K = 2$, the Heawood graph attains it for $K = 3$, the Tutte 8-cage when $K = 4$, and the Tutte 12-cage for $K = 6$. Note that no graph was found to attain this bound when $K = 5$. All four graphs are $(3, g)$-cages, or 3-regular graphs with girth g and minimum possible order. Cages for cubic graphs have received considerable attention and are known for girth $g = 3, \ldots, 12$ [28]. Yet, for degree $d > 3$, several cages remain undetermined.

The algebraic connectivity of complete bipartite graphs is known to equal the cardinality of the smallest set [5]. That is, $\lambda_2(K_{b,d}) = \min\{b, d\}$. In the case that both sets have the same cardinality, $\lambda_2(K_{d,d}) = d$ and these graphs form $(d, 4)$-cages. That is, d-regular graphs with girth 4 and minimum possible order. Considering that $\lambda_2(G) \leq d$ when $K = 2$, the bound is attained by all complete bipartite graphs of the form $K_{d,d}$ for $d \geq 3$.

For larger values of K, cages are also likely to have an algebraic connectivity which attains the value of the general bound. Indeed, they exist for any combination of girth and degree and their high girth implies that these graphs are likely to have high algebraic connectivity [29]. Yet, their order is exponential in girth regardless of degree [30]. For a graph, this means that a small increase in the length of its smallest cycle will result in a great increase in the number of its vertices.

In order to determine whether the bound of Theorem 2.1.5 is reached in regular graphs with a degree higher than 3, we performed a computational search on quartic graphs based on their girth and order. These graphs were taken from the House of Graphs and a short program to find their algebraic connectivity was written in Maple. The results can be found in Table 2.3, where algebraic connectivity is denoted by AC.

Note that, for $d = 4$, $s = d - 1 = 3$, and $K = 3$, the upper bound provides a value of 2.2679. Our search revealed that this bound is reached by a graph of order 20 with girth 5 and by a graph of order 26 with girth 6. The first graph does not have enough vertices to contain $T_{3,3}$, but the second is indeed the (4,6)-cage. This latter graph does contain $T_{3,3}$ as a subgraph.

This table leads to some further observations. First, note that, for fixed girth, both the minimum and maximum algebraic connectivity decrease as the number of vertices increases. For example, on girth 3, the minimum algebraic connectivity for a quartic graph on 7 vertices is 3 while it is 0.3081 for one on 12 vertices. On the other hand, for a fixed number of vertices, graphs with larger girth have higher algebraic connectivity. For example, consider all quartic graphs on 13 vertices. For those with girth 3, their minimum algebraic connectivity is 0.2679 while for those with girth 4, it is 1.6038. Hence, this implies a relationship between girth and algebraic connectivity.

It is also worth mentioning that some graphs in the table have equal minimum and maximum algebraic connectivity. This is only the case when there is a unique graph being considered. As the number of graphs grows exponentially, the table only summarizes graphs of order 5 to 35 and of girth 3 to 6.

In the next section, we provide a two term asymptotic estimate for an eigenvalue of the Laplacian matrix of specific cubic graphs. We then numerically show that this estimate applies to their algebraic connectivity.

Algebraic Connectivity of Quartic Graphs Based on Girth and Order			
Girth	Order	Minimum AC	Maximum AC
3	5	5	5
	6	4	4
	7	3	3.1981
	8	2	2.7639
	9	1.4384	3
	10	0.6278	2.5859
	11	0.3542	2.4679
	12	0.3081	2.4384
	13	0.2679	2.3285
4	8	4	4
	10	2.7639	3
	11	2.6021	2.3542
	12	2	3
	13	1.6038	2.6228
	14	1	2.5858
	15	0.7458	2.3820
	16	0.3542	2.4384
5	19	2.2087	2.2087
	20	2	2.2679
	21	1.9733	2.0743
	22	1.7625	2
	23	1.6232	2
6	26	2.2679	2.2679
	28	2	2
	30	1.7639	2
	32	1.5742	2
	34	1.4116	1.6972
	35	1.4011	2

Table 2.3: Minimum and maximum algebraic connectivity of quartic graphs based on girth and order.

2.2 A Two Term Asymptotic Estimate

As in the previous section, we consider the root-connected perfect binary tree family T_K, where K is the height of each tree. This time, however, we insist on a particular leaf configuration. To do so, we first label the root vertex of the right subtree of the binary tree with root r_1 by u_1 and the root vertex of its left subtree by v_1. Similarly, we label the root vertex of the left subtree of the binary tree with root r_2 by u_2 and the root vertex of its right subtree by v_2. In the configuration of interest, we only connect the leaf vertices of the subtree with root u_1 to the leaf vertices of the subtree with root u_2. Likewise, we only connect the leaf vertices of the subtree with root v_1 to the leaf vertices of the subtree with root v_2. We say that a cubic graph with this configuration of the leaf vertices is formed by the T_K graph. Figure 2.4 provides an example of such a cubic graph, in this case formed by the T_3 graph.

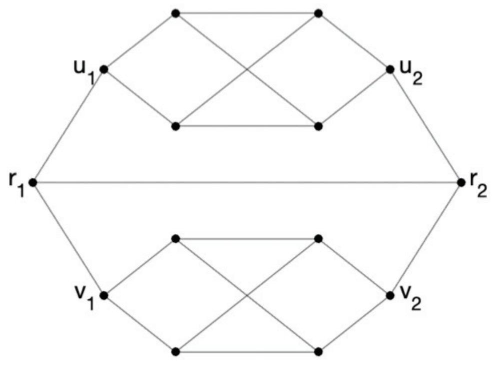


Figure 2.4: The cubic graph formed by the T_3 graph.

The next theorem provides an asymptotic estimate for an eigenvalue of their Laplacian matrix. Table 2.4 suggests that this eigenvalue is the algebraic connectivity.

Theorem 2.2.1 *Let G be a cubic graph formed by the T_K graph. Then its Laplacian matrix has an eigenvalue λ such that $\lambda \sim 2^{-K}(1-2^{1-K})$ as $K \to \infty$. That is, the value of λ approaches $2^{-K}(1-2^{1-K})$ for large enough K.*

Proof. As an eigenvector can be represented as an assignment of weights to the vertices of G, we assign them as follows: for the root vertices, assign weight $x_1 = 0$; for the root vertices of the subtrees labelled u_1 and u_2, assign weight x_2; for the vertices at level k of the binary tree consisting of these subtrees, assign weight x_k; for the root vertices of the subtrees labelled v_1 and v_2, assign weight $-x_2$; for the vertices at level k of the binary tree consisting of these subtrees, assign weight $-x_k$. This is a valid choice for $x = (x_1, \ldots, x_n)^T$ since the sum of its entries is zero and so this vector is orthogonal to the all-ones vector.

We now derive the equations for λx_i. Note that the root vertex r_1 with weight x_1 is adjacent to r_2 also with weight x_1, to the vertex u_1 with weight x_2, and to vertex u_2 with weight $-x_2$. For the root vertices, we then have that

$$\lambda x_1 = (x_1 - x_1) + (x_1 - x_2) + (x_1 - (-x_2)). \tag{2.22}$$

Since the root vertices have weight $x_1 = 0$, equation (2.22) simplifies to

$$\lambda 0 = (0 - x_2) + (0 - (-x_2)) \tag{2.23}$$

which simply yields $0 = 0$. Equation (2.22) can thus be omitted in the eigenvalue problem in (2.27).

Next, note that the vertex labelled u_1 has weight x_2 and is adjacent to the root r_1 with weight 0 and to two vertices at level 3 with weight x_3. Up to a factor of -1, we can then write

$$\lambda x_2 = (x_2 - 0) + 2(x_2 - x_3) \tag{2.24}$$

For $k = 3, \ldots, K-1$, any vertex at level k with weight x_k is adjacent to a parent vertex with weight x_{k-1} and to two child vertices with weights x_{k+1}. We can then write that

$$\lambda x_k = (x_k - x_{k-1}) + 2(x_k - x_{k+1}),\ k = 3, \ldots, K-1 \tag{2.25}$$

The leaf vertices with weight x_K are adjacent to a parent vertex with weight x_{K-1} and to two other leaf vertices with weight x_K. We then have that

$$\lambda x_K = (x_K - x_{K-1}) + 2(x_K - x_K) \tag{2.26}$$

Equations (2.24), (2.25), and (2.26) can be combined to create the eigenvalue problem

$$\begin{cases} \lambda x_2 & = (x_2 - 0) + 2(x_2 - x_3) \\ \lambda x_k & = (x_k - x_{k-1}) + 2(x_k - x_{k+1}),\ k = 3, \ldots, K-1; \\ \lambda x_K & = (x_K - x_{K-1}) + 2(x_K - x_K). \end{cases} \tag{2.27}$$

These coefficients can then be used to write the J x J matrix

$$M = \begin{bmatrix} 3 & -2 & & & & \\ -1 & 3 & -2 & & & \\ & -1 & \ddots & \ddots & & \\ & & \ddots & 3 & -2 & \\ & & & -1 & 3 & -2 \\ & & & & -1 & 1 \end{bmatrix} \tag{2.28}$$

where $J = K - 1$. As before, this matrix was obtained by considering the coefficients of the weights in the eigenvalue problem. By letting $d = 3$, we can note that the second to the second-to-last rows of M are the same as those of the tridiagonal matrix considered in (2.8) of Theorem 2.1.5. As equation (2.6) was satisfied by $\mu = d - (d-1)z - \frac{1}{z}$, equation (2.25) will be satisfied by

$$\lambda = 3 - 2z - \frac{1}{z} \tag{2.29}$$

for some z to be determined. Our ansatz for the eigenvector will then also remain the same. Namely, our ansatz is that

$$x_k = Az^k + B\left(\frac{1}{2z}\right)^k \tag{2.30}$$

for $k = 1, \ldots, K$ and non-zero A and B.

Letting $j = k - 1$, equation (2.25) may also be rewritten as

$$\lambda x_j = 3x_j - 2x_{j+1} - x_{j-1},\ j = 2, \ldots, J-1. \tag{2.31}$$

Then, rewriting (2.24) in terms of (2.31) yields

$$\lambda x_1 = (3x_1 - 2x_2 - x_0) + x_0$$

and rewriting (2.26) in terms of (2.31) results in

$$\lambda x_J = (3x_J - 2x_{J+1} - x_{J-1}) - 2x_J + 2x_{J+1}.$$

From (2.31), we then obtain that

$$x_0 = 0 \text{ and } x_{J+1} - x_J = 0$$

Now, by (2.30), these become

$$A + B = 0 \text{ and } Az^{J+1} + B\left(\frac{1}{2z}\right)^{J+1} - Az^J - B\left(\frac{1}{2z}\right)^J = 0$$

As a matrix equation, these correspond to

$$\begin{bmatrix} 1 & 1 \\ z^{J+1} - z^J & (\frac{1}{2z})^{J+1} - (\frac{1}{2z})^J \end{bmatrix} \begin{pmatrix} A \\ B \end{pmatrix} = 0$$

where $\begin{pmatrix} A \\ B \end{pmatrix}$ belongs to the kernel of the matrix. As a non-zero vector is in its kernel, the 2×2 matrix must be singular with a determinant of zero. Taking its determinant, we obtain the polynomial equation

$$1 - 2z - 2^{J+1}z^{2J+2} + 2^{J+1}z^{2J+1} = 0$$

or, factoring out $2^{J+1}z^{2J+1}$ from the last two terms,

$$1 - 2z - 2^{J+1}z^{2J+1}(z-1) = 0. \tag{2.32}$$

As this polynomial is not further factorable, z cannot be isolated and so we use asymptotics to further simplify it. On graphs with many levels, with large values of J and K, Figure 2.5 shows that z approaches both $\frac{1}{2}$ and 1. Furthermore, note that each polynomial has a root at $z = \frac{\sqrt{2}}{2}$ corresponding to the previously mentioned zero eigenvector obtained from letting $A = -B$. Also note that with $K = 2$, all other values of z are complex while some of them are real for $K \geq 3$. In turn, we will see that the eigenvalue tends exponentially toward zero. We use asymptotics to determine this decay rate.

We consider the case of $z = 1 - \epsilon$ for some small $\epsilon > 0$. Note that by substituting $z = 1 - \epsilon$ into equation (2.29), we obtain that

$$\lambda = \epsilon - 2\epsilon^2 \tag{2.33}$$

For z near 1, we can approximate equation (2.32) by

$$1 - 2z - 2^{J+1}(z-1) \sim 0. \tag{2.34}$$

Now, substituting $1 - \epsilon$ in for z, equation (2.34) becomes

$$1 - 2(1-\epsilon) - 2^{J+1}((1-\epsilon)-1) \sim 0.$$

Once simplified, this provides

$$\epsilon = \frac{1}{2^{J+1}+2}$$

and so $\epsilon \sim 2^{-(J+1)}$. As we let $J = K-1$, $K = J+1$, and so we obtain that $\epsilon \sim 2^{-K}$. Using this value for ϵ in equation (2.33) results in $\lambda \sim 2^{-K}(1-2^{1-K})$. ∎

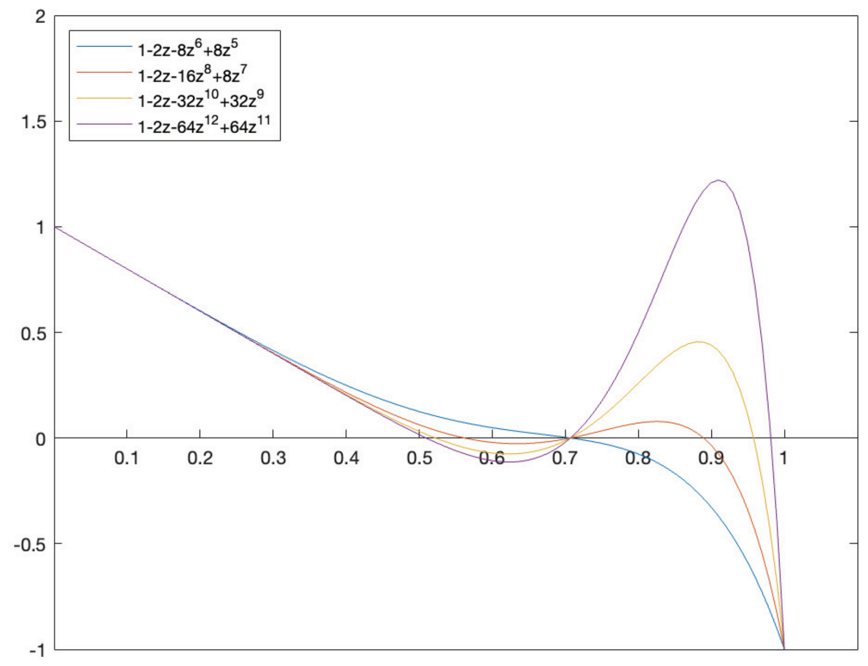


Figure 2.5: Behaviour of the function $1 - 2z - 2^{J+1}z^{2J+2} + 2^{J+1}z^{2J+1}$ for $J = 2, \ldots, 5$. Note the root at $z = \frac{\sqrt{2}}{2}$ and that the other roots approach both $\frac{1}{2}$ and 1 as $J \to \infty$.

Numerical Values and Asymptotic Estimates for Cubic Graphs							
Height (K Value)	3	4	5	6	7	8	9
Algebraic Connectivity	0.2679	0.0968	0.0403	0.0181	0.0085	0.0041	0.0020
$2^{-K}(1-2^{1-K})$	0.0938	0.0547	0.0293	0.0151	0.0077	0.0039	0.0019

Table 2.4: Numerical values and asymptotic estimates for the algebraic connectivity of cubic graphs formed by the T_K graph.

The algebraic connectivity of these graphs and the two term asymptotic estimate for different values of K are shown in Table 2.4. As can be seen in the table, the asymptotics provide a good estimate of the algebraic connectivity for $K \geq 6$. Note that both values tend to 0.

So far, we were interested in the algebraic connectivity of regular graphs having root-connected binary trees with the same number of levels as their subgraphs. In the next section, we provide an upper bound for the algebraic connectivity of cubic graphs having root-connected binary trees with trees differing by one level as their subgraphs.

2.3 A Second Upper Bound

Root-connected binary trees may have leaves at different levels. In this section, we focus on trees with leaves differing by one level. For example, the root-connected binary trees in Figure 2.6 consists of a tree rooted at r_1 with leaves on level 2 and a tree rooted at r_2 with leaves on level 3. Here, the roots are taken to be on level 0.

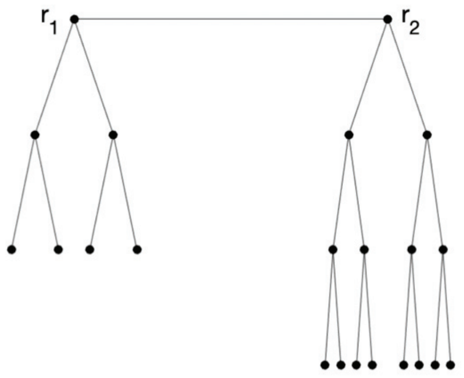


Figure 2.6: A root-connected binary tree with leaves on levels 2 and 3.

Alternatively, and as can be seen in Figure 2.7, we can describe these graphs as level binary trees having a root of degree 3. This form can be obtained by considering root r_1 as a child of root r_2. We refer to the single root by r and the roots of the three binary subtrees by u_1, u_2, and u_3. The height of each perfect binary tree is denoted by K. Based on this structure, we now provide the main theorem of this section.

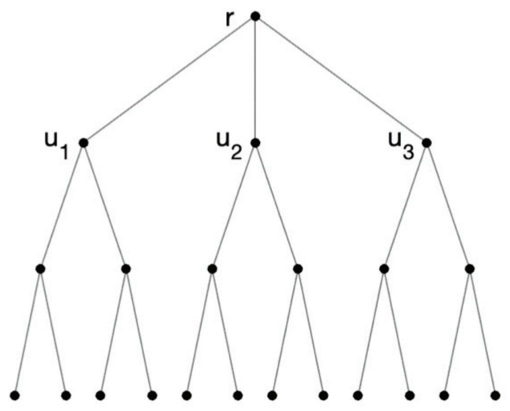


Figure 2.7: A level binary tree with a root of degree 3.

Theorem 2.3.1 *Let G be a cubic graph having a level binary tree with a root of degree 3 as a subgraph. Then $\lambda_2(G) \leq 3 - 2\sqrt{2}\cos(t/K)$, where K is the height of each tree and t is the smallest root of $\sin(t) + \sqrt{2}\sin\left(t + \frac{t}{K}\right)$ in the interval $\left(\frac{\pi}{2}, \pi\right)$.*

Proof. We begin by assigning weights to each vertex of G as follows: for the root vertex r, assign a weight of 0; for the vertices on level k, assign weight x_k; for all other vertices, assign a weight of 0. Multiply the weight of each vertex of the binary tree with root u_1 by $\bar{\omega} = e^{\frac{-2\pi i}{3}}$ and the vertices of the binary tree with root u_3 by $\omega = e^{\frac{2\pi i}{3}}$.

We now derive the equations for λx_i based on the structure of this graph. First note that the root vertex r with weight 0 is adjacent to the vertex labelled u_1 with weight $\bar{\omega}x_1$, the vertex labelled u_2 with weight x_1, and the vertex u_3 with weight ωx_1. We then have that

$$\lambda 0 = (0 - \bar{\omega}x_1) + (0 - x_1) + (0 - \omega x_1). \tag{2.35}$$

Equation (2.35) can be simplified to

$$x_1 + \omega x_1 + \bar{\omega}x_1 = 0. \tag{2.36}$$

Factoring out x_1 in equation (2.36), we obtain that

$$x_1(1 + \omega + \bar{\omega}) = 0 \tag{2.37}$$

and so that either $x_1 = 0$ or $\omega + \bar{\omega} = -1$. Indeed, since we let $\omega = e^{\frac{2\pi i}{3}}$, we have that $\omega + \bar{\omega} = 2\cos(\frac{2\pi}{3}) = -1$ from the complex definition of cosine. As equation (2.35) does not provide an equation for λ, it can be omitted from the eigenvalue problem.

Now consider any vertex at level 1 with weight x_1. This vertex is adjacent to the root vertex with weight 0 and to two vertices at level 2 with weight x_2. Its equation is then

$$\lambda x_1 = (x_1 - 0) + 2(x_1 - x_2). \tag{2.38}$$

The vertices at level 1 with weight $\bar{\omega}x_1$ will have the same equation up to a factor of $\bar{\omega}$. Similarly, those with weight ωx_1 will also satisfy equation (2.38) up to a factor of ω. We can thus represent all of the vertices at level 1 by equation (2.38).

Next, note that equations (2.3) and (2.25) still apply for the vertices at level $k = 2, \ldots, K-1$. That is, as the vertices at these levels of the binary tree are not adjacent to the root or to any leaves, we may write

$$\lambda x_k = (x_k - x_{k-1}) + 2(x_k - x_{k+1}),\ k = 2, \ldots, K-1 \tag{2.39}$$

where the parent vertex of the one with weight x_k has weight x_{k-1} and the two children of the vertex with weight x_k have weight x_{k+1}.

We now consider the leaf vertices of the binary trees. By connecting each leaf vertex to two other leaf vertices, one from each distinct tree, equation (2.40) can be simplified to contain only real coefficients. We refer to the graph obtained by this configuration of the leaf vertices by H, with corresponding $K \times K$ matrix M and eigenvalue μ. The equation for this configuration of the leaf vertices is then

$$\mu x_K = (x_K - x_{K-1}) + (x_K - \omega x_K) + (x_K - \bar{\omega} x_K) \tag{2.40}$$

as each leaf with weight x_K is adjacent to a parent vertex with weight x_{K-1}, a leaf vertex of the subtree rooted at u_3 with weight ωx_K, and a leaf vertex of the subtree rooted at u_1 with weight $\bar{\omega} x_K$. Equation (2.40) can be reduced to

$$\mu x_K = (x_K - x_{K-1}) + x_K(1 - \omega + 1 - \bar{\omega}) \tag{2.41}$$

by factoring out x_K from the last two terms of the equation. As $\omega + \bar{\omega} = -1$, equation (2.41) can be simplified to

$$\mu x_K = (x_K - x_{K-1}) + 3x_K. \tag{2.42}$$

Note that as G and H both contain the level binary trees with a root of degree 3 as a subgraph, we can substitute μ for λ in equations (2.38) and (2.39). Together with (2.42), these three equations can be combined to create the eigenvalue problem

$$\begin{cases} \mu x_1 = (x_1 - 0) + 2(x_1 - x_2) & (2.43) \\ \mu x_k = (x_j - x_{j-1}) + 2(x_j - x_{j+1}),\ k = 2...K-1 & (2.44) \\ \mu x_K = (x_K - x_{K-1}) + 3x_K. & (2.45) \end{cases}$$

As in the previous two proofs, we can consider equations (2.43), (2.44), and (2.45) as a system of equations with corresponding K x K matrix

$$M = \begin{bmatrix} 3 & -2 & & & & \\ -1 & 3 & -2 & & & \\ & -1 & \ddots & \ddots & & \\ & & \ddots & 3 & -2 & \\ & & & -1 & 3 & -2 \\ & & & & -1 & 4 \end{bmatrix}.$$

As the first to second-to-last rows have the same entries as the ones for the tridiagonal matrix in (2.28), equation (2.44) will also be satisfied by

$$\mu = 3 - 2z - \frac{1}{z} \tag{2.46}$$

for some z to be determined. Furthermore, our ansatz for the eigenvector will also remain the same. That is, we have

$$x_k = Az^k + B\left(\frac{1}{2z}\right)^k \tag{2.47}$$

for $k = 1, \ldots, K$ and non-zero A and B.

Similarly as in (2.31), equation (2.44) can be written as

$$\mu x_k = 3x_k - 2x_{k+1} - x_{k-1},\ k = 2, \ldots, K-1. \tag{2.48}$$

Then, writing equation (2.43) in terms of equation (2.48), we obtain

$$\mu x_1 = (3x_1 - 2x_2 - x_0) + x_0. \tag{2.49}$$

Similarly, equation (2.45) can be rewritten in terms of (2.48) as

$$\mu x_K = (3x_K - 2x_{K+1} - x_{K-1}) + x_K + 2x_{K+1}. \tag{2.50}$$

From (2.48), we then obtain that

$$x_0 = 0 \text{ and } x_K + 2x_{K+1} = 0. \tag{2.51}$$

Now, using equation (2.47), the two equations in (2.51) can be written as

$$A + B = 0 \text{ and } Az^K + B\left(\frac{1}{2z}\right)^K + 2Az^{K+1} + 2B\left(\frac{1}{2z}\right)^{K+1} = 0. \tag{2.52}$$

Putting the equations in (2.52) in matrix form results in

$$\begin{bmatrix} 1 & 1 \\ z^K + 2z^{K+1} & (\frac{1}{2z})^K + 2(\frac{1}{2z})^{K+1} \end{bmatrix} \begin{pmatrix} A \\ B \end{pmatrix} = 0 \tag{2.53}$$

where $\begin{pmatrix} A \\ B \end{pmatrix}$ belongs to the kernel of the matrix. As a non-zero vector belongs in its kernel, the 2×2 matrix must be singular with a determinant of zero. Taking its determinant, we obtain the polynomial equation

$$1 + z - 2^K z^{2K+1} - 2^{K+1} z^{2K+2} = 0. \tag{2.54}$$

Making the substitution $z = \frac{\sqrt{2}}{2}e^{i\theta}$ and simplifying equation (2.54) yields the polynomial

$$e^{Ki\theta} - e^{-Ki\theta} - \sqrt{2}e^{-i\theta(K+1)} + \sqrt{2}e^{i\theta(K+1)} = 0 \tag{2.55}$$

Letting $t = K\theta$ in equation (2.55) results in

$$e^{ti} - e^{-ti} - \sqrt{2}e^{-i(t+\frac{t}{K})} + \sqrt{2}e^{i(t+\frac{t}{K})} = 0. \tag{2.56}$$

Recall that the complex definition of the sine function is given by $\sin(t) = \frac{e^{it}-e^{-it}}{2i}$ for any t. Equation (2.56) can then be rewritten as

$$\sin(t) + \sqrt{2}\sin\left(t + \frac{t}{K}\right) = 0. \tag{2.57}$$

Note that the function in (2.57) is continuous for $K > 0$. For $K > 1$, it is also positive for $t = \frac{\pi}{2}$ and negative for $t = \pi$. To see this, first recall that $\sin\left(t + \frac{t}{K}\right) = \sin(t)\cos\left(\frac{t}{K}\right) + \cos(t)\sin\left(\frac{t}{K}\right)$ by a trigonometric identity. Then, we can write equation (2.57) as

$$\sin(t) + \sqrt{2}\left(\sin(t)\cos\left(\frac{t}{K}\right) + \cos(t)\sin\left(\frac{t}{K}\right)\right) = 0. \tag{2.58}$$

For $t = \frac{\pi}{2}$, the function in (2.58) simplifies to $1 + \sqrt{2}\cos\left(\frac{\pi}{2K}\right)$. This value is positive for $K \geq 1$, with $1 + \sqrt{2}\cos\left(\frac{\pi}{2K}\right) \sim 1 + \sqrt{2}$ as $K \to \infty$ and a minimum of 1 when $K = 1$. For $t = \pi$, the function in (2.58) simplifies to $-\sqrt{2}\sin\left(\frac{\pi}{K}\right)$. This value is negative for $K \geq 1$, with $-\sqrt{2}\sin\left(\frac{\pi}{K}\right) \sim 0$ as $K \to \infty$ and a minimum of $-\sqrt{2}$ when $K = 2$. Also note that $t = \pi$ is a root when $K = 1$. Hence, by the Intermediate Value Theorem, the function in (2.57) has a root in the interval $(\frac{\pi}{2}, \pi)$.

The bound is obtained after simplification by plugging in $z = \frac{\sqrt{2}}{2}e^{i\theta}$ with $\theta = \frac{t}{K}$ into equation (2.46). ■

We have now provided an upper bound for the algebraic connectivity of cubic graphs having level binary trees of height K with a root of degree 3 as subgraphs. Some of the upper bound's values are listed in Table 2.5.

Second Upper Bound Values for the Algebraic Connectivity of Cubic Graphs								
Height (K Value)	2	3	4	5	6	7	8	9
Smallest Root t	2.4189	2.6242	2.7383	2.8110	2.8615	2.8985	2.9269	2.9494
Upper Bound Value	2	1.1864	0.8088	0.6069	0.4872	0.4106	0.3588	0.3221

Table 2.5: Upper bound values for the algebraic connectivity of cubic graphs having level binary trees with a root of degree 3 as subgraphs.

When compared to the upper bound values in Table 2.1 with $d = 3$, we can see that these values are only slightly lower. However, this is to be expected since these trees can be thought of as differing by only one level.

We also note that the bound is attained by the Petersen graph when $K = 2$. This graph is a (3,5)-cage and contains a level binary tree with a root of degree 3 as a subgraph. To find that the algebraic connectivity of the Petersen graph achieves the bound of Theorem 2.3.1, a similar computational search as was done for quartic graphs in the first section was performed. This time, it was accomplished on cubic graphs of order 4 to 62 and with girth 3 to 9. Its results can be found in Table 2.6.

After some examination, we conclude that the bound is not attained when $K = 3$. In fact, the largest possible algebraic connectivity amongst all graphs of order 22 or greater was 1. As the level binary tree with height 3 has order 22, any graph with this subgraph must either have the same order or contain more vertices. This bound is thus also unlikely to be achieved for higher values of K.

This table also confirms our observations on quartic graphs. Namely, for fixed girth, both the minimum and maximum algebraic connectivity quickly decrease as the number of vertices increases. Furthermore, as previously noted for graphs with fixed order, those with larger girth have higher algebraic connectivity.

Algebraic Connectivity of Cubic Graphs Based on Girth and Order			
Girth	Order	Minimum AC	Maximum AC
3	4	4	4
	6	2	2
	8	0.763932	1.438447
	10	0.221542	1.120614
	12	0.167742	1
	14	0.104893	0.885092
	16	0.084042	0.822590
	18	0.062022	0.763932
4	6	3	3
	8	2	2
	10	1	1.438447
	12	0.438447	1.267949
	14	0.144227	1.068732
	16	0.121938	1
	18	0.100085	0.903097
	20	0.067316	0.845793
5	10	2	2
	12	1.438447	1.467911
	14	1.133802	1.289171
	16	0.763932	1.172909
	18	0.438447	1.043705
	20	0.221543	1
	22	0.082714	0.913969
6	14	1.585786	1.585786
	16	1.267949	1.267949
	18	1	1.267949
	20	0.829914	1.064568
	22	0.653484	1
	24	0.438447	0.999999
7	24	0.999999	0.999999
	26	0.913870	0.947370
	28	0.753020	0.999999
	30	0.664749	0.844084
	32	0.585786	0.864221
8	30	0.999999	0.999999
	34	0.785680	0.785680
	36	0.737450	0.763932
	38	0.646100	0.763932
	40	0.585786	0.763932
9	58	0.540470	0.637660
	60	0.485863	0.697224
	62	0.481866	0.603671

Table 2.6: Minimum and maximum algebraic connectivity of cubic graphs based on girth and order.

Chapter 3

Algebraic Connectivity of Necklace Graphs

We begin this chapter with the definition of a necklace graph.

Definition 3.0.1 *Let G be a connected graph with two labelled vertices, say u and v. Then the necklace graph $N_{G,c}$ is the graph composed of c copies of G, say $G_1, \ldots, G_c$, with labelled vertices $u_1, \ldots, u_c$ and $v_1, \ldots, v_c$, and where, for $i = 1, \ldots, c-1$, vertex u_i is made adjacent to vertex v_{i+1}. To close the necklace, create an edge between vertex u_c and vertex v_1.*

As illustrated in Figure 3.1, the necklace graphs with fewest vertices are cycles of even order. These are 2-regular graphs that contain copies of the complete graph K_2. The number of vertices in these cycles can be determined by the number of copies. In fact, as each copy provides two vertices, the cycle's order will be twice the number of copies. That is, a cycle containing c copies will have order $2c$. As mentioned in Chapter 2, their algebraic connectivity is given by $\lambda_2(C_n) = 2(1 - \cos(\frac{2\pi}{n}))$. In this chapter, the first section establishes results for general necklace graphs while the second introduces regular necklace graphs.

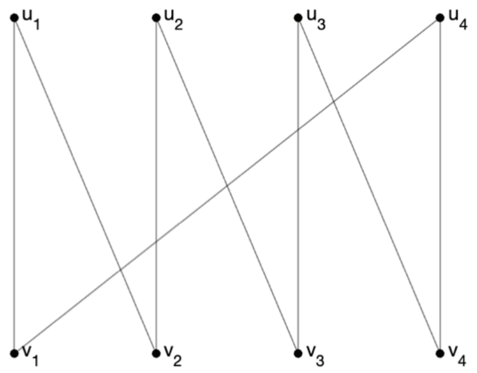


Figure 3.1: The necklace graph $N_{K_2,4}$ composed of 4 copies of the complete graph K_2 is a cycle of length 8.

3.1 General Necklace Graphs

A clear relationship exists between the starting graph G and the resulting necklace graph $N_{G,c}$. In fact, the choice of G and the number of its copies determines a unique necklace graph. That is, different starting graphs, or a different number of copies of the same graph, create different necklace graphs. Furthermore, if G is a graph on n vertices and with c copies, then its necklace graph will have order nc. For example, 3 copies of a graph of order 4 will result in a necklace graph of order 12. A relationship between their Laplacian matrices should then also exist. The following theorem provides this correspondence.

Theorem 3.1.1 *Let G be a graph with n vertices and L_G be its $n \times n$ Laplacian matrix. Let L_N be the $nc \times nc$ Laplacian matrix of the necklace graph $N_{G,c}$ and let $\omega = e^{\frac{2\pi ki}{c}}$, $k = 1, \ldots, c$. For each k, let the $n \times n$ matrix L_ω be given by*

$$L_\omega = L_G + \begin{bmatrix} 1 & -\omega & 0 & \ldots & 0 \\ -\bar{\omega} & 1 & 0 & \ldots & 0 \\ 0 & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 0 \end{bmatrix}$$

where the last matrix also has dimensions $n \times n$. Then, λ is an eigenvalue of L_ω for some k if and only if it is an eigenvalue of L_N. That is, the spectrum of L_N is given by the union with multiplicity of the spectra of L_ω over all k.

Proof. For $i = 1, \ldots, n$ and $j = 1, \ldots, c$, assign weight $x_{i,j}$ to the i^{th} vertex of the j^{th} copy of G in the necklace graph. The vertices can be chosen in any order, provided that each vertex labelled u is given weight $x_{1,j}$, each vertex labelled v is given weight $x_{2,j}$, and the labelling of the copies is consistent. Let $\Big(x_1, \ldots, x_n\Big)^T$ be an eigenvector of L_ω for some k and let λ be its corresponding eigenvalue. Then, by definition, we have that

$$\lambda \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = L_\omega \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}. \tag{3.1}$$

Using the structure of L_ω, note that equation (3.1) can be rewritten as the system

$$\begin{cases} \lambda x_1 = \sum_{(1,e)\in E(G)}(x_1 - x_e) + (x_1 - \omega x_2) \\ \lambda x_2 = \sum_{(2,f)\in E(G)}(x_2 - x_f) + (x_2 - \bar{\omega} x_1) \\ \lambda x_i = \sum_{(i,g)\in E(G)}(x_i - x_g) & \text{for } i = 3, \dots, n. \end{cases} \tag{3.2}$$

Our ansatz, or educated guess for the structure of the eigenvector of the Laplacian matrix of the necklace graph, is that

$$x_{i,j} = \omega^{j-1} x_i \tag{3.3}$$

for $j = 1, \dots, c$. Multiplying each equation in system (3.2) by ω^{j-1} yields the system

$$\begin{cases} \omega^{j-1}\lambda x_1 = \omega^{j-1}(\sum_{(1,e)\in E(G)}(x_1 - x_e) + (x_1 - \omega x_2)) \\ \omega^{j-1}\lambda x_2 = \omega^{j-1}(\sum_{(2,f)\in E(G)}(x_2 - x_f) + (x_2 - \bar{\omega} x_1)) \\ \omega^{j-1}\lambda x_i = \omega^{j-1}\sum_{(i,g)\in E(G)}(x_{i,j} - x_{g,j}) & \text{for } i = 3, \dots, n, \end{cases} \tag{3.4}$$

which, by our ansatz, is equivalent to

$$\begin{cases} \lambda x_{1,j} = \sum_{(1,e)\in E(G_j)}(x_{1,j} - x_{e,j}) + (x_{1,j} - x_{2,j+1}) \\ \lambda x_{2,j} = \sum_{(2,f)\in E(G_j)}(x_{2,j} - x_{f,j}) + (x_{2,j} - x_{1,j-1}) \\ \lambda x_{i,j} = \sum_{(i,g)\in E(G_j)}(x_{i,j} - x_{g,j}) & \text{for } i = 3, \dots, n. \end{cases} \tag{3.5}$$

Note that, for each copy G_j in the necklace graph, vertex u_j has weight $x_{1,j}$ and is adjacent to vertex v_{j+1} with weight $x_{2,j+1}$ in copy G_{j+1}. Furthermore, vertex v_j has weight $x_{2,j}$ and is adjacent to vertex u_{j-1} with weight $x_{1,j-1}$ in copy G_{j-1}. Finally, note that the vertices with weights $x_{3,j}, \dots, x_{n,j}$ are only adjacent to vertices in their own copy. Thus, for $1 \le j \le c$, the system of equations in (3.5) represents the structure of copy G_j in the necklace graph. By definition, the vertex set of the necklace graph is the union of the vertex sets of its copies $G_1, \dots, G_c$. Hence, the Laplacian matrix L_N can be obtained by considering the union of the equations in system (3.5) over all values of j. It may then also be rewritten by the equation

$$\lambda \begin{pmatrix} x_{1,1} \\ \vdots \\ x_{n,c} \end{pmatrix} = L_N \begin{pmatrix} x_{1,1} \\ \vdots \\ x_{n,c} \end{pmatrix},$$

and so λ must be an eigenvalue of L_N. Furthermore, the weight vector $(x_{1,1},\ldots,x_{n,c})^T$ consisting of the weights of all vertices in the necklace graph must be an eigenvector.

We now prove the converse. That is, suppose that λ is an eigenvalue of L_N. From the first part, we know that the eigenvalues of L_ω for some k are also eigenvalues of L_N. As each L_ω matrix has dimensions $n \times n$ and there are c distinct L_ω matrices, one for each value of k, there are precisely nc eigenvalues in the union with multiplicity of the spectra of L_ω over all k. As the spectrum of L_N also has nc eigenvalues, λ must be an eigenvalue of L_ω for some k. ■

This concept is particularly useful when considering necklaces with many copies. As each copy contributes n eigenvalues, the necklace graph will contain nc eigenvalues in total while L_ω will only contain n for each k. Thus, considering those of L_ω itself may be more efficient when determining the eigenvalues of necklace graphs. Under certain conditions, we now show that the eigenvector formed by ansatz (3.3) is indeed a Fiedler vector. To do so, we refer back to Lemma 1.0.1.

By this lemma, an eigenvector which partitions a graph into two blocks and which contains only non-zero components might be a Fiedler vector. Furthermore, we can use its contrapositive to prove that all other eigenvectors are not Fiedler vectors. That is, if an eigenvector contains only non-zero entries and separates the graph into more than two blocks, then it is not a Fiedler vector. Thus, as the second smallest eigenvalue of the Laplacian matrix is unique up to multiplicity, the first eigenvector must indeed be a Fiedler vector. We now introduce the second theorem of this section.

Theorem 3.1.2 *Let G be a connected graph of order n and let $N_{G,c}$ be its necklace graph with c copies. Let $x_{i,j}$ denote the weight of the i^{th} vertex, $1 \le i \le n$, of the j^{th} copy, $1 \le j \le c$, of G. Furthermore, let $\omega = e^{\frac{2\pi k i}{c}}$, $k = 1,\ldots,c$, denote a c^{th} root of unity. Finally, let $(x_1,\ldots,x_n)^T$ be an eigenvector of the previously defined L_ω matrix such that $x_i > 0$ for all i. If $x_{i,j} = \omega^{j-1}x_i$ and $k = 1$, then the eigenvector Φ_ω composed of these entries is a Fiedler vector.*

Proof. As the vertices of each copy are multiplied by the same power of ω, we may write the eigenvector of the necklace graph as

$$\Phi_\omega = \begin{pmatrix} x_1 & \ldots & x_n & \omega x_1 & \ldots & \omega x_n & \ldots & \omega^{c-1}x_1 & \ldots & \omega^{c-1}x_n \end{pmatrix}^T.$$

Note that this eigenvector is complex as it contains a root of unity. As complex eigenvectors come in conjugate pairs, the eigenvector associated with $\bar{\omega}$, which we denote by $\Phi_{\bar{\omega}}$, is also an eigenvector. Now, recall that all of the eigenvalues of the Laplacian matrix are real. This implies that Φ_ω and $\Phi_{\bar{\omega}}$ are eigenvectors for the same real eigenvalue. Then, their sum, $\Phi = \Phi_\omega + \Phi_{\bar{\omega}}$, is a linear combination of two eigenvectors in the same eigenspace and thus also an eigenvector. For this eigenvector, observe that

$$\Phi = \begin{pmatrix} 2x_1 & \dots & 2x_n & \dots & 2\cos(\frac{2\pi k(c-1)}{c})x_1 & \dots & 2\cos(\frac{2\pi k(c-1)}{c})x_n \end{pmatrix}^T$$

and that each repetition of $x_1, \dots, x_n$ in the vector represents a distinct copy in the necklace graph. We may thus represent each copy by

$$\Phi_j = 2\begin{pmatrix} x_1\cos(\frac{2\pi k(j-1)}{c}) & \dots & x_n\cos(\frac{2\pi k(j-1)}{c}) \end{pmatrix} \text{ for } j = 1, \dots, c.$$

The component of Φ corresponding to vertex i in copy j can then also be represented by

$$\Phi_{i,j} = 2x_i \cos\left(\frac{2\pi k(j-1)}{c}\right).$$

As all of the x_i's are positive, the sequence of $\Phi_{i,j}$'s in Φ changes sign if and only if $\cos(\frac{2\pi k(j-1)}{c})$ does. Furthermore, as all of the x_i's are non-zero, $\Phi_{i,j}$ equals zero if and only if $\cos(\frac{2\pi k(j-1)}{c}) = 0$. This occurs when

$$j = \frac{c+4k}{4k} + \frac{ct}{2k} \tag{3.6}$$

for some $t = 0, \dots, 2k-1$ and can be obtained by setting $\frac{2\pi k(j-1)}{c} = \frac{\pi}{2} + \pi t$.

Now, recall that j must be a natural number since it represents the index of a copy in the necklace graph. For $k = 1$, the sequence of $\Phi_{i,j}$'s changes sign twice. Namely, for $t = 0$ and $t = 1$, it changes from being positive when $j = 1, \dots, \lfloor\frac{c+4}{4}\rfloor$ to being negative when $j = \lceil\frac{c+4}{4}\rceil, \dots, \lfloor\frac{c+4}{4} + \frac{c}{2}\rfloor$ to being positive again when $j = \lceil\frac{c+4}{4} + \frac{c}{2}\rceil, \dots, c$. Together, these values of j partition the necklace graph into exactly two blocks. The first block consists of the positive values of cosine while the second consists of its negative values. For $k \geq 2$, $\cos(\frac{2\pi k(j-1)}{c})$ changes sign more than twice and thus Φ partitions the necklace graph into more than 2 blocks.

Note that if $k = 1$ and c is a multiple of 4, equation (3.6) results in integer values for j for which $\Phi_{i,j} = 0$. Namely, if $c = 4p$ for some natural number p, then equation

(3.6) reduces to the cases $j = (p+1)$ for $t = 0$ and $j = 3p+1$ for $t = 1$. In this case, Φ would contain some zero entries and by Lemma 1.0.1 may not separate the graph into two blocks. Thus, it may not be the Fiedler vector. In all other cases, Φ contains only non-zero entries and thus the choice $k = 1$ corresponds to the Fiedler vector.

In the case that c is a multiple of 4, we begin by considering the linear combination $\hat{\Phi} = \Phi_\omega - \Phi_{\bar{\omega}}$. In this case, we obtain the vector

$$\hat{\Phi} = \begin{pmatrix} 2x_1 & \dots & 2x_n & \dots & 2\sin(\frac{2\pi k(c-1)}{c})x_1 & \dots & 2\sin(\frac{2\pi k(c-1)}{c})x_n \end{pmatrix}^T$$

and so that

$$\hat{\Phi}_{i,j} = 2x_i \sin\left(\frac{2\pi k(j-1)}{c}\right) \text{ for } i = 1, \dots, n \text{ and } j = 1, \dots, c.$$

Like before, the structure of $\hat{\Phi}$ only depends on the expression $\sin(\frac{2\pi k(j-1)}{c})$. This time, we obtain that $\sin(\frac{2\pi k(j-1)}{c}) = 0$ if and only if

$$j = \frac{ct + 2k}{2k}$$

for $t = 0, \dots, 2k-1$ by setting $\frac{2\pi k(j-1)}{c} = \pi t$.

By a similar argument, the choice $k = 1$ for $\hat{\Phi}$ results in a Fiedler vector if and only if c is not a multiple of 2. Yet, $\Phi_{i,j}$ and $\hat{\Phi}_{i,j}$ are never equal to zero for the same value of j. To see this, note that

$$\frac{c+4k}{4k} + \frac{ct}{2k} = \frac{ct+2k}{2k}$$

if and only if $c = 0$, which is impossible since c denotes the number of copies in the necklace graph. Thus, we may shift each entry in Φ by some small $\epsilon > 0$ to get the linear combination $\Phi^* = \Phi + \epsilon\hat{\Phi}$.

As ϵ is small, the signs in the sequence of $\Phi^*_{i,j}$'s in Φ^* will mostly depend on the value of $\cos(\frac{2\pi k(j-1)}{c})$. Furthermore, no previously non-zero entry will become zero for small enough ϵ. This adjustment thus results in an eigenvector with all non-zero entries partitioning the necklace graph into exactly two blocks when $k = 1$ regardless of the number of copies. Thus, as it was obtained by a linear combination of eigenvectors in the same eigenspace, this vector is also a Fiedler vector for $k = 1$.

Note that it may be written as $\Phi^* = (1+\epsilon)\Phi_\omega + (1-\epsilon)\Phi_{\bar{\omega}}$ and thus Φ_ω and $\Phi_{\bar{\omega}}$ are Fiedler vectors for $k = 1$ as well. ■

Combined with Theorem 3.1.1, this theorem allows us to quickly determine the algebraic connectivity of any necklace graph. That is, since its Fiedler vector corresponds to the choice $k = 1$, its algebraic connectivity will be equal to the smallest eigenvalue of the $n \times n$ matrix L_ω with $k = 1$. Furthermore, the zero eigenvalue can be obtained by the smallest eigenvalue of L_ω with $k = c$. This comes directly from the fact that $\omega = 1$ when $k = c$ and so that L_ω is the Laplacian matrix of the graph $G + e$. Regular necklace graphs are introduced in the next section.

3.2 Regular Necklace Graphs

We define a d-regular necklace graph as being a necklace graph in which every vertex has degree d. These graphs can be constructed from c copies of d-regular graphs with one edge fewer, namely the edge $\{u_i, v_i\}$ for $i = 1, \ldots, c$. By omitting this edge, every other vertex has degree d while the vertices u and v have degree $d - 1$. Nevertheless, as these vertices are made adjacent to vertices in other copies to form the necklace graph, the resulting necklace graph is d-regular. We now propose a family of regular necklace graphs, which can be obtained by following these steps:

1. Begin with c copies of the complete graph K_{d-1}.
2. For each copy, create two new vertices labelled u and v.
3. Join the newly created vertices to each vertex of K_{d-1}.
4. Create the necklace graph by following the definition.

In this family, necklaces are composed of copies of $K_{d+1} \backslash \{u, v\}$. That is, of complete graphs on $d + 1$ vertices with the edge $\{u, v\}$ being removed. For example, consider the diamond graph D obtained from the complete graph K_4 by removing an edge. Figure 3.2 illustrates the diamond necklace graph $N_{D,5}$ with 5 copies. Note that this graph is indeed 3-regular. Furthermore, cycles of order $3c$ can be obtained by considering copies of $K_3 \backslash \{u, v\}$.

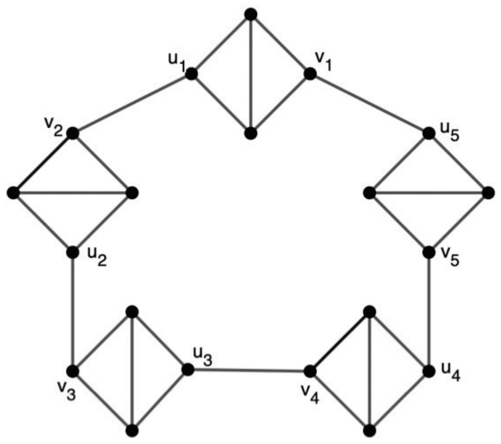


Figure 3.2: The necklace graph $N_{D,5}$ composed of 5 copies of the diamond graph D.

The next theorem provides the complete spectrum of their Laplacian matrix.

Theorem 3.2.1 *Let $G = K_{d+1}\backslash\{u, v\}$ be a graph of order $n = d+1$ and let $N_{G,c}$ be its d-regular necklace graph with c copies. Also, let $\omega = e^{\frac{2\pi ki}{c}}$ for $k = 1, \ldots, c$. Then, the spectrum of the Laplacian matrix L_N consists of an eigenvalue of n with multiplicity $(n-3)c$ and of the $3c$ roots of the family of polynomials $\lambda(\lambda-n)^2+2(n-2)(\cos(\frac{2\pi k}{c})-1)$, with 3 roots for each value of k.*

Proof. From Theorem 3.1.1, the spectrum of L_N is the same as the union with multiplicity of the eigenvalues of L_ω, $\omega = e^{\frac{2\pi ki}{c}}$, over all $k = 1, \ldots, c$. Thus, we begin by determining the eigenvalues of L_ω for some k. In this case, this matrix has dimensions $n \times n$ and can be written as follows:

$$L_\omega = \begin{bmatrix} d & -\omega & -1 & -1 & \ldots & -1 \\ -\bar{\omega} & d & -1 & -1 & \ldots & -1 \\ -1 & -1 & d & -1 & \ldots & -1 \\ -1 & -1 & -1 & \ddots & \ldots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & -1 \\ -1 & -1 & -1 & \ldots & -1 & d \end{bmatrix}.$$

We first show that this matrix contains an eigenvalue of n with multiplicity $(n-3)$. Our ansatz for the eigenvector is as follows: for the vertices labelled u and v, assign a weight of zero; for all other vertices, assign different weights, say $x_3, \ldots, x_n$, such that $x_3 + \cdots + x_n = 0$. Note that this is an equation in $(n-2)$ variables which has $(n-3)$ independent solutions of the form $x_3 = -1$, $x_i = 1$ for some i, $4 \leq i \leq n$, and 0 otherwise. We refer to these vectors by ν_i and show that they all correspond to an eigenvalue of n. Recall that to be an eigenvalue of L_ω, λ must satisfy the equation

$$L_\omega \nu_i = \lambda \nu_i. \tag{3.7}$$

In other words, we have that

$$\begin{bmatrix} d & -\omega & -1 & -1 & \dots & -1 \\ -\bar{\omega} & d & -1 & -1 & \dots & -1 \\ -1 & -1 & d & -1 & \dots & -1 \\ -1 & -1 & -1 & \ddots & \dots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & -1 \\ -1 & -1 & -1 & \dots & -1 & d \end{bmatrix} \begin{pmatrix} 0 \\ 0 \\ -1 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \lambda \begin{pmatrix} 0 \\ 0 \\ -1 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \tag{3.8}$$

where ν_i has a value of 1 in entry i. Then, equation (3.9) reduces to

$$\begin{pmatrix} 0 \\ 0 \\ -(d+1) \\ 0 \\ \vdots \\ 0 \\ d+1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -\lambda \\ 0 \\ \vdots \\ 0 \\ \lambda \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

so that $\lambda = d + 1$. Hence, as $d + 1 = n$, $\lambda = n$ regardless of the choice of i. As $(n-3)$ independent eigenvectors have this structure, L_ω contains an eigenvalue of

n with multiplicity $(n-3)$. Hence, as L_N consists of c copies of G, it contains an eigenvalue of n with multiplicity $(n-3)c$.

We now show that the spectrum of L_N also contains the $3c$ roots of the family of polynomials $\lambda(\lambda-n)^2+2(n-2)(\cos(\frac{2\pi k}{c})-1)$ over all k. Our second ansatz is as follows: for the vertex labelled u, assign weight x_1; for the vertex labelled v, assign weight x_2; for all other vertices in G, assign the same weight, say x_3. Since any eigenvalue must satisfy equation (3.8), we have that

$$\begin{bmatrix} d & -\omega & -1 & -1 & \dots & -1 \\ -\bar{\omega} & d & -1 & -1 & \dots & -1 \\ -1 & -1 & d & -1 & \dots & -1 \\ -1 & -1 & -1 & \ddots & \dots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & -1 \\ -1 & -1 & -1 & \dots & -1 & d \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_3 \end{pmatrix} = \lambda \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_3 \end{pmatrix}. \tag{3.9}$$

Equation (3.10) can then be represented by the system of equations

$$\begin{cases} \lambda x_1 & = dx_1 - \omega x_2 - (d-1)x_3 \\ \lambda x_2 & = -\bar{\omega} x_1 + dx_2 - (d-1)x_3 \\ \lambda x_3 & = -x_1 - x_2 + (d-(n-3))x_3 \end{cases} \tag{3.10}$$

where, using the fact that $n = d+1$, the equation for λx_3 can be simplified to $\lambda x_3 = -x_1 - x_2 + 2x_3$. Thus, λ is an eigenvalue of L_ω if and only if it is an eigenvalue of the 3×3 reduced matrix

$$L_R = \begin{bmatrix} d & -\omega & -(d-1) \\ -\bar{\omega} & d & -(d-1) \\ -1 & -1 & 2 \end{bmatrix}.$$

The exact values of x_1, x_2, and x_3 can also be determined from the system of equations

$$\begin{cases} (d-\lambda)x_1 - \omega x_2 - (d-1)x_3 & = 0 \\ -\bar{\omega} x_1 + (d-\lambda)x_2 - (d-1)x_3 & = 0 \\ -x_1 + x_2 + (2-\lambda)x_3 & = 0. \end{cases} \tag{3.11}$$

Through some algebra, it can be checked that this system yields

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} (2-\lambda) - \frac{d-1+2\bar{\omega}-\bar{\omega}\lambda}{\bar{\omega}+d-\lambda} \\ \frac{d-1+2\bar{\omega}-\bar{\omega}\lambda}{\bar{\omega}+d-\lambda} \\ 1 \end{pmatrix}.$$

To find an expression for λ, we now treat this problem as a standard eigenvalue problem. The characteristic polynomial of L_R is given by

$$\lambda^3 - 2(d+1)\lambda^2 + (d^2+2d+1)\lambda + (d-1)\omega + (d-1)\bar{\omega} - 2(d-1)$$

and, by factoring and using the complex cosine definition, can be rewritten as

$$\lambda(\lambda-(d+1))^2 + 2(d-1)(\cos\left(\frac{2\pi k}{c}\right) - 1). \tag{3.12}$$

This corresponds to a family of polynomials having a total of $3c$ roots, with 3 roots for each value of k. As $n = d+1$, we can rewrite it as

$$\lambda(\lambda-n)^2 + 2(n-2)(\cos\left(\frac{2\pi k}{c}\right) - 1). \tag{3.13}$$

■

From our results on general necklace graphs, we now show that the algebraic connectivity of this family of regular necklace graphs corresponds to the smallest root of equation (3.14) with $k = 1$. Furthermore, we provide a first term asymptotic estimate for an eigenvalue of these necklace graphs based on the number of copies. We then numerically show that this estimate is applicable to their algebraic connectivity.

Corollary 3.2.2 *Let $G = K_{d+1}\backslash\{u,v\}$ be a graph of order n and let $N_{G,c}$ be its d-regular necklace graph with c copies. Then its algebraic connectivity corresponds to the smallest root of $\lambda(\lambda-n)^2 + 2(n-2)(\cos\left(\frac{2\pi}{c}\right) - 1)$. Furthermore, the Laplacian matrix of $N_{G,c}$ has an eigenvalue λ such that $\lambda = \frac{4(d-1)\pi^2}{(d+1)^2c^2} + O(\frac{1}{c^4})$.*

Proof. We first show that the algebraic connectivity is given by the smallest root of the polynomial

$$\lambda(\lambda-n)^2 + 2(n-2)(\cos\left(\frac{2\pi}{c}\right) - 1), \tag{3.14}$$

corresponding to the expression in (3.14) with $k = 1$. Note that finding the roots of this expression is equivalent to finding the intersection points of the equation

$$\lambda(\lambda - n)^2 = -2(n-2)(\cos\left(\frac{2\pi k}{c}\right) - 1). \tag{3.15}$$

We first note the properties of $\lambda(\lambda - n)^2$ by considering it as a function of λ. This function is a cubic polynomial with three roots, a single root at $\lambda = 0$ and a double root at $\lambda = n$. Its derivative is given by $(\lambda - n)^2 + 2\lambda(\lambda - n)$, whose roots are $\lambda = \frac{n}{3}$ and $\lambda = n$. For $\lambda > 0$, this cubic is increasing for $0 < \lambda < \frac{n}{3}$, decreasing for $\frac{n}{3} < \lambda < n$, and increasing again for $\lambda > n$. It thus has a local maximum of $\frac{4n^3}{27}$ at $\lambda = \frac{n}{3}$ and a local minimum of 0 at $\lambda = n$.

As it does not depend on λ and the values of n and c are fixed by the necklace graph, we consider the expression $-2(n-2)(\cos(\frac{2\pi k}{c}) - 1)$ as a sequence of constants depending only on the value of k. As sequences are restrictions of differentiable functions to the natural numbers, we consider the function $f(\lambda) = -2(n-2)(\cos(\frac{2\pi\lambda}{c}) - 1)$. Its derivative is given by $\frac{4\pi(n-2)}{c}\sin(\frac{2\pi\lambda}{c})$ and has roots at every multiple of $\lambda = \frac{c}{2}$ and $\lambda = c$. On the interval $[0, c]$, $f(\lambda)$ is increasing for $0 < \lambda < \frac{c}{2}$ and decreasing for $\frac{c}{2} < \lambda < c$. It hence has a maximum of $4n - 8$ at $\lambda = \frac{c}{2}$ and a minimum of 0 at $\lambda = c$. The sequence, in k, then has the same maximum and minimum values. It is also increasing for $k = 1, \ldots, \frac{c}{2}$ and decreasing for $k = \frac{c}{2}, \ldots, c$. However, note that c must be even for k to be a natural number. If c is odd, then the sequence is increasing for $k = 1, \ldots, \lfloor\frac{c}{2}\rfloor$, constant for $k = \lfloor\frac{c}{2}\rfloor$ and $k = \lceil\frac{c}{2}\rceil$, and decreasing for $k = \lceil\frac{c}{2}\rceil, \ldots, c$. In either case, the sequence is increasing for $k = 1, \ldots, \lfloor\frac{c}{2}\rfloor$.

Note that the cubic polynomial is increasing from 0 to $\frac{4n^3}{27}$ on the interval $[0, \frac{n}{3}]$ as well. As the sequence of constants is bounded below by 0 and above by $4n - 8$, we have that $0 \leq -2(n-2)(\cos(\frac{2\pi k}{c}) - 1) \leq 4n - 8$ for any $k = 1, \ldots, c$. Furthermore, for all n, we have that $\frac{4n^3}{27} \geq 4n - 8$ where equality is reached for $n = 3$. The existence of an intersection point is then guaranteed by the Intermediate Value Theorem. That is, since $\lambda(\lambda - n)^2$ is continuous on $[0, n]$ and $0 \leq -2(n-2)(\cos(\frac{2\pi k}{c}) - 1) \leq \frac{4n^3}{27}$ for all k, there is a $\lambda_k \in [0, n]$ such that $\lambda_k = -2(n-2)(\cos(\frac{2\pi k}{c}) - 1)$.

As the sequence is increasing for $k = 1, \ldots, \lfloor\frac{c}{2}\rfloor$, the smallest non-zero intersection point corresponds to the choice $k = 1$. Thus, the algebraic connectivity corresponds to the smallest root of equation (3.15).

We now derive the first-term asymptotic expansion for λ, an eigenvalue of the Laplacian matrix of the necklace graph corresponding to $k = 1$. Recall that the Taylor series for cosine is given by $\cos(x) = 1 - \frac{1}{2}x^2 + O(x^4)$. Then, $\cos(\frac{2\pi}{c}) - 1 = (1 - \frac{1}{2}(\frac{2\pi}{c})^2 + O(\frac{1}{c^4})) - 1 = -\frac{2\pi^2}{c^2} + O(\frac{1}{c^4})$. Using this together with equation (3.13) yields

$$\lambda(\lambda - (d+1))^2 = \frac{4(d-1)\pi^2}{c^2} + O\left(\frac{1}{c^4}\right)$$

Temporarily dropping the $O(\frac{1}{c^4})$ term and letting $\lambda = \frac{\lambda_0}{c^2}$ results in the first term asymptotic expansion

$$\frac{\lambda_0}{c^2}(\frac{\lambda_0}{c^2} - (d+1))^2 = \frac{4(d-1)\pi^2}{c^2}$$

so that

$$\frac{(d+1)^2}{c^2}\lambda_0 = \frac{4(d-1)\pi^2}{c^2}$$

and

$$\lambda_0 = \frac{4(d-1)\pi^2}{(d+1)^2}.$$

Hence, after reinserting the $O(\frac{1}{c^4})$ term,

$$\lambda = \frac{4(d-1)\pi^2}{(d+1)^2c^2} + O\left(\frac{1}{c^4}\right)$$

and the equality is obtained.

■

Figure 3.3 provides an example illustrating that the algebraic connectivity corresponds to the smallest root of equation (3.15) with $k = 1$. Note that this choice is represented by the orange line and that this line is the first to intersect the cubic function. Furthermore, for even c, there are $\frac{3c}{2}$ intersection points, each corresponding to a distinct eigenvalue of the necklace graph. The choice $k = \frac{c}{2}$ corresponds to three unique eigenvalues while the remaining eigenvalues are repeated when the sequence of constants is decreasing. The roots of the polynomial are also eigenvalues corresponding to the choice $k = c$. For odd c, there are $\frac{3(c-1)}{2}$ such points. The remaining 3 intersections points are repeats of the case $k = \lfloor \frac{c}{2} \rfloor$. Thus, all non-zero eigenvalues corresponding to the x values of the intersection points have a multiplicity of 2.

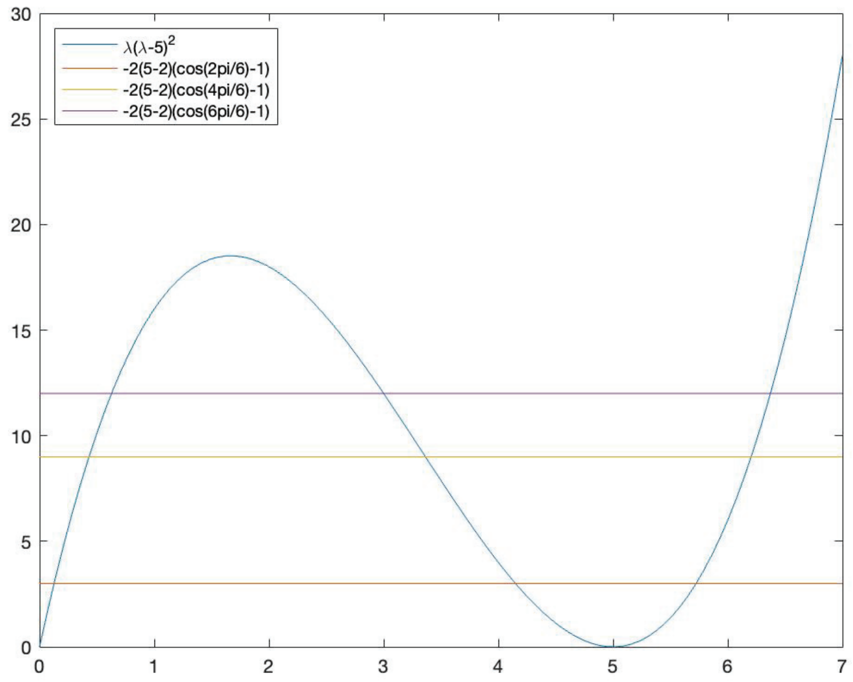


Figure 3.3: Intersection points between the cubic function $\lambda(\lambda-5)^2$ and the expression $-2(5-2)(\cos(\frac{2\pi k}{6})-1)$ with $n = 5$, $c = 6$, and $k = 1, 2, 3$. The orange line corresponds to $k = 1$ and the first intersection point has coordinates (0.1263,3).

Table 3.1 lists the first term asymptotic estimates for an eigenvalue of the Laplacian matrix of d-regular necklace graphs formed by c copies of the graph $K_{d+1}\backslash\{u, v\}$. Note that, when compared to the actual algebraic connectivity values in Table 3.2, these asymptotic estimates are overapproximations. Furthermore, necklace graphs with few copies will have significantly lower algebraic connectivity than given by these first term estimates. Also note that their algebraic connectivity decreases regardless of whether the number of copies or the regularity of the graph is increased.

As a final observation, note that necklace graphs contain cycle subgraphs. In particular, the d-regular necklace graphs studied in this section contain spanning cycle subgraphs of order $(d + 1)c$, where c denotes the number of copies of $K_{d+1}\backslash\{u, v\}$. By Lemma 1.0.2, their algebraic connectivity will be bounded below by the algebraic connectivity of their cycle subgraphs, given by $\lambda_2(C_{(d+1)c}) = 2(1 - \cos(\frac{2\pi}{(d+1)c}))$.

First Term Asymptotic Estimates for Regular Necklace Graphs								
c \ d	3	4	5	6	7	8	9	10
2	1.2337	1.1844	1.0966	1.0071	0.9253	0.8529	0.7896	0.7341
3	0.5483	0.5264	0.4874	0.4476	0.4112	0.3791	0.3509	0.3263
4	0.3084	0.2961	0.2742	0.2518	0.2313	0.2132	0.1974	0.1835
5	0.1974	0.1895	0.1755	0.1611	0.1480	0.1365	0.1263	0.1175
6	0.1371	0.1316	0.1218	0.1119	0.1028	0.0948	0.0877	0.0816
7	0.1007	0.0967	0.0895	0.0822	0.0755	0.0696	0.0645	0.0599
8	0.0771	0.0740	0.0685	0.0629	0.0578	0.0533	0.0493	0.0459
9	0.0609	0.0585	0.0542	0.0497	0.0457	0.0421	0.0390	0.0363
10	0.0493	0.0474	0.0439	0.0403	0.0370	0.0341	0.0316	0.0294

Table 3.1: First term asymptotic estimates valid for the algebraic connectivity of d-regular necklace graphs containing c copies of $K_{d+1}\backslash\{u, v\}$, the complete graph K_{d+1} with the edge $\{u, v\}$ being removed.

Numerical Values for Regular Necklace Graphs								
c \ d	3	4	5	6	7	8	9	10
2	0.7639	0.6277	0.5359	0.4689	0.4174	0.3765	0.3431	0.3153
3	0.4859	0.4312	0.3799	0.3380	0.3039	0.2759	0.2526	0.2329
4	0.2907	0.2679	0.2412	0.2174	0.1971	0.1800	0.1654	0.1530
5	0.1904	0.1783	0.1622	0.1471	0.1340	0.1228	0.1131	0.1048
6	0.1338	0.1263	0.1155	0.1052	0.0960	0.0881	0.0813	0.0754
7	0.0990	0.0939	0.0861	0.0786	0.0719	0.0660	0.0610	0.0566
8	0.0761	0.0724	0.0666	0.0608	0.0557	0.0512	0.0473	0.0439
9	0.0603	0.0575	0.0529	0.0484	0.0444	0.0408	0.0377	0.0350
10	0.0489	0.0467	0.0431	0.0394	0.0361	0.0333	0.0307	0.0286

Table 3.2: Numerical values for the algebraic connectivity of d-regular necklace graphs containing c copies of $K_{d+1}\backslash\{u, v\}$, the complete graph K_{d+1} with the edge $\{u, v\}$ being removed, obtained from the smallest root of equation (3.13) with $k = 1$.

Chapter 4

Algebraic Connectivity of Path-Like Graphs

In [22], Guiduli defines a graph as being *path-like* if it can be built from blocks with bridges in between. He refers to the first and last blocks as end blocks and the remaining ones as middle blocks. His work showed that cubic graphs possessing this structure have minimal algebraic connectivity amongst all connected cubic graphs of the same order. In this chapter, we extend these results to regular graphs with odd degree and conjecture that these also have minimal algebraic connectivity. We first introduce hourglass graphs, a new graph family derived from path-like graphs.

4.1 Hourglass Graphs

We begin with the definition of an hourglass graph.

Definition 4.1.1 *Let G be a 2-vertex-connected graph with a labelled vertex, say u. The hourglass graph H_G is the graph composed of two copies of G, say G_1 and G_2, with adjacent labelled vertices u_1 and u_2.*

Hourglass graphs are path-like graphs consisting of two copies of an end block. This is a restriction on Guiduli's definition, which did not insist on the two end blocks being the same. Yet, we cannot conclude that these graphs have minimal algebraic connectivity amongst all graphs of the same order since Guiduli's work only applies to 3-regular graphs.

As the complete graph K_3 is the smallest 2-connected graph, the smallest hourglass graph is the graph $H_{K_3} = 2K_3 + \{u_1, u_2\}$. This graph is shown in Figure 4.1. Using MATLAB, we found that this graph has an algebraic connectivity of 0.4384. As expected, this value is not minimal amongst connected graphs of order 6 since the path graph on 6 vertices, P_6, has an algebraic connectivity of 0.2679. In fact, amongst all connected graphs of order n, it is known that path graphs minimize algebraic connectivity [31]. Their algebraic connectivity is given by $\lambda_2(P_n) = 2(1 - \cos(\frac{\pi}{n}))$.

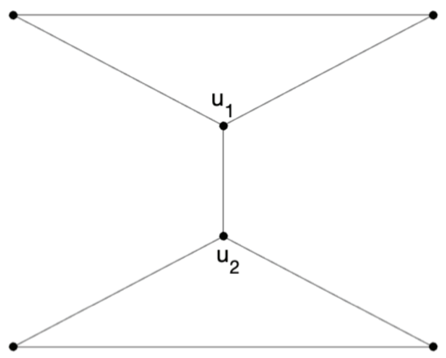


Figure 4.1: The smallest hourglass graph H_{K_3}.

A d-regular hourglass graph is an hourglass graph in which every vertex has degree d. As proved in the following proposition, these graphs must have odd degree and each copy of G must have an odd number of vertices. In turn, a d-regular hourglass graph must have even order.

Proposition 4.1.2 *Let G be a 2-vertex-connected graph of order n and let H_G be its hourglass graph. If H_G is d-regular, then both n and d are odd.*

Proof. In order for H_G to be d-regular, every vertex not labelled u in G must have degree d. Note that the edge $\{u_1, u_2\}$ is a bridge in H_G and that this edge is contained in neither copies. To account for the bridge edge, vertex u must have degree $d-1$. By the Handshaking Lemma, the sum of the degrees of all vertices must equal twice the number of edges. In other words,

$$2e = \sum_{v \in V(G)} deg(v) \tag{4.1}$$

where e denotes the total number of edges and $deg(v)$ the degree of vertex v in G. Now, by our previous argument, we have that $\sum_{v\in V(G)} deg(v) = (n-1)d + (d-1)$ which simplifies to $\sum_{v\in V(G)} deg(v) = dn - 1$.

Now, by solving for the total number of edges in equation (4.1), we obtain that

$$e = \frac{dn-1}{2}.$$

As e represents the number of edges, it must be a natural number. Hence, $dn - 1$ must be divisible by 2. If both d and n are even, or either d or n is even and the other odd, then their product dn is even and so $dn - 1$ is not divisible by 2. Thus, if H_G is a d-regular graph, then both d and n must be odd. ■

For any $d \geq 3$, a d-regular hourglass graph of order $n = 4d - 2$ can be created by following the steps below. Note that, by the first step of our construction, both of its copies will contain the complete bipartite graph $K_{d-1,d-1}$ as a subgraph. We refer to the bipartition within G_1 by X_1 and Y_1 and to the bipartition within G_2 by X_2 and Y_2. Also note that the resulting hourglass graph is not bipartite and that the last step requires the degree to be odd. The cubic case with $d = 3$ is shown in Figure 4.2.

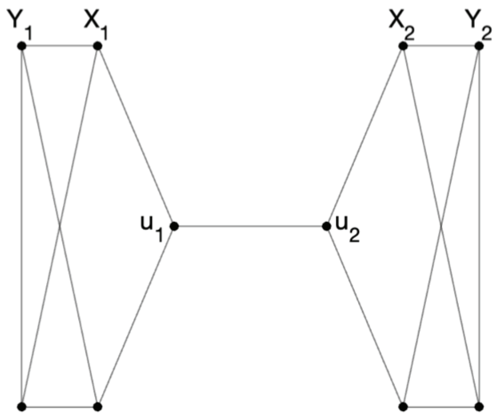


Figure 4.2: A 3-regular hourglass graph.

1. Start with two copies of the complete bipartite graph $K_{d-1,d-1}$, say G_1 and G_2.
2. Create two new vertices, u_1 and u_2, and the edge between them, $\{u_1, u_2\}$.
3. Join vertex u_1 to all vertices in the X_1 part of the bipartition within G_1.
4. Join vertex u_2 to all vertices in the X_2 part of the bipartition within G_2.
5. Connect the vertices in Y_1 and Y_2 as consecutive disjoint pairs.

As the graph in Figure 4.2 is cubic, Guiduli's work showed that it is an algebraic connectivity minimizer for cubic graphs on 10 vertices. The 5-regular hourglass graph obtained by following the previous steps is shown in Figure 4.3. As Guiduli's work only applied to cubic graphs, whether this graph is an algebraic connectivity minimizer

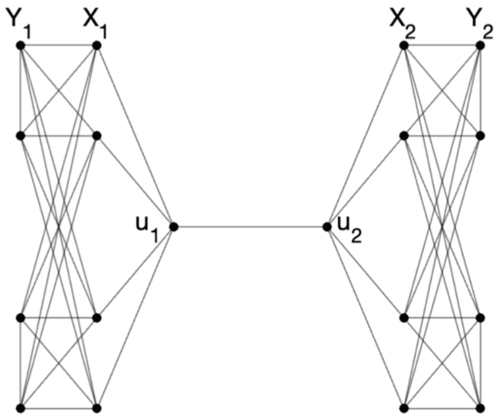


Figure 4.3: The 5-regular hourglass graph obtained by following the given steps.

amongst 5-regular graphs remains to be determined. For this reason, we state the following conjecture for regular hourglass graphs before providing an upper bound for their algebraic connectivity.

Conjecture 4.1.3 *Amongst all odd degree d-regular graphs on $n = 4d - 2$ vertices, the ones that minimize algebraic connectivity are the given regular hourglass graphs.*

We now provide an upper bound for the algebraic connectivity of regular hourglass graphs formed by following the previous steps.

Theorem 4.1.4 *Let H_G be a d-regular hourglass graph obtained by following the given steps. Then an upper bound for its algebraic connectivity is given by*

$$\lambda_2(H_G) \leq 2\sqrt{\frac{d^2-d+1}{3}}\cos\left(\frac{1}{3}\arccos\left(\frac{-3(d^2-3d+2)}{2(d^2-d+1)}\sqrt{\frac{3}{d^2-d+1}}\right)-\frac{4\pi}{3}\right)+d.$$

Proof. Recall that an eigenvector can be thought of as an assignment of weights to the vertices of H_G. We thus assign them to G_1 as follows: for the vertices in the Y_1 part of the bipartition, assign weight x_1; for vertices in the X_1 part, assign weight x_2; for the vertex labelled u_1, assign weight x_3. For the vertices in G_2, assign weight $-x_3$ to the vertex labelled u_2, weight $-x_2$ to the vertices in X_2, and weight $-x_1$ to the vertices in Y_2. This assignment is valid since the sum of the weights is zero and so this vector is orthogonal to the all-ones vector.

This choice of weights leads to the eigenvalue problem

$$\begin{cases} \lambda x_1 &= (x_1 - x_1) + (d-1)(x_1 - x_2) \\ \lambda x_2 &= (x_2 - x_3) + (d-1)(x_2 - x_1) \\ \lambda x_3 &= (x_3 - (-x_3)) + (d-1)(x_3 - x_2) \end{cases} \tag{4.2}$$

with corresponding 3×3 matrix

$$M = \begin{bmatrix} d-1 & -(d-1) & 0 \\ -(d-1) & d & -1 \\ 0 & -(d-1) & d+1 \end{bmatrix}.$$

As was done with regular necklace graphs, we can find that its characteristic polynomial is given by

$$\lambda^3 - 3d\lambda^2 + (2d^2 + d - 1)\lambda - 2d + 2.$$

Making the substitution $\lambda = t + d$ yields the depressed cubic [32] with no t^2 term

$$f(t) = t^3 - (d^2 - d + 1)t + d^2 - 3d + 2. \tag{4.3}$$

Recall that the eigenvalues of the Laplacian matrix are all real. We now show that the three eigenvalues of M are also real by first considering the function in (4.3).

First, note that its derivative is given by $f'(t) = 3t^2 - d^2 + d - 1$ and that $f'(t)$ has roots at $t = \sqrt{\frac{d^2-d+1}{3}}$ and $t = -\sqrt{\frac{d^2-d+1}{3}}$. As $f'(t)$ is a quadratic with positive leading term, $f'(t) \to \infty$ as $t \to \infty$ and $f'(t) \to \infty$ as $t \to -\infty$. Furthermore, $f'(t) < 0$ for $t = 0$ as $-d^2 + d - 1 < 0$ for all d. Thus, $f(t)$ has a local maximum at $t = -\sqrt{\frac{d^2-d+1}{3}}$ and a local minimum at $t = \sqrt{\frac{d^2-d+1}{3}}$.

Since $f(t)$ is a cubic with positive leading term, $f(t) \to -\infty$ as $t \to -\infty$ and $f(t) \to \infty$ as $t \to \infty$. For $t = -\sqrt{\frac{d^2-d+1}{3}}$, we have $f(t) \approx d^2 - d\sqrt{d^2 - d}$ so that $f(t) > 0$ for all $d \geq 1$. Also, for $t = \sqrt{\frac{d^2-d+1}{3}}$, we have $f(t) \approx -2d^2\sqrt{3d^4 - 3d^2 + 3}$ and so that $f(t) < 0$ for all $d \geq 1$. Hence, as $f(t)$ is continuous and changes sign in the intervals $\left(-\infty, -\sqrt{\frac{d^2-d+1}{3}}\right)$, $\left(-\sqrt{\frac{d^2-d+1}{3}}, \sqrt{\frac{d^2-d+1}{3}}\right)$, and $\left(\sqrt{\frac{d^2-d+1}{3}}, \infty\right)$, $f(t)$ must have three real roots by the Intermediate Value Theorem.

For any depressed cubic of the form $t^3 + pt + q = 0$, these may be expressed as

$$t_k = 2\sqrt{\frac{-p}{3}} \cos\left(\frac{1}{3}\arccos\left(\frac{3q}{2p}\sqrt{\frac{-3}{p}}\right) - \frac{2\pi k}{3}\right) \quad \text{for } k = 0,1,2$$

by Viète [33]. Here, $p = -(d^2 - d + 1)$ and $q = d^2 - 3d + 2$ yield

$$t_k = 2\sqrt{\frac{d^2 - d + 1}{3}} \cos\left(\frac{1}{3} \arccos\left(\frac{-3(d^2 - 3d + 2)}{2(d^2 - d + 1)}\sqrt{\frac{3}{d^2 - d + 1}}\right) - \frac{2\pi k}{3}\right)$$

for $k = 0, 1, 2$. Letting $k = 2$ yields the smallest value of t_k and reusing the substitution $\lambda = t + d$ results in

$$\lambda = 2\sqrt{\frac{d^2 - d + 1}{3}} \cos\left(\frac{1}{3} \arccos\left(\frac{-3(d^2 - 3d + 2)}{2(d^2 - d + 1)}\sqrt{\frac{3}{d^2 - d + 1}}\right) - \frac{4\pi}{3}\right) + d.$$

As the algebraic connectivity is the second smallest eigenvalue of the Laplacian matrix, we have that $\lambda_2(H_G) \leq \lambda$. ■

Table 4.1 provides the values and asymptotic behaviour of the algebraic connectivity of these regular hourglass graphs. We extend these results to path-like graphs with one middle block in the next section.

4.2 Path-Like Graphs With One Middle Block

Path-like graphs with higher order can be obtained from hourglass graphs by adding middle blocks. In particular, for odd d-regular path-like graphs with b total blocks and order $n = (4d - 2) + (b - 2)(d + 1)$, these blocks consist of $K_{d+1} \backslash \{v_1, v_2\}$, the complete graph on $d + 1$ vertices with the edge $\{v_1, v_2\}$ removed. The total number of vertices consists of the number of vertices in the two end blocks, $4d - 2$, and $b - 2$ middle blocks of order $d + 1$. Note that the choice $b = 2$ corresponds to the regular hourglass graphs and that any two vertices in the complete graphs can be labelled v_1 and v_2. Figure 4.4 shows a cubic path-like graph of order 14 with one middle block. In this case, the middle block is composed of the graph $K_4 \backslash \{v_1, v_2\}$.

The procedure below can be followed to create any odd degree d-regular path-like graph with b blocks in total. We refer to the first end block by G_1, to the middle blocks by G_i for $i = 2, \ldots, b - 1$, and to the last end block by G_b.

1. Follow the previous steps to create the end blocks G_1 and G_b.

2. Create $b - 2$ copies of the graph $K_{d+1} \backslash \{v_1, v_2\}$, say G_i for $i = 2, \ldots, b - 1$.

3. Draw an edge between vertex u_1 in G_1 to vertex v_1 in G_2.

4. For $i = 2, \ldots, b-2$, draw an edge between vertex v_2 in G_i to vertex v_1 in G_{i+1}.

5. Draw an edge between vertex v_2 in G_{b-1} to vertex u_2 in G_b.

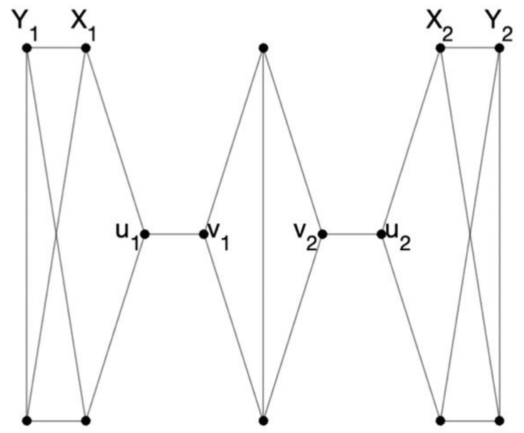


Figure 4.4: The cubic path-like graph with 3 blocks obtained by following the steps.

We now provide an upper bound for the algebraic connectivity of d-regular path-like graphs with one middle block. Its values are listed in Table 4.1 for $d = 3, \ldots, 15$.

Theorem 4.2.1 *Let G be a d-regular path-like graph with one middle block obtained by following the given steps. Then an upper bound for its algebraic connectivity is given by*

$$\lambda_2(G) \leq 2\sqrt{\frac{d^2-d+1}{3}}\cos\left(\frac{1}{3}\arccos\left(\frac{-3(d^2-2d+1)}{2(d^2-d+1)}\sqrt{\frac{3}{d^2-d+1}}\right)-\frac{4\pi}{3}\right)+d.$$

Proof. We assign weights to the vertices of the end blocks as in Theorem 4.1.4. For the middle block, assign a weight of x_4 to vertex v_1, a weight of $-x_4$ to vertex v_2, and a weight of 0 to all vertices not labelled v_1 or v_2. This assignment provides a valid weight vector since the sum of the weights is zero and thus this vector is orthogonal to the all-ones vector. This choice of weights then leads to the eigenvalue problem

$$\begin{cases} \lambda x_1 &= (x_1 - x_1) + (d-1)(x_1 - x_2) \\ \lambda x_2 &= (x_2 - x_3) + (d-1)(x_2 - x_1) \\ \lambda x_3 &= (x_3 - x_4) + (d-1)(x_3 - x_2) \\ \lambda x_4 &= (x_4 - x_3) + (d-1)(x_4 - 0) \end{cases} \tag{4.4}$$

with corresponding 4×4 matrix

$$M = \begin{bmatrix} d-1 & -(d-1) & 0 & 0 \\ -(d-1) & d & -1 & 0 \\ 0 & -(d-1) & d & -1 \\ 0 & 0 & -1 & d \end{bmatrix}.$$

The characteristic polynomial of M is given by

$$\lambda^4 - (4d-1)\lambda^3 + (5d^2 - 2d - 1)\lambda^2 - (2d^3 - d^2 - d)\lambda + d^2 - 2d + 1.$$

Note that this polynomial can be factored as

$$(\lambda - d + 1)(-\lambda^3 + 3d\lambda^2 - (2d^2 + d - 1)\lambda - d + 1),$$

indicating an integer eigenvalue of $d-1$. We now determine the remaining eigenvalues from the cubic polynomial.

Making the substitution $\lambda = t + d$ yields the depressed cubic

$$t^3 - (d^2 - d + 1)t + d^2 - 2d + 1,$$

with $p = -(d^2 - d + 1)$ and $q = d^2 - 2d + 1$. Its roots are given by

$$t_k = 2\sqrt{\frac{d^2 - d + 1}{3}} \cos\left(\frac{1}{3} \arccos\left(\frac{-3(d^2 - 2d + 1)}{2(d^2 - d + 1)}\sqrt{\frac{3}{d^2 - d + 1}}\right) - \frac{2\pi k}{3}\right)$$

for $k = 0, 1, 2$. Letting $k = 2$ yields the smallest value of t_k and reusing the substitution $\lambda = t + d$ results in

$$\lambda = 2\sqrt{\frac{d^2 - d + 1}{3}} \cos\left(\frac{1}{3} \arccos\left(\frac{-3(d^2 - 2d + 1)}{2(d^2 - d + 1)}\sqrt{\frac{3}{d^2 - d + 1}}\right) - \frac{4\pi}{3}\right) + d.$$

As the algebraic connectivity is the second smallest eigenvalue of the Laplacian matrix, we have that $\lambda_2(G) \leq \lambda$. ■

Table 4.1 includes the values for the algebraic connectivity of odd degree d-regular path-like graphs with one middle block. These values were obtained for hourglass graphs by using the equation in Theorem 4.1.4 and for path-like graphs with one middle block by using the equation in Theorem 4.2.1. They were also independently

Numerical Values of Odd Degree d-Regular Path-like Graphs							
Degree	3	5	7	9	11	13	15
Hourglass	0.2215	0.1547	0.1182	0.0956	0.0802	0.0691	0.0607
Middle Block Path-like	0.1049	0.0757	0.0584	0.0474	0.0399	0.0344	0.0303

Table 4.1: Numerical values of the algebraic connectivity of odd degree d-regular hourglass graphs and path-like graphs with one middle block.

obtained by MATLAB and agree with the values found in Table 2.6 in the case of cubic graphs.

As can be observed, the algebraic connectivity of both graph families approaches zero as the degree, and hence the number of vertices, is increased. The algebraic connectivity of the hourglass graphs is also greater than the one for the path-like graphs with one middle block for all odd d. Hence, the algebraic connectivity of the hourglass graphs may serve as an upper bound for odd degree d-regular path-like graphs. The minimal algebraic connectivity of cubic graphs is also known to equal $(1 + o(1))\frac{2\pi^2}{n^2}$, where n is the number of vertices [24].

Supported by Guiduli's research on cubic graphs, we state the following conjecture.

Conjecture 4.2.2 *Amongst all odd d-regular graphs on $n = (4d-2)+(b-2)(d+1)$ vertices, the ones which minimize algebraic connectivity are the given path-like graphs with b blocks.*

Research on quartic, or four-regular graphs, with minimal algebraic connectivity reveals to be more challenging. In 2020, Abdi, Ghorbani, and Imrich [24] found that these graphs must also possess a path-like structure. However, this structure now has four end block and five middle block possibilities. Long blocks, or blocks composed of shorter blocks, are also likely. Abdi and Ghorbani later specified that there is only one middle block option for a quartic graph to have minimal algebraic connectivity [25], but generalizations for even degree d-regular path-like graphs remain unlikely.

In the upcoming chapters, we shift our attention to a variety of computational algorithms used to construct graphs with fixed order and size. Our interest lies in determining which algorithm returns graphs with the highest algebraic connectivity as well as the resulting graph's structure in terms of degree sequence.

Chapter 5

Algorithms on Empty Graphs

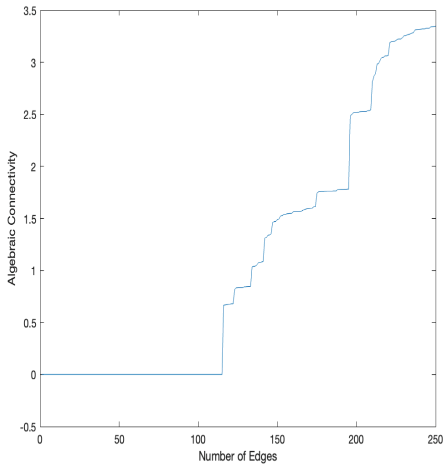
We now shift our focus to heuristic algorithms that can be used to create simple graphs of fixed order and size. As opposed to most of our results in the previous chapters, these graphs are not required to be regular. Hence, the goal of these algorithms is to maximize a graph's algebraic connectivity subject to a fixed number of vertices and edges. Each algorithm's performance is based on its run time as well as on the algebraic connectivity of the resulting graph. Thus, algorithms which return graphs with high algebraic connectivity in less time are considered more efficient.

In this chapter, we introduce four algorithms on originally empty graphs of order 50. One edge, chosen by varying criteria, is added to the graph until it has a final size of 250. Each method's performance will be analyzed through an algebraic connectivity plot and a degree histogram of the resulting graph. The first algorithm we present relies entirely on a random assignment of edges.

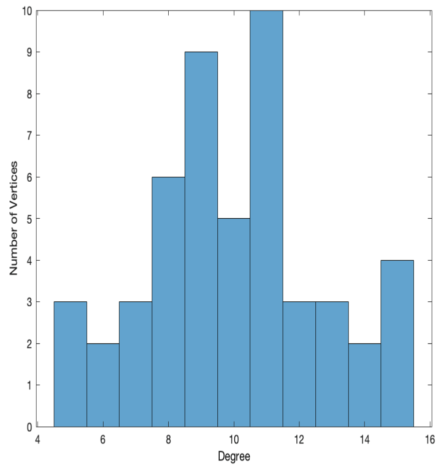
5.1 Random Edge Addition

Adding edges at random is likely the easiest way of adding edges to a graph. In this method, two integers ranging from 1 to the number of vertices in the graph are arbitrarily chosen at each iteration. These two integers become the edge endpoints. Before the edge can be added, two conditions must be checked. First, the two numbers cannot be equal to each other. Otherwise, they would form a loop in the graph. The second condition is that the same pair of integers cannot be chosen more than once. If it were, multiple edges between their corresponding endpoints would be formed. In either case, a simple graph would not be generated. All other options are viable.

Figure 5.1a illustrates the algebraic connectivity plot of this method. Notice that the algebraic connectivity remains at 0 until 120 edges are added. This result is far from desirable as the algebraic connectivity of a spanning tree containing 49 edges is already greater than zero. Thus, this method is not optimal in creating graphs with high algebraic connectivity. This leads us to consider algorithms which form spanning trees and hence increase algebraic connectivity earlier on in the process as more efficient. A jump in algebraic connectivity is also noticeable after the addition of the next edge. Namely, the algebraic connectivity increases from 0 to 0.6223 before another short plateau is reached. This pattern frequently reappears up until all 250 edges are in the graph.



(a) Algebraic Connectivity Plot



(b) Degree Histogram

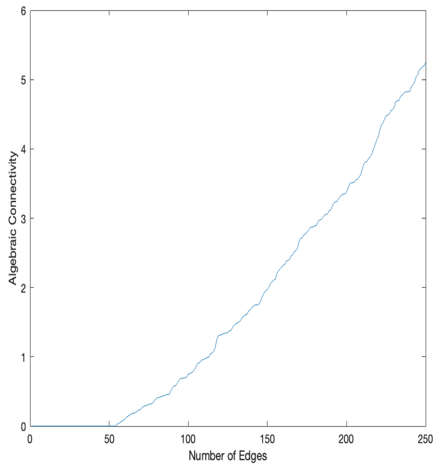
Figure 5.1: Algebraic connectivity plot and degree histogram for the random edge addition algorithm on graphs with 50 vertices and 250 final edges.

As this algorithm relies on randomness, each plot and resulting algebraic connectivity differs. For this reason, 1000 trials were performed and an average algebraic connectivity of 3.3280 was found. The minimum observed algebraic connectivity was 0.9024 and the maximum was 4.4999. This experiment resulted in an overall broad range of 3.5975 for the possible algebraic connectivity of all graphs formed by random edge addition. This range indicates that graph structure has a wide effect on algebraic connectivity. Still, a pattern emerges for graphs of larger order. For very large and sparse random Erdős–Rényi graphs, it is known that the algebraic connectivity tends to zero as the number of vertices is increased [34]. We now turn our attention to the degree distribution of the resulting graph.

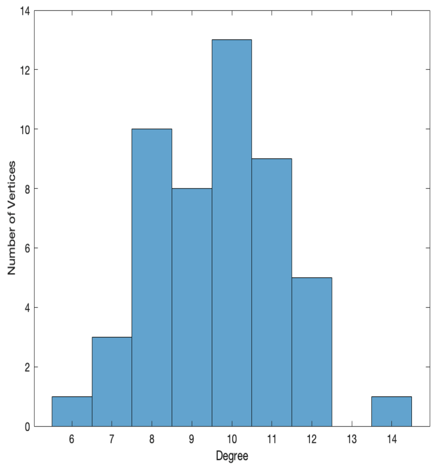
As depicted in the histogram in Figure 5.1b, the sampled graph has a broad range of degrees. It has 3 vertices of minimum degree 5 and 4 vertices with a maximum degree of 15, which creates a range of 11 possible degrees. The majority of vertices appear to have around 8 to 11 neighbours. A close symmetry is also noteworthy. The second algorithm we present preserves random edge selection alongside a careful filter before edge addition.

5.2 The Achlioptas Process

First described by Dimitris Achlioptas at a Fields Institute workshop in 2000 [35], adding a strategy to determine which edges get added is an improvement on random edge addition. His proposed algorithm starts on an empty graph and selects a random subset of all possible edges at each iteration. Initially, Achlioptas only considered subsets consisting of two edges. However, the size of this subset can range anywhere from one edge to all possible edges. In the case that only one edge is chosen each time, the algorithm is identical to random edge addition. Otherwise, each edge must be tested by some rule. For our purpose, the edge which increases algebraic connectivity the most will be added to the graph. Nowadays, any algorithm that adds a strategy to a random graph is known as an Achlioptas process [36].



(a) Algebraic Connectivity Plot



(b) Degree Histogram

Figure 5.2: Algebraic connectivity plot and degree histogram of the Achlioptas process algorithm with 60 potential edges per iteration on graphs with 50 vertices and 250 final edges.

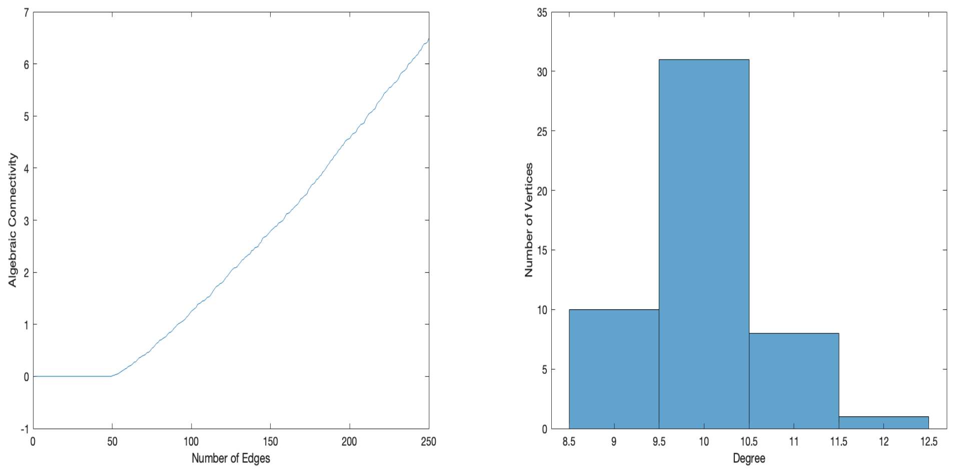
We first allowed this algorithm to choose 60 edges per iteration. Since a complete graph on 50 vertices has 1225 edges, this corresponds to nearly 5 percent of the total edges being considered in each run through the algorithm. The results are shown in Figure 5.2a. At first glance, we can observe that the algebraic connectivity increases away from zero once 50 edges are in the graph. Furthermore, the 'staircase' pattern characteristic of random edge addition gets replaced by a smoother curve. These are both improvements when compared to the previous plot. Out of 100 trials, an average algebraic connectivity of 5.2732 was observed. A maximum of 5.6740 and minimum of 5.2151 were also reached by the algorithm.

The histogram in Figure 5.2b showcases the degree distribution of a resulting graph. Its symmetry, likely appearing from the underlying random edge selection, is to be noted. The minimum degree is 6 and the maximum degree is 14, resulting in a range of 8.

This algorithm was also tested by varying the number of chosen edges per iteration. Namely, edge selection was increased to 120 edges and then to 250 edges. These numbers corresponds to nearly 10 percent and 20 percent of all edges, respectively. In the case of 120 edges being chosen, an average algebraic connectivity of 5.2599 was reached. In the other case, an average algebraic connectivity of 5.2553 was obtained. The maximum algebraic connectivity did not vary, with a value of 5.5737 in the first case and a value of 5.5204 in the other. The same was true for the minimum, with respective values of 5.1932 and 4.9529. Hence, increasing the number selected edges per iteration does not result in graphs with higher algebraic connectivity. We now present an algorithm that does not rely on randomness.

5.3 The Edge-Augmentation Algorithm

The edge-augmentation algorithm, first implemented by Ghosh and Boyd in 2006 [37], takes the Fiedler vector into account. This algorithm begins by computing the eigenvector corresponding to the second smallest eigenvalue and sorting its entries in ascending order. By sorting the Fiedler vector, we guarantee that the absolute difference between the first and the last entry will be a maximum. Its indices are then sorted with respect to their newly ordered entries. It is important to note that each index represents a vertex in the graph. The first and last indices are then located and an edge is added between their vertices in the graph. These steps are repeated until the desired number of edges is reached.



(a) Algebraic Connectivity Plot (b) Degree Histogram

Figure 5.3: Algebraic connectivity plot and degree histogram of the standard edge-augmentation algorithm on graphs with 50 vertices and 250 final edges.

Figure 5.3a illustrates the output of this algorithm. As the graph is originally disconnected, it has an algebraic connectivity of zero. For the algebraic connectivity to increase, the graph must become connected. As there are 50 vertices, a minimum of 49 edges must be added to form a spanning tree. This explains why the algebraic connectivity stays at zero for the first 50 iterations. As more edges are added, its shape begins to resemble a smooth line. A final algebraic connectivity of 6.4878 is obtained when using this algorithm. As the Fiedler vector is sorted at each iteration, this method does not rely on randomness and thus always returns the same value.

As shown in the histogram in Figure 5.3b, the final graph has a fairly symmetric degree distribution. Most vertices have degree 10, while a near equal amount are adjacent to either 9 or 11 neighbours. A single vertex has degree 12. This degree distribution also has a smaller range than the two algorithms we previously discussed, which may be partially due to the absence of randomness. The histogram above also illustrates an underlying feature necessary for large algebraic connectivity. Graphs with higher minimum degree have higher algebraic connectivity. That is, a graph's algebraic connectivity is bounded above by its minimum degree [38].

Although this algorithm produces graphs with high algebraic connectivity, it is also quite restrictive. That is, instead of considering all possible vertices, it only looks at the first and last entries of the sorted Fiedler vector. This limitation leaves little to no room for randomness and may impact the graph's initial construction. This raises a question. What happens if the Fiedler vector contains more than one equal maximum or minimum entry? The following modification to the edge-augmentation algorithm provides an answer.

5.3.1 Random Tree Augmentation

Recall that the original edge-augmentation algorithm starts on an empty graph. As it originally contains no edges, there is no clear distinction between any of the vertices. Thus, by assigning the first few edges at random, different trees may be formed. Also recall that the standard edge-augmentation algorithm only considers the first and last entry of the Fiedler vector. This algorithm thus varies from the one implemented by Ghosh and Boyd as it originally considers more entries in the Fiedler vector to create the underlying tree. Thankfully, only a minor change is required to account for this randomness factor.

This modification happens once the Fiedler vector is computed. Instead of sorting it to create a single maximum and minimum, all of its entries are considered. Whenever more than one maximum is found, its corresponding indices are found and one of them is chosen at random. A similar search and selection is done for the minimum. Finally, an edge between the vertices corresponding to the two randomly chosen indices is added to the graph.

A single maximum and multiple minima is a further possibility. In this case, the index of the maximum is kept and the minimum index is chosen at random. Likewise, a maximum index is randomly picked if it has multiple candidates and a single minimum exists. If there is a single maximum and minimum, then this method is identical to the original edge-augmentation algorithm.

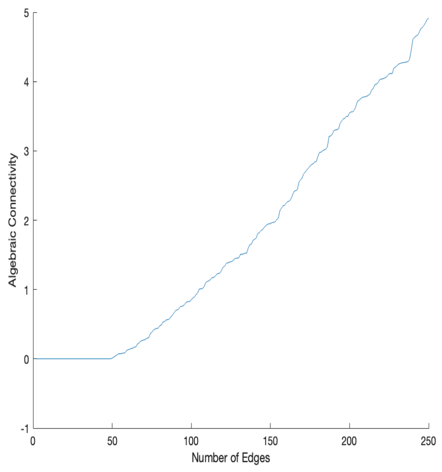
Ultimately, this modification does not result in graphs with higher average algebraic connectivity. Out of 1000 trials, the resulting average algebraic connectivity was 6.3858. This is 1.57 percent worse than having performed the edge-augmentation algorithm by itself. A minimum of 3.6816 and a maximum of 6.5843 were also observed. Interestingly, one graph had a slightly higher algebraic connectivity than the one found by the edge-augmentation algorithm. The structure of the underlying tree may thus have a small positive impact on the final algebraic connectivity. However, the creation of a random tree does not increase algebraic connectivity overall. There was no notable difference in degree distribution.

5.4 Brute Force Search

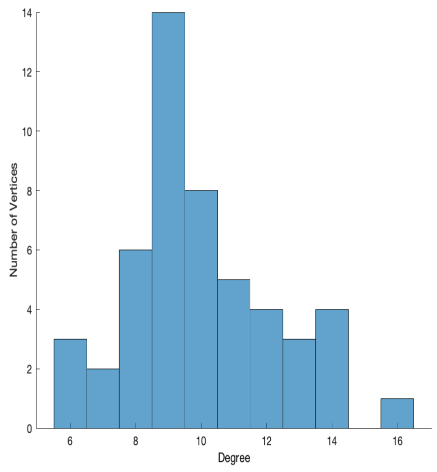
Brute force search is quite different from the three algorithms we have examined so far. As opposed to only considering a handful of edges, this algorithm tests every possible edge absent from the graph. For an initially empty graph on 50 vertices, this corresponds to 1225 candidates. Although the first several edges do not have an impact on algebraic connectivity, the impact of a brute force search is made evident once the graph becomes connected. At this point, the graph will have a non-zero algebraic connectivity. This positive value is then treated as a baseline for the addition of the next edge. For our purposes, we use the edge-augmentation algorithm to select the first 49 edges. This allows the brute force search algorithm to start with a non-zero algebraic connectivity.

Edge addition is done by treating each potential edge separately. At the beginning, one such edge is added to the pre-existing graph with known algebraic connectivity. Then, this parameter is computed for the new graph. The difference in algebraic connectivity is then stored in a list. The prospective edge is then deleted in order to return to the original graph. This process is repeated until each absent edge has been tested. Ultimately, the list is sorted and the edge which increases algebraic connectivity the most is added to the graph. This algorithm continues until the desired number of edges are present.

Figure 5.4a highlights the algebraic connectivity plot of the brute force search algorithm. First, note that this parameter begins increasing once 49 edges are present in the graph. This is to be expected since it initially depends on the edge-augmentation algorithm. As seen for the Achlioptas process, the plot is also smoother than the one provided by random edge addition. In the end, this algorithm achieves a constant algebraic connectivity of 4.9138.



(a) Algebraic Connectivity Plot



(b) Degree Histogram

Figure 5.4: Algebraic connectivity plot and degree histogram of the brute force search algorithm with 50 vertices and 250 final edges.

The final graph's degree distribution can be seen in the form of a histogram in Figure 5.4b. Notice that the resulting graph has the same range of degrees as in the random edge addition algorithm. In it, there are three vertices of minimum degree 6 and a unique one of maximum degree 16. This generates a range of 11 possible degrees. Still, an abundant number of vertices have degree 9.

We now proceed to compare the four algorithms in terms of resulting algebraic connectivity and computational complexity.

5.5 Comparison

All four algorithms originally begin on an empty graph of order 50. Then, at each iteration, a single edge is added until 250 of them are present. In the first two methods, randomness plays a role in which edges are added. In the other two, randomness is mostly excluded. Still, several significant differences emerge from deeper analysis.

The average and maximum algebraic connectivity returned by each algorithm discussed in this chapter are found in Table 5.1. For the random edge addition algorithm, the average was taken over 1000 iterations while it was taken over 100 iterations for the Achlioptas process. Note that the table denotes algebraic connectivity by AC and that the edge-augmentation algorithm returns graphs with the highest algebraic connectivity overall. Still, the problem of finding optimal graphs on a given number of vertices and edges is known to be NP-complete [39].

Algorithm Comparison in Terms of Algebraic Connectivity		
Algorithm	Average AC	Maximum AC
Random Edge Addition	3.3280	4.4999
Achlioptas Process	4.3813	5.6740
Brute Force	4.9138	4.9138
Edge-Augmentation	6.4878	6.4878

Table 5.1: Comparison between average and maximum final algebraic connectivity returned by the four algorithms on empty graphs.

Figure 5.5 showcases the four plots on the same set of axes. From it, it is made clear that the edge-augmentation algorithm not only produces graphs with a higher final algebraic connectivity overall, but ones with a higher algebraic connectivity for any number of edges once the graph is connected.

In terms of computational complexity, both the edge-augmentation algorithm and random edge addition are $O(n^2 log(n))$. The first n comes from iteration, while the $nlog(n)$ comes from sorting. Random edge addition only sorts the eigenvalues, while the edge-augmentation algorithm sorts both the eigenvalues and the Fiedler vector. The Achlioptas process, on the other hand, runs in $O(n^3 log(n))$. It includes nested iterations, once for all edges and once for the chosen edges, as well as sorting eigenvalues. For this reason, the Achlioptas process is more computationally expensive and inefficient for larger graphs. Brute force search, as should be expected, is even worse as it runs in $O(n^4 log(n))$. This algorithm involves three nested iterations as well as eigenvalue sorting. The first iteration is for all edges, while the other two involve looking through all the entries of the adjacency matrix. Altogether, we conclude that the edge-augmentation algorithm is most efficient since it provides graphs with larger algebraic connectivity in less time.

The minimum degree and maximum distance (MDMD) algorithm presented by Li, Hao, Wang, and Wei in 2018 [40] also deserves a mention. Similarly to the edge-augmentation algorithm, this algorithm returns graphs with high algebraic connectivity based on the Fiedler vector itself. Furthermore, as it considers the minimum degree and maximum distance instead of the Laplacian matrix and its eigenvalues, it has a lower time complexity $(O(n+m))$ than the edge-augmentation algorithm for very large and sparse graphs. Yet, edge addition by the edge-augmentation algorithm still outperforms it in most simulations [40].

So far, our interest resided in starting algorithms from empty graphs. Now that we know that the edge-augmentation algorithm performs well, we will test it on other graph families. To do so, we will first create the underlying graph by using its adjacency matrix or a different algorithm. Once created, the edge-augmentation algorithm will add the remaining edges to reach the size of the graph of interest. In particular, we will focus on how this algorithm performs starting from random graphs, random regular graphs, and complete bipartite graphs. The last two are known to be algebraic connectivity optimizers under different conditions [21]. Long-term patterns will also be of interest.

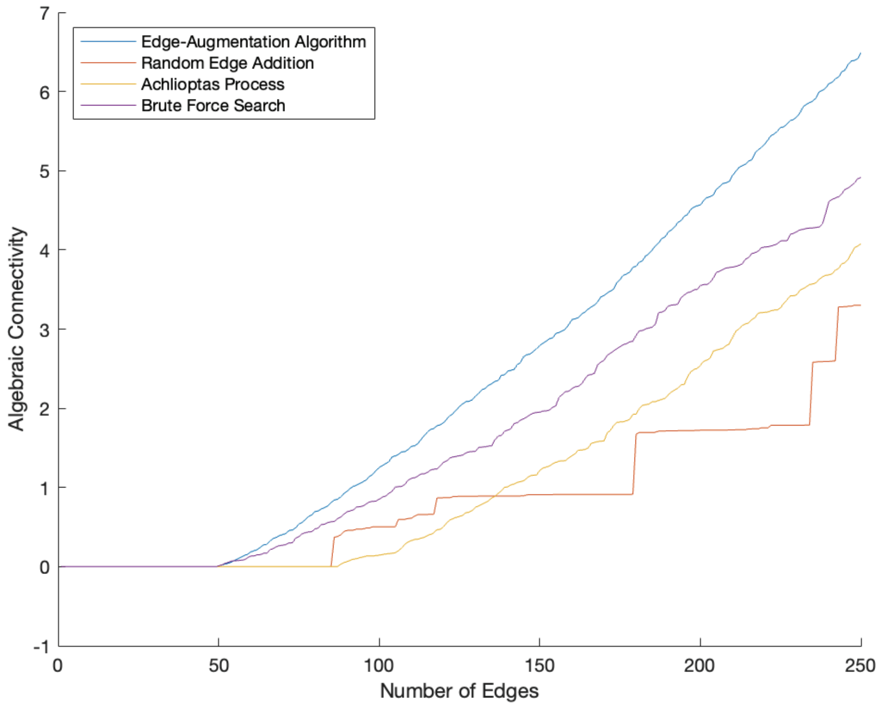


Figure 5.5: Comparison between the algebraic connectivity plots from the standard edge-augmentation algorithm, the Achlioptas process, random edge addition, and brute force search.

Chapter 6

Augmentation from Graph Families

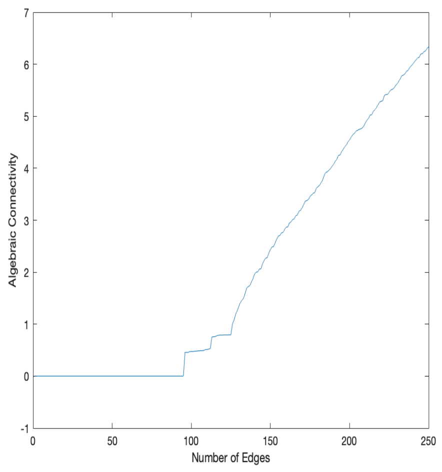
6.1 Starting from Random Graphs

We begin this chapter by examining what happens to algebraic connectivity when the random edge addition and edge-augmentation algorithms are combined. By that, we mean that half of the total edges are added randomly either at the beginning or the end while the remaining ones are chosen by the edge-augmentation algorithm. This method is different from random tree augmentation as random edge addition does not take the entries of the Fiedler vector into account and a larger number of edges are randomly chosen.

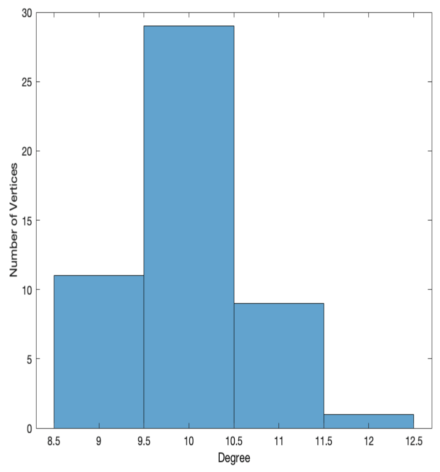
Figure 6.1 illustrates this process with the addition of 125 random edges at the beginning. The remaining edges are then added by the edge-augmentation algorithm. Unsurprisingly, when compared to our previous results, the plot visible in Figure 6.1a looks like a combination of the two algorithms. The increasing plateaus characteristic of random edge addition are present in the first half, while the edge-augmentation curve is noticeable in the second half. On average, out of 1000 trials, this algorithm's resulting final algebraic connectivity is 6.3084. This result is worse than the edge-augmentation algorithm by nearly 2.7 percent, but a vast improvement on simple random edge addition. A minimum of 4.3158 and a maximum of 6.5029 were also observed. Thus, this new method does not provide a general improvement on previous findings.

Figure 6.1b showcases a histogram of the final graph's degree distribution. As opposed to the plot, features of both algorithms are not apparent. In fact, the degrees seem to behave similarly to the ones solely obtained by the edge-augmentation algorithm. Thus, the effect of this added randomness factor is not noticeable amongst the vertices.

This algorithm could also have been performed in the opposite order. That is, edge-augmentation could be performed first followed by the remaining edges being



(a) Algebraic Connectivity Plot



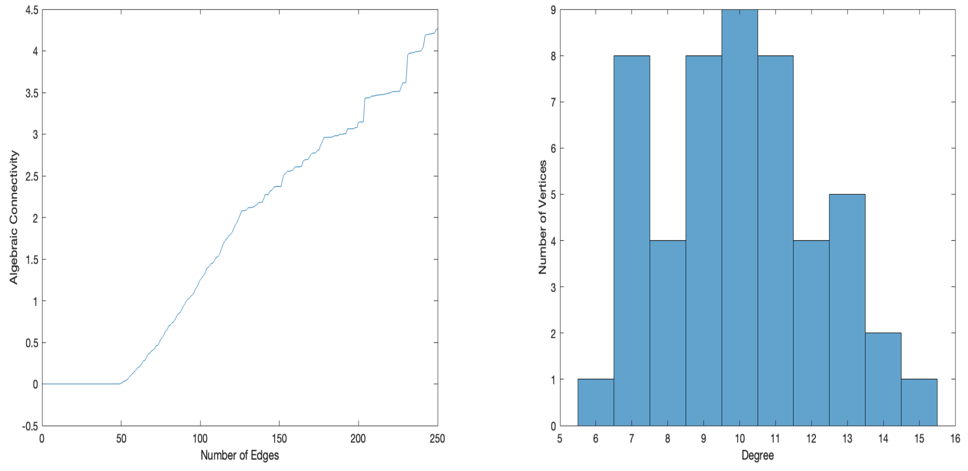
(b) Degree Histogram

Figure 6.1: Algebraic connectivity plot and degree histogram of the standard edge-augmentation algorithm starting from a random graph.

randomly assigned. Here, the condition that those edges were not already added by the edge-augmentation algorithm must be ensured. Otherwise, multiple edges between some vertices might be created. Since most of the edges added by the edge-augmentation algorithm will be used to connect the graph, this combination is likely worse than using the algorithm by itself. Once the graph is connected, the remaining edges, mostly chosen at random, will have a bigger impact on algebraic connectivity.

The results are displayed in Figure 6.2. As expected, it can be seen on the left that the edge-augmentation algorithm started increasing algebraic connectivity once 50 edges were contained in the graph. The algorithm reached an algebraic connectivity of 2.0244 before random edge addition began. However, as the remaining edges were added randomly and thus likely not optimally, a low average algebraic connectivity of 4.2445 was obtained after 1000 trials. This result is still better than using random edge addition alone by 27.5 percent. However, when compared to the edge-augmentation algorithm, it is 34.5 percent worse. A minimum of 3.0929 and a maximum of 4.9547 were also recorded. The random edge addition plateaus are still present, and the transition between the two algorithms is abrupt.

As can be seen in Figure 6.2b, the final graph's degree distribution is closely related to the one found by the random edge addition algorithm. Although it seems to retain its symmetry, the degrees range from a minimum of 6 to a maximum of 15. This range is much wider than the one provided by the edge-augmentation algorithm. Nonetheless, this result should not be surprising since randomness plays a larger role in this algorithm than in the previous one. Thus, the edges chosen once the graph is connected are likely more important than the initial ones in determining the final graph's algebraic connectivity.



(a) Algebraic Connectivity Plot (b) Degree Histogram

Figure 6.2: Algebraic connectivity plot and degree histogram of the standard edge-augmentation algorithm followed by random edge addition.

This notion can be clearly seen when the three algorithms are plotted on the same axes, as in Figure 6.3. The plot of the edge-augmentation algorithm followed by random edge addition is coloured yellow. In the beginning, this plot is identical to the standard edge-augmentation algorithm, in blue. However, they start to differ as soon as we begin adding random edges. In the end, a large gap in algebraic connectivity is noticeable between the two plots. The addition of random edges at the start has less of an impact on the final algebraic connectivity, as can be seen in orange. While this method and the edge-augmentation algorithm initially increase algebraic connectivity differently, both will produce a similar algebraic connectivity asymptotically. Thus, we conclude that randomness has a bigger impact in the later phase of graph creation.

Now that we have a better understanding of the impact of randomness on algebraic connectivity, we turn our attention to random regular graphs.

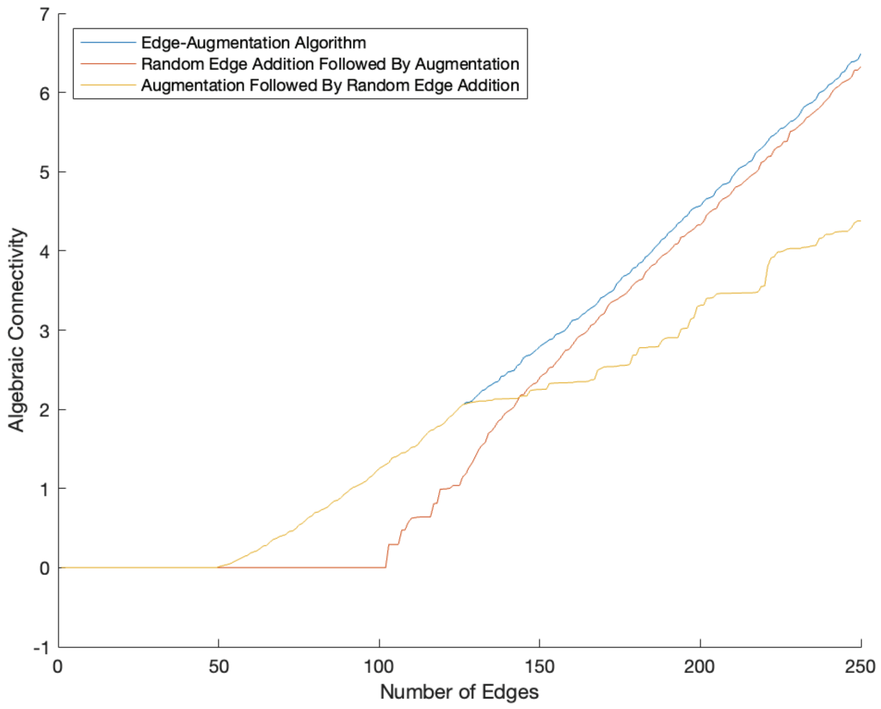


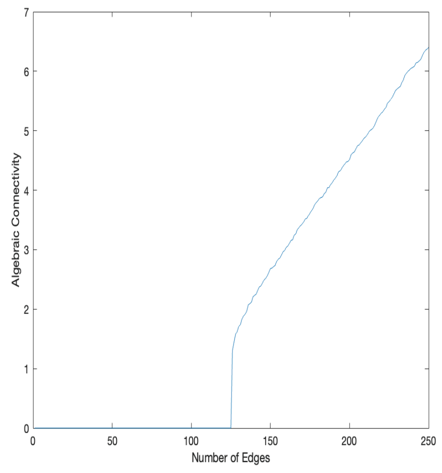
Figure 6.3: Comparison between the standard edge-augmentation algorithm, random edge addition followed by augmentation, and augmentation followed by random edge addition for graphs with 50 vertices and 250 total edges.

6.2 Starting from Random Regular Graphs

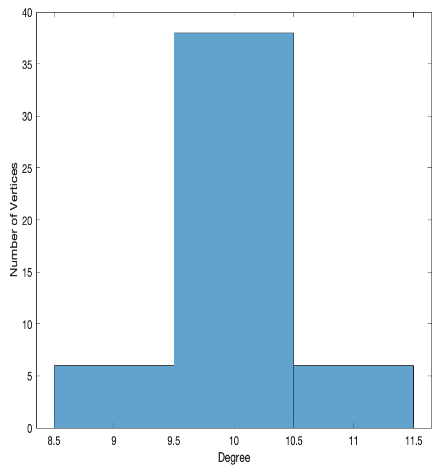
In this section, we implement the edge-augmentation algorithm starting on a random d-regular graph. These graphs were obtained using a heuristic algorithm [41] and are of interest since they combine the structure of the edge-augmentation algorithm as well as the freedom of random edge addition.

In a random d-regular graph, edges are chosen at random amongst all n possible vertices. Nevertheless, every vertex must ultimately have degree d. Thus, vertices become saturated once they have d neighbours and no new edges can be added to them. As a consequence of the well-known handshaking lemma, these graphs will have $dn/2$ edges.

The observation that an empty graph can be thought of as being zero-regular since every vertex has degree 0 motivates us to investigate how the edge-augmentation algorithm behaves if it starts on different random regular graphs. We consider the 5-regular graph as a good template for this family. It is important to keep in mind that these graphs already fix some edges. That is, the edges that were used to create the regular graphs contribute to the total size of the graph of interest and cannot be reused by the edge-augmentation algorithm. In the case of the 5-regular graph on 50 vertices, 125 edges are fixed. Thus, exactly half of the edges remain to be added by the edge-augmentation algorithm. As the graphs grow in regularity, they also grow in size. Hence, as fewer edges get added by the edge-augmentation algorithm, it is reasonable to expect that the final graph's algebraic connectivity will not be higher in regular graphs of higher degree.



(a) Algebraic Connectivity Plot



(b) Degree Histogram

Figure 6.4: Algebraic connectivity plot and degree histogram of the standard edge-augmentation algorithm starting from a random 5-regular graph.

Figure 6.4 showcases the results of the edge-augmentation algorithm as applied to a random 5-regular graph. On the left, it can be observed that the algebraic connectivity is zero until the graph's creation. Then, it jumps up to 1.3663. This is slightly higher than the expected algebraic connectivity of 1. In the end, out of 1000 trials, a final average algebraic connectivity of 6.3380 was obtained. A minimum of 3.7610 and a maximum of 6.5297 were also observed. Ultimately, performing the edge-augmentation algorithm on the empty graph still results in a higher algebraic connectivity overall. Next, we examine this resulting graph's degree distribution.

From the histogram in Figure 6.4b, it is noticeable that the degree distribution is almost identical to the one obtained when performing the edge-augmentation algorithm on an empty graph. The range of degrees, from 9 to 11, is the same as well as their frequency. Most vertices have degree 10 while an equal amount have either degree 9 or 11. The only notable difference is the absence of the degree 12 vertex. This result indicates that no real differences in terms of degree exist between starting from an empty graph or from another regular graph. Thus, it is likely that the edge-augmentation algorithm builds close to regular graphs. We will now examine these traits in graphs of different regularities.

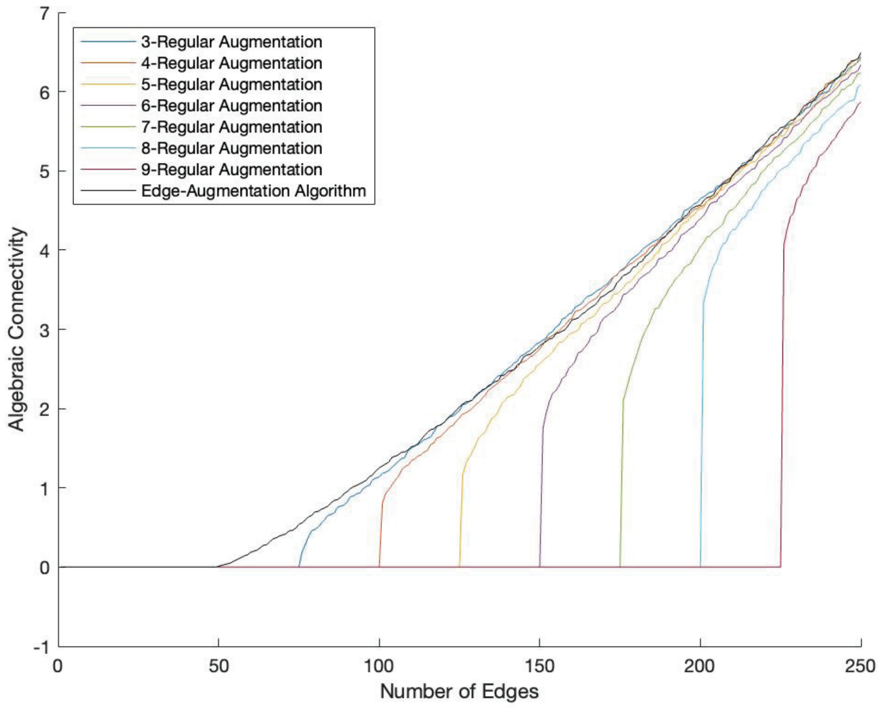


Figure 6.5: Algebraic connectivity plots of the edge-augmentation algorithm starting from a random 3-regular to a random 9-regular graph.

Figure 6.5 illustrates the general behaviour of starting the edge-augmentation algorithm on a random regular graph. In it, a randomly chosen 3-regular to 9-regular graph is used as a base for augmentation. The standard edge-augmentation algorithm, in black, is also present to facilitate comparison.

It is noteworthy that the initial algebraic connectivity increases in each case in an almost linear fashion. Also, starting augmentation from the 3-regular up to the 8-regular graphs results in graphs with almost equal final algebraic connectivity. The case of the 9-regular graph is slightly different since 225 edges are fixed by the underlying structure and the edge-augmentation algorithm is left with very few iterations. These values are shown in Table 6.1, where both the initial and final algebraic connectivity were computed by MATLAB. As before, algebraic connectivity is denoted by AC.

Algebraic Connectivity of Random Regular Graphs			
Regularity	Initial AC	Final AC	Difference
3	0.2983	6.3965	6.0982
4	0.6178	6.3798	5.7620
5	1.3663	6.3380	4.9717
6	1.9709	6.3121	4.3412
7	2.5647	6.2270	3.6623
8	3.4783	6.1004	2.6221
9	4.3305	5.8617	1.5312

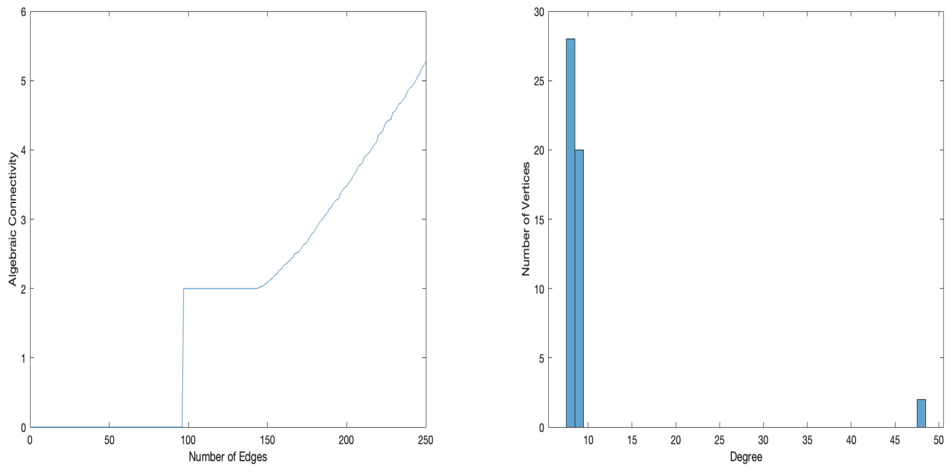
Table 6.1: Comparison between initial and final algebraic connectivity amongst random regular graphs.

This table shows that although the initial algebraic connectivity increases, the final algebraic connectivity decreases since more of the edges were used to create the regular graphs. In the next section, we look at another graph family known to be early algebraic connectivity maximizers, the complete bipartite graphs.

6.3 Starting from Complete Bipartite Graphs

Complete bipartite graphs are highly structured graphs known to have high algebraic connectivity. This is true particularly when these graphs are compared to others with the same number of vertices and edges, see Ogiwara, Fukami, and Takahashi [42]. From Chapter 2.1, recall that their algebraic connectivity is equal to the cardinality of their smaller part.

Amongst all trees, it is known that the star graph S_{49} is an algebraic connectivity maximizer [21]. This graph is also the complete bipartite graph $K_{1,49}$. Yet, we choose $K_{2,48}$ as a good representative of this graph family since its structure more closely resembles that of larger complete bipartite graphs. For graphs with 50 vertices, $K_{3,47}$ and $K_{4,46}$ will likely have too many fixed edges for the edge-augmentation algorithm to make a noticeable difference.



(a) Algebraic Connectivity Plot (b) Degree Histogram

Figure 6.6: Algebraic connectivity plot and degree histogram of the standard edge-augmentation algorithm starting from the complete bipartite graph $K_{2,48}$.

The results of the edge-augmentation algorithm starting on $K_{2,48}$ are shown in Figure 6.6. As expected, the algebraic connectivity is initially zero before jumping to 2 once the bipartite graph is created. As can be seen in the plot, fifty additional edges need to be added to the graph before algebraic connectivity begins increasing. As $K_{2,48}$ fixes 96 edges at the beginning, only 154 edges remain to be added by the edge-augmentation algorithm. In the end, a final algebraic connectivity of 5.2735 is obtained. This is around 18.7 percent worse than performing the edge-augmentation algorithm on an empty graph. Still, the degree distribution should be analyzed.

A first observation from the degree histogram is the presence of two high degree vertices. Both have degree 48 and correspond to the two central hubs of the complete bipartite graph. All other vertices have a degree of either 8 or 9. This display indicates that the starting graph keeps its essential features even when the edge-augmentation algorithm adds many more edges. We now explore the trends for starting augmentation on any complete bipartite graph.

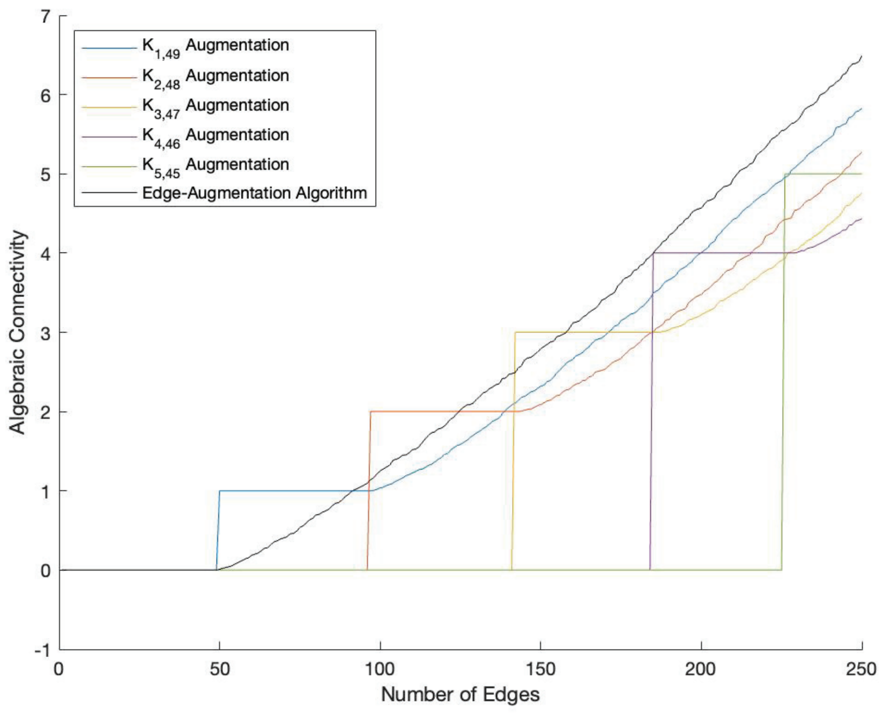


Figure 6.7: Algebraic connectivity plots of the edge-augmentation algorithm starting on the complete bipartite graphs $K_{1,49}$ to $K_{5,45}$.

Figure 6.7 showcases the general behaviour of starting the edge-augmentation algorithm on a complete bipartite graph. In it, the star graph $K_{1,49}$ up to the complete bipartite graph $K_{5,45}$ are used as a base for augmentation. The standard edge-augmentation algorithm, in black, is also drawn to facilitate comparison. As expected, the initial algebraic connectivity of each complete bipartite graph is equal to the cardinality of its minimum set. For example, the initial algebraic connectivity of $K_{3,47}$ is 3. Seen as a whole, the plots of the complete bipartite graphs seem to form what looks like a staircase.

Interestingly, the star graph has better algebraic connectivity than the edge-augmentation algorithm when 50 edges are present. This observation should not be surprising since the star graph is an early algebraic connectivity maximizer. Still, the standard edge-augmentation algorithm overtakes the one on the star graph starting at 92 edges. This pattern can also be seen in the case when $K_{2,48}$ and $K_{3,47}$ are used as a base. These methods are overtaken by the standard edge-augmentation algorithm starting at 125 and 158 edges, respectively.

These observations indicate that complete bipartite graphs have higher algebraic connectivity than the graphs created by the edge-augmentation algorithm for graphs of certain sizes. Nevertheless, this is not true starting at $K_{4,46}$, where the edge-augmentation algorithm is best overall. In summary, it seems like the underlying structure of the complete bipartite graph has advantages in the short-term only. Table 6.2 summarizes this information.

Algebraic Connectivity of Complete Bipartite Graphs			
Bipartition	Initial AC	Final AC	Difference
1,49	1	5.8242	4.8242
2,48	2	5.2735	3.2735
3,47	3	4.7602	1.7602
4,46	4	4.4336	0.4336
5,45	5	5	0.0000

Table 6.2: Comparison between initial and final algebraic connectivity amongst complete bipartite graphs.

We have now seen that starting the edge-augmentation on an empty graph results in graphs with the highest algebraic connectivity. This is due to the algorithm's simplicity and because an empty graph allows it to add all of the strongest edges. On the other hand, complete bipartite graphs are too structurally rigid while random regular graphs are computationally complex to build. We have also seen that the addition of some fixed random edges to our graph only hinders the edge-augmentation algorithm. Still, an algorithm relying exclusively on the Fiedler vector may not be optimal. The next chapter explores whether the addition of strongest edges and deletion of weakest edges could improve edge augmentation.

Chapter 7

Edge Addition and Deletion

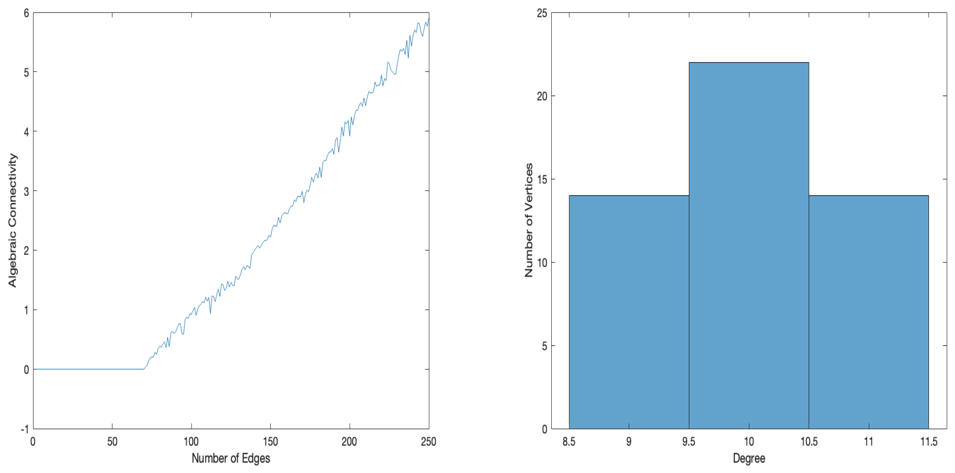
In this chapter, we propose that some of the edges created by the edge-augmentation algorithm may not be optimal. That is, that some edges may not significantly increase algebraic connectivity. With respect to the Fiedler vector, those edges correspond to the absolute value of their entry difference being a minimum amongst all pairs of vertices. Thus, deleting them could increase the final graph's algebraic connectivity. Similarly, only the strongest edges should be added. These are the edges that increase the algebraic connectivity the most. For the algorithm using the Fiedler vector, the absolute value of their entry difference should be maximized as in the standard edge-augmentation algorithm.

We begin by looking at using the Fiedler vector exclusively. Then, we explore this concept using all possible edges. We also consider their combination, either by adding using the Fiedler vector and deleting using brute force or by adding using brute force and deleting using the Fiedler vector. Finally, we conclude with a comparison of these new algorithms.

7.1 Using the Fiedler Vector

As seen in Chapter 5, algorithms that take the Fiedler vector into account outperform those that omit it. This section's method aims to improve the edge-augmentation algorithm by adding and deleting edges according to this vector. In fact, both algorithms use the same method to add edges. That is, both look to optimize the absolute difference amongst all pairs of vector entries. The number of edges added and deleted can be changed, but we focus on adding two edges and deleting a single one to ease comparison to the edge-augmentation algorithm. In total, one edge is added to the graph after each iteration. In this new algorithm, a difference matrix and a sorted adjacency list need to be considered. The difference matrix indicates which edges are the weakest or strongest. The adjacency list is then sorted according to this matrix. The function of the adjacency list is to classify whether or not an edge is already present between the different pairs of vertices. Thus, this list ensures that adding an edge where there is already one or deleting an edge that does not exist is avoided. Hence, the generation of a simple graph at the end is guaranteed. It is important to note that the newly created edges may not be the ones with the greatest absolute difference if those edges already exist in the graph. Instead, they are chosen amongst those that are not yet present. Similarly, the weakest edge must be in the graph in order to be deleted. Thus, this algorithm must take this adjacency list into consideration.

Figure 7.1a shows the resulting plot of this algorithm. Although the overall shape looks quite similar to the edge-augmentation algorithm, the curve is much less smooth. Instead, the plot exhibits more of a spiky behaviour where the algebraic connectivity rises and falls after the addition of nearly each edge. This may result from the deletion of the weakest edge, although this is unlikely since one edge is still being added after each iteration. In fact, it appears that edge deletion decreases the final graph's algebraic connectivity. In the end, a value of 5.9090, 8.9 percent worse than using edge addition exclusively, is reached.



(a) Algebraic Connectivity Plot (b) Degree Histogram

Figure 7.1: Algebraic connectivity plot and degree histogram of the algorithm adding and deleting edges according to the Fiedler vector.

A histogram representing the final degree distribution can be seen in Figure 7.1b. Even after considering edge deletion, this graph's degree distribution closely resembles that of the edge-augmentation algorithm. However, the degrees are more spread out between the three values than in the original method. There are more vertices with degree 9 or 11 and slightly less vertices with degree 10.

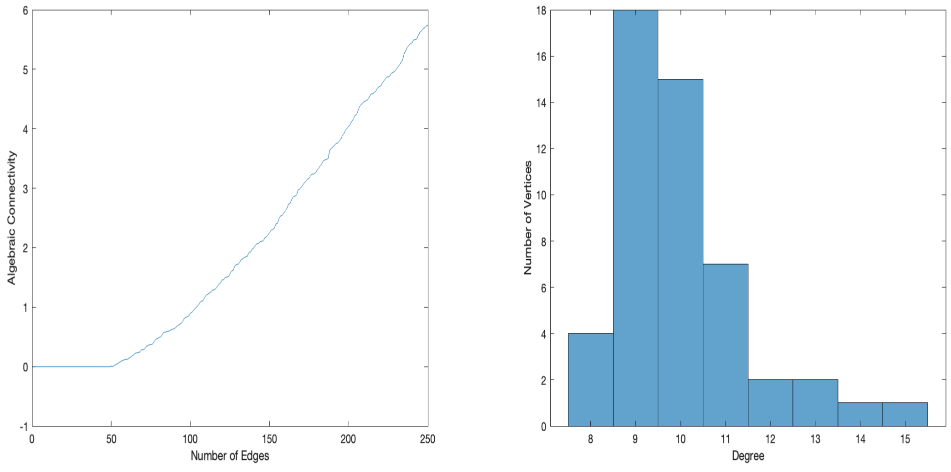
We now know that the inclusion of a deletion step in the edge-augmentation algorithm does not improve it. We are inclined to conclude that the edge-augmentation algorithm chooses every edge for a reason, which may be associated with regularity. This algorithm is also more computationally expensive than the standard one. In fact, it runs in cubic time. In Big O notation, this corresponds to $O(n^3)$ instead of $O(n^2log(n))$. Thus, it also takes much longer to execute as the number of vertices gets large. Next, we explore whether considering all possible edges for addition and deletion can improve our results.

7.2 Using Brute Force

Brute force search was first introduced in section 5.4. Although it resulted in graphs with lower final algebraic connectivity than the ones created by the edge-augmentation algorithm, brute force search may benefit from edge deletion since it may balance out the overall edges in the graph. Therefore, we consider edge addition and deletion separately.

In order to determine which edges to add, each of the absent edges are tested individually. The algebraic connectivity of the original graph is first found. Then, a new edge is added and the algebraic connectivity of this new graph is determined. In theory, it should be greater than the previous one since an edge was added and so the graph is more connected. Their difference in algebraic connectivity is then stored in a list. Finally, the same edge is deleted to return the initial graph. This procedure is repeated for each of the absent edges. The two pairs of entries with the largest differences in the sorted list are then chosen as the edges to be added to the graph.

A similar process is used to determine which edge is to be deleted. This time, however, all of the edges currently in the graph are examined. First, the graph's original algebraic connectivity is computed. Then, one of its edges is deleted and the resulting algebraic connectivity is found. The next step is to determine their difference and store it in a list. Finally, the edge is added back into the graph to reinitialize it. From the sorted deletion list, the pair with the difference that minimizes the change in algebraic connectivity is chosen as the edge to be deleted from the graph. Note that, as opposed to Fiedler vector addition and deletion, this new algorithm does not need an adjacency list to check whether these edges can be added or deleted. Instead, this is done directly by checking the adjacency matrix one entry at the time.



(a) Algebraic Connectivity Plot (b) Degree Histogram

Figure 7.2: Algebraic connectivity plot and degree histogram of the brute force addition and deletion algorithm.

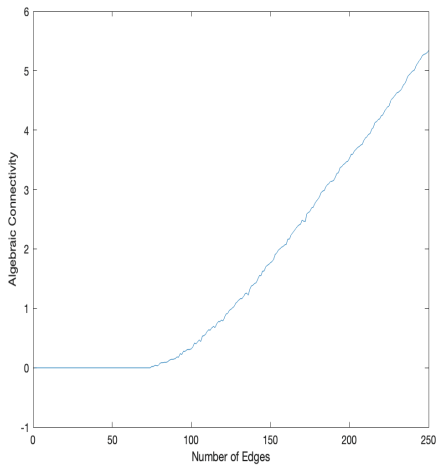
Figure 7.2a shows the algebraic connectivity plot of this algorithm. We can see that the algebraic connectivity begins increasing as soon as 49 edges are in the graph. Another observation is the absence of plateaus. When including both the addition and deletion of edges through brute force search, the plot smoothens. In the end, an algebraic connectivity of 5.7359 is observed, an improvement of 14 percent over the original brute force search algorithm.

As can be seen in Figure 7.2b, the degree distribution of the final graph is affected by edge deletion. In Chapter 5, we noted that the resulting degree distributions were fairly symmetric. However, this is not the case in this new algorithm as the histogram appears to be skewed to the right. Most vertices are still centralized in the degree 9 or 10 range, but several have higher degree.

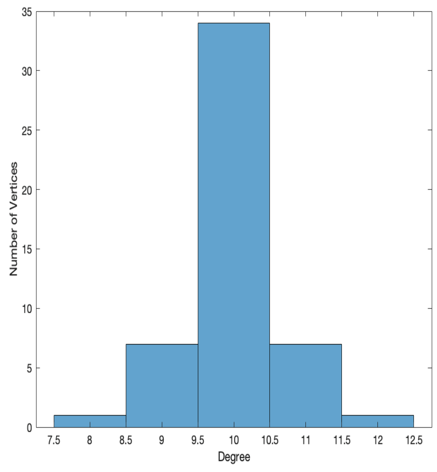
When compared, both of these new algorithms return a nearly equal algebraic connectivity. This fact motivates the exploration of what occurs when the two algorithms are combined.

7.3 Combining the Fiedler Vector and Brute Force

Figure 7.3a shows the result of the Fiedler vector addition and brute force deletion algorithm. The plot indicates that a minimum of 65 edges is still required to connect the graph. The shape of the curve is consistent with the previous two algorithms, although the transition is smoother than simply using the Fiedler vector. The spikes present in that algorithm are also less apparent. In the end, a final algebraic connectivity of 5.3372 is reached.



(a) Algebraic Connectivity Plot

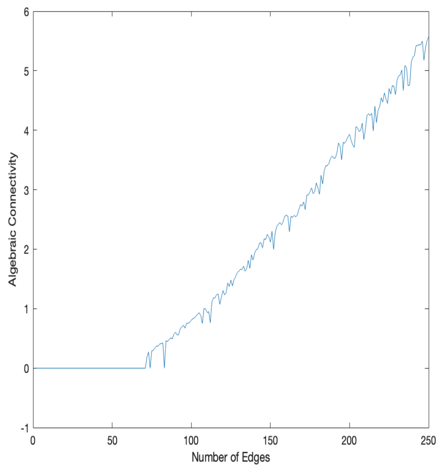


(b) Degree Histogram

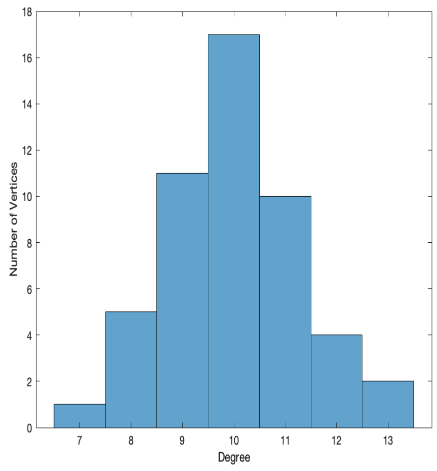
Figure 7.3: Algebraic connectivity plot and degree histogram of the Fiedler vector addition and brute force deletion algorithm on graphs with 50 vertices and 250 edges.

The final graph's degree distribution is shown in Figure 7.3b. The resulting distribution is symmetric with most of the vertices having degree 10. An equal amount of vertices have degree 8 or 9 and 11 or 12.

The other option is brute force addition and Fiedler vector deletion. The output of this algorithm is shown in Figure 7.4. From the plot on the left, it is apparent that the change in algebraic connectivity is much less smooth. The spikes that were present in the Fiedler vector algorithm resurface here and adding edges through brute force does not have an impact on this feature. In the end, a graph with an algebraic connectivity of 5.5679 is obtained. This is 4.1 percent better than Fiedler addition and brute force deletion. Still, the combination of the two methods results in graphs with worse algebraic connectivity than using either the Fiedler vector or brute force exclusively.



(a) Algebraic Connectivity Plot



(b) Degree Histogram

Figure 7.4: Algebraic connectivity plot and degree histogram of the brute force addition and Fiedler vector deletion algorithm on graphs with 50 vertices and 250 edges.

The degree histogram on the right is also worth examining. Note that its shape is also symmetric and that it has a slightly wider range in degrees. One single vertex has a minimum degree of 7 and two vertices have a maximum degree of 13. However, most of the vertices still have degree 10.

Although these algorithms introduce the deletion of edges, it is still pertinent to compare them to the edge-augmentation algorithm since it can be used as a threshold to judge efficacy. The three algorithms on the same set of axes can be seen in Figure 7.5. Note that, as expected, the edge-augmentation algorithm does best overall. Brute force addition and Fiedler deletion comes in second place, although the final gap is quite apparent. Numerically, the Fiedler vector addition and brute force deletion algorithm is 17.7 percent worse and the brute force addition and Fiedler vector deletion is 14.2 percent worse than the standard edge-augmentation algorithm. In terms of algebraic connectivity between the two addition and deletion algorithms, it appears that there is a wider difference at the beginning than at the end. In fact, the two algorithms cross several times once only a few edges remain to be added. We finish by investigating the local effect of deleting an edge by brute force search on the edge-augmentation algorithm.

7.3.1 An Improved Algebraic Connectivity

In Chapter 5, we determined that the edge-augmentation algorithm returns graphs with the highest overall algebraic connectivity overall. Furthermore, in this chapter, we concluded that the combination of this algorithm with the one using brute force does not result in an improvement. Yet, this is only true on a global level. That is, the previously considered algorithms added and deleted edges at each iteration. We propose that a higher algebraic connectivity can be reached by deleting the last edge added by the edge-augmentation algorithm using brute force before adding the truly final edge to the graph.

Compared to the previous methods, this procedure does not increase the number of edges in the graph. In fact, the number of edges remains constant as one edge is deleted and replaced by a second one. Thus, when compared to the other methods in this chapter, it may increase the algebraic connectivity of an existing graph without increasing its size. Hence, it cannot be used to create larger graphs.

From Chapter 5, recall that the algebraic connectivity of the graph obtained by the edge-augmentation algorithm was 6.4878. Interestingly, deleting the last edge by brute force does not decrease this value. That is, the algebraic connectivity of the graph on 249 edges is the same as the one for 250 edges. Yet, when the edge-augmentation algorithm is allowed to add another edge, the final algebraic connectivity is 6.5055. This is an improvement of 0.27 percent for a graph with the same size. Also note that this value is the same as the value obtained by the edge-augmentation itself for a graph with a total of 251 edges. Thus, deleting an edge by brute force before adding the final one has a small positive impact on the final algebraic connectivity of a graph. We conclude this thesis with a brief summary and possible directions for future work.

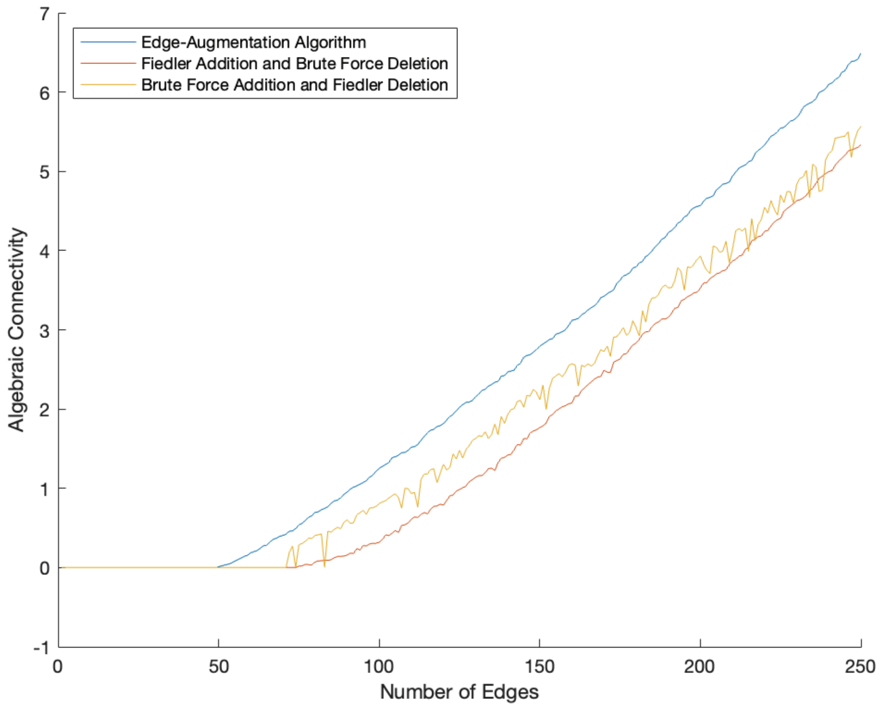


Figure 7.5: Comparison between the edge-augmentation algorithm, the Fiedler vector addition and brute force deletion algorithm, and the brute force addition and Fiedler vector deletion algorithm for graphs with 50 vertices and 250 total edges.

Chapter 8

Conclusion

This thesis was divided into two parts. In the first part, we began by considering cubic graphs with perfect root-connected binary trees as subgraphs. Through some observations, we established a relationship between their diameter and the height of their subgraphs. A necessary condition for a cubic graph to contain a root-connected binary tree as a subgraph was also provided. We then proceeded to generalize Kolokolnikov's bound, valid for these cubic graphs, to hold for any d-regular graph containing perfect root-connected s-ary trees as subgraphs. We also proposed that this upper bound is most likely to be achieved by cage graphs.

Based on a specific configuration of the leaf vertices of the perfect root-connected binary trees, we derived a two term asymptotic estimate for an eigenvalue of the Laplacian matrix of the resulting cubic graph. Through numerical evidence, this estimate was shown to be applicable to their algebraic connectivity. It was also shown that this leaf configuration resulted in graphs with an algebraic connectivity quickly tending to zero as the height of the trees increased.

We then considered cubic graphs with root-connected binary trees having leaves at two consecutive levels. These graphs could also be thought of as level binary trees with a root of degree 3. A new upper bound for the algebraic connectivity of these graphs involving the smallest root of a function containing sine was then established. The Petersen graph, or (3,5)-cage, was the only graph found to achieve this bound. As was done for the first bound, generalizing this bound to hold for any d-regular graph in the future would be insightful.

In Chapter 3, our interest turned to a new graph family, which we named the necklace graphs. The full spectrum of the Laplacian matrix of general necklace graphs was found in terms of the union with multiplicity of the eigenvalues of matrices involving roots of unity. Its Fiedler vectors and thus algebraic connectivity were proved to be associated to the first root of unity.

We then focused our attention to a specific family of d-regular necklace graphs. Namely, necklace graphs formed by c copies of the complete graph $K_{d+1}\backslash\{u, v\}$ with the edge $\{u, v\}$ removed. The complete spectrum of their Laplacian matrix as well as an expression for their algebraic connectivity was established. A first term asymptotic estimate applicable to their algebraic connectivity was also found. Numerical values for their algebraic connectivity were then provided. The spectrum of the Laplacian matrix of other necklace graph families could be explored in the future.

In the fourth chapter, we explored Guiduli's work on path-like cubic graphs. His work showed that these graphs have minimal algebraic connectivity amongst all cubic graphs of the same order. In this chapter, we generalized the cubic case to odd degree d-regular graphs and gave steps for their construction.

To do so, we first defined hourglass graphs, a graph family derived from the path-like graphs. In a proposition, we showed that d-regular hourglass graphs must have odd degree and an even number of vertices. An upper bound for their algebraic connectivity was provided as well as a conjecture that these graphs minimize algebraic connectivity amongst all regular graphs of order $n = 4d - 2$.

We then extended these results to path-like graphs with one middle block consisting of the graph $K_{d+1}\backslash\{u, v\}$. As for regular hourglass graphs, we found an upper bound for their algebraic connectivity. Supported by Guiduli's work, we conjectured that these graphs minimize algebraic connectivity amongst all d-regular graphs of order $n = (4d - 2) + (b - 2)(d + 1)$. These conjectures could be examined in future research. Furthermore, the algebraic connectivity of these graphs in terms of the number of blocks could be addressed.

The second part of the thesis began in the fifth chapter. The purpose of this part of the thesis was to find an algorithm that constructs graphs with high algebraic connectivity in a short amount of time. We introduced four algorithms on empty graphs—namely, random edge addition, the Achlioptas process, the edge-augmentation algorithm, and brute force search. Two of them, random edge addition and the Achlioptas process, relied on randomness, while the other two, the edge-augmentation algorithm and brute force search, were more systematic. In the end, we found that the edge-augmentation algorithm produced graphs with overall highest algebraic connectivity.

In the sixth chapter, we considered different graph families as the starting point for the edge-augmentation algorithm. Random graphs, which we examined in the previous chapter, as well as random regular graphs and complete bipartite graphs were discussed. We concluded that, when compared to starting the edge-augmentation algorithm on an empty graph, none of these performed as well.

In the seventh chapter, we explored how the feature of edge deletion impacts algebraic connectivity. We investigated this notion using two different criteria: the Fiedler vector and brute force search. We also combined these two algorithms to observe whether they produced graphs with overall higher algebraic connectivity. In the end, we found a method that improved the edge-augmentation algorithm by only 0.27 percent. Thus, we concluded that edge deletion does not have a significant positive impact on the algebraic connectivity of graphs.

We may now conclude that using the edge-augmentation algorithm on an empty graph with no edges being deleted is likely optimal for creating graphs with high algebraic connectivity. Still, this thesis only covered a finite number of options to increase algebraic connectivity in simple graphs and more methods may exist.

Using plots and histograms, we were able to find common features in between all algorithms involving edge-augmentation. Namely, the final graphs seemed to have a symmetric degree distribution with most of the vertices being centralized around a certain degree. In our case, most of the vertices seemed to have degree 10. The graph on 50 vertices and 250 edges obtained by the edge-augmentation algorithm is shown in Figure 8.1. In the future, finding a general graph family with these features would certainly be interesting.

We end this thesis with Table 8.1, which summarizes the data for all algorithms starting on an empty graph. It applies to graphs with 50 vertices and 250 total edges exclusively. The average algebraic connectivity out of either 100 or 1000 trials was taken for the methods involving randomness. For convenience, the table is organized by highest to lowest final algebraic connectivity.

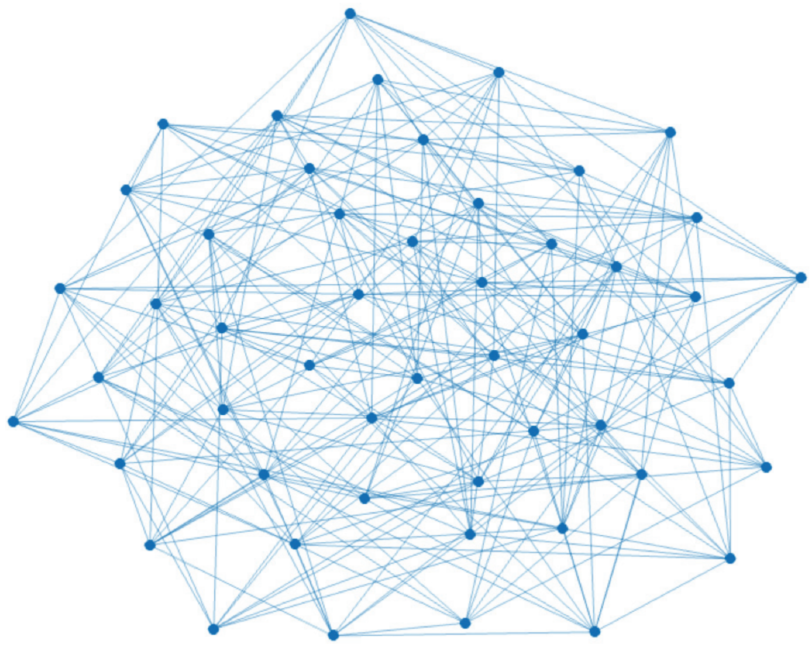


Figure 8.1: The graph obtained by the edge-augmentation algorithm with 50 vertices and 250 edges.

Summary of Algorithms Increasing Algebraic Connectivity	
Algorithm	Final Algebraic Connectivity
Edge-Augmentation Algorithm	6.4878
Random Tree Augmentation	6.3858
Random Edge Addition & Augmentation	6.3084
Fiedler Vector Addition and Deletion	5.9090
Brute Force Addition and Deletion	5.7359
Brute Force Addition, Fiedler Deletion	5.5679
Fiedler Addition, Brute Force Deletion	5.3372
Achlioptas Process (60 Choice)	5.2732
Achlioptas Process (120 Choice)	5.2599
Achlioptas Process (250 Choice)	5.2553
Brute Force Search	4.9138
Augmentation & Random Edge Addition	4.2445
Random Edge Addition	3.3280

Table 8.1: Summary of algorithms increasing algebraic connectivity from an originally empty graph.

Bibliography

[1] W. Watkins. The laplacian quadratic form and edge connectivity of a graph. *Electronic Journal of Linear Algebra*, 30:437–442, 2015.

[2] R. Merris. Laplacian matrices of graphs: A survey. *Linear Algebra and its Applications*, 198:143–176, 1994.

[3] D. K. Hammond, Y. Gur, and C. R. Johnson. Graph diffusion distance : A difference measure for weighted graphs based on the graph laplacian exponential kernel. *IEEE Global Conference on Signal and Information Processing*, pages 1–4, 2013.

[4] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23:298–305, 1973.

[5] N. M. M. de Abreu. Old and new results on algebraic connectivity of graphs. *Linear Algebra and its Applications*, 423:53–73, 2007.

[6] M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, 25:619–633, 1975.

[7] X. Li, M. Chen, H. Su, and C. Li. Distributed bounds on the algebraic connectivity of graphs with application to agent networks. *IEEE Transactions on Cybernetics*, 47:2121–2131, 2017.

[8] A. Ghosh and S. Boyd. Upper bounds on algebraic connectivity via convex optimization. *Linear Algebra and its Applications*, 418:693–707, 2006.

[9] Z. Stanic. Lower bounds for the algebraic connectivity of graphs with specified subgraphs. *Electronic Journal of Graph Theory and Applications*, 9:257–263, 2021.

[10] A. A. Rad, M. Jalili, and M. Hasler. A lower bound for algebraic connectivity based on the connection-graph-stability method. *Linear Algebra and its Applications*, 435:186–192, 2011.

[11] X. Jing, Y. Fan, and Y. Tan. Lower bounds for algebraic connectivity of graphs in terms of matching number or edge covering number. *Ars Combinatoria - Waterloo then Winnipeg*, pages 1–8, 2014.

[12] M. Aouchiche, P. Hansen, and D. Stevanović. A sharp upper bound on algebraic connectivity using domination number. *Linear Algebra and its Applications*, 432:2879–2893, 2010.

[13] B. Mohar. Isoperimetric numbers of graphs. *Journal of Combinatorial Theory*, 47:274–291, 1989.

[14] R. Montenegro and P. Tetali. Mathematical aspects of mixing times in markov chains. *Foundations and Trends in Theoretical Computer Science*, 1:237–354, 2006.

[15] S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society, New Series*, 43:439–561, 2006.

[16] M. Ram Murty. Ramanujan graphs. *Journal of the Ramanujan Mathematical Society*, 18:1–20, 2003.

[17] J. F. Lutzeyer and A. T. Walden. Comparing graph spectra of adjacency and laplacian matrices. *IEEE*, pages 1–14, 2017.

[18] B. Nica. *A Brief Introduction to Spectral Graph Theory.* European Mathematical Society, 2010.

[19] S. A. Cakiroglu. An upper bound on the algebraic connectivity of regular graphs. pages 1–10, 2019.

[20] A. Abiad, B. Brimkov, X. Martinez-Rivera, S. O, and J. Zhang. Spectral bounds for the connectivity of regular graphs with given order. pages 1–24, 2018.

[21] T. Kolokolnikov. Maximizing algebraic connectivity for certain families of graphs. *Linear Algebra and its Applications*, 471:122–140, 2015.

[22] B. Guiduli. The structure of trivalent graphs with minimal eigenvalue gap. *Journal of Algebraic Combinatorics*, 6:321–329, 1997.

[23] C. Brand, B. Guiduli, and W. Imrich. Characterization of trivalent graphs with minimal eigenvalue gap. *Croatica Chemica Acta*, 80:193–201, 2007.

[24] M. Abdi, E. Ghorbani, and W. Imrich. Regular graphs with minimum spectral gap. *European Journal of Combinatorics*, 95:1–18, 2021.

[25] M. Abdi and E. Ghorbani. Quartic graphs with minimum spectral gap, 2020.

[26] A. Nilli. Tight estimates for eigenvalues of regular graphs. *The electronic journal of combinatorics*, 11:1–4, 2004.

[27] B. D. McKay. The expected eigenvalue distribution of a large regular graph. *Linear Algebra and its Applications*, 40:203–216, 1981.

[28] G. Exoo and R. Jaycay. Dynamic cage survey. *The Electronic Journal of Combinatorics*, 2013.

[29] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8:261–277, 1998.

[30] B. Bollobás and E. Szemerédi. Girth of sparse graphs. *Journal of Graph Theory*, 39:194–200, 2002.

[31] S. Belhaiza, N. M. M. de Abreu, P. Hansen, and C. S. Oliveira. *Variable Neighborhood Search for Extremal Graphs. XI. Bounds on Algebraic Connectivity.* Springer, 2005.

[32] A. T. Tiruneh. A simplified expression for the solution of cubic polynomial equations using function evaluation. pages 1–10, 2020.

[33] R.W.D. Nickalls. Viète, descartes and the cubic equation. *The Mathematical Gazette*, 90:203–208, 2006.

[34] P. Erdos and A. Renyi. On the evolution of random graphs. *Mathematical Institute of the Hungarian Academy of Sciences*, 5:17–61, 1960.

[35] O. Riordan and L. Warnke. Achlioptas process phase transitions are continuous. *The Annals of Applied Probability*, 22:1450–1464, 2012.

[36] M. Kang and K. Panagiotou. On the connectivity threshold of achlioptas processes. pages 291–304, 2014.

[37] A. Ghosh and S. Boyd. Growing well-connected graphs. *45th IEEE Conference on Decision and Control*, pages 6605–6611, 2006.

[38] Z. Lin and L. Miao. Upper bounds on the algebraic connectivity of graphs. *Electronic Journal of Linear Algebra*, 38:77–84, 2022.

[39] D. Mosk-Aoyama. Maximum algebraic connectivity augmentation is np-hard. *Operations Research Letters*, 36:677–679, 2008.

[40] G. Li, Z. F. Hao, H. Huang, and H. Wei. Maximizing algebraic connectivity via minimum degree and maximum distance. *IEEE Access*, 6:41249–41255, 2018.

[41] A. Steger and N. C. Wormald. Generating random regular graphs quickly. *Combinatorics, Probability and Computing*, 8:377–396, 1999.

[42] K. Ogiwara, T. Fukami, and N. Takahashi. Maximizing algebraic connectivity in the space of graphs with fixed number of vertices and edges. *IEEE Transactions on Control of Network Systems*, 4:359–368, 2017.

Appendix A

Matlab M Files

This appendix presents the code behind the four algorithms discussed in Chapter 5.

A.1 Random Edge Addition

This algorithm creates a graph by adding valid random edges to an originally empty graph.

```
% Initialize variables
count = 1;
alg_conn_plot = [];

% Specify the number of vertices
n = 50;

% Create an n x n empty graph
A = zeros(n);

% Loop while more edges need to be added
while count <= 250

    % Choose two integers from 1 to n at random
    u = randi(n);
    v = randi(n);

    % Check whether the edge is allowed
    if u ~= v && A(u,v) == 0 && A(v,u) == 0
```

```
        % Add the edge to the graph
        A(u,v) = 1;

        A(v,u) = 1;
        % Find the algebraic connectivity
        D = diag(sum(A));
        L = D - A;
        [V, D] = eig(L);
        e = diag(D);
        [o1,o2] = sort(e);
        alg_conn = o1(2);
        % Store the algebraic connectivity
        alg_conn_plot(end+1,:) = alg_conn;
        % Increase the count
        count = count + 1;
    else
        % Repeat with another edge
        continue;
    end
end

% Return the final algebraic connectivity
final_alg_conn = alg_conn_plot(end);
```

A.2 The Achlioptas Process

This algorithm begins by choosing several potential edges at random. Then, out of these candidates, the one which increases the algebraic connectivity the most gets added to the graph.

```
% Initialize variables
pot_edges = [];
alg_conn_list = [];
alg_conn_plot = [];

% Specify the number of vertices
n = 50;

% Create two n x n empty graphs
A = zeros(n);
B = zeros(n);

% Select the number of candidate edges
count = 1;
max_count = 60;

% Loop until the desired number of edges are in the graph
for i=1:250

    % Choose valid candidate edges at random
    while count <= max_count
        u = randi(n);
        v = randi(n);
        if u ~= v && A(u,v) == 0 && A(v,u) == 0 ...
         && B(u,v) == 0 && B(v,u) == 0
            pot_edges(end+1,:) = [u v];
            B(u,v) = 1;
            B(v,u) = 1;
            count = count + 1;
        end
    end
```

```
% Test each potential edge
for i=1:numel(pot_edges(:,1))
    A(pot_edges(i,1),pot_edges(i,2)) = 1;
    A(pot_edges(i,2),pot_edges(i,1)) = 1;
    D = diag(sum(A));
    L = D - A;
    [V, D] = eig(L);
    e = diag(D);
    [o1,o2] = sort(e);
    alg_conn = o1(2);
    alg_conn_list(end+1,:) = [alg_conn,pot_edges(i,1),...
    pot_edges(i,2)];
    A(pot_edges(i,1),pot_edges(i,2)) = 0;
    A(pot_edges(i,2),pot_edges(i,1)) = 0;
end

% Sort rows from smallest to biggest
alg_conn_list = sortrows(alg_conn_list,1);

% Choose the edge which most augments algebraic connectivity
alg_conn_plot(end+1,:) = alg_conn_list(end,1);

% Add that edge to the graph
A(alg_conn_list(end,2),alg_conn_list(end,3)) = 1;
A(alg_conn_list(end,3),alg_conn_list(end,2)) = 1;

% Reset variables
count = 1;
pot_edges = [];
alg_conn_list = [];
B = zeros(n);
```

```
end

% Return the final algebraic connectivity
final_alg_conn = alg_conn_plot(end);
```

A.3 The Edge-Augmentation Algorithm

This algorithm adds the edge which maximizes the Fiedler vector at each iteration.

```
% Initialize variables
alg_conn_plot = [];

% Specify the number of vertices
n = 50;

% Create an n x n empty graph
A = zeros(n);

% Loop until the desired number of edges are in the graph
for i=1:250

    % Find the algebraic connectivity
    D = diag(sum(A));
    L = D - A;
    [V, D] = eig(L);
    e = diag(D);
    [o1, o2] = sort(e);
    alg_conn = o1(2);

    % Store the algebraic connectivity
    alg_conn_plot(end+1,:) = alg_conn;

    % Find and sort the Fiedler vector
    fiedler = V(:, 2);
    [o1, o2] = sort(fiedler);

    % Find the maximum and minimum values
    maximum = o1(end);
    minimum = o1(1);

    % Find their indices
    index_max = o2(end);
    index_min = o2(1);

    % Add the edge to the graph
    A(index_max, index_min) = 1;
    A(index_min, index_max) = 1;

end

% Return the final algebraic connectivity
final_alg_conn = alg_conn_plot(end);
```

A.4 Brute Force Search

This algorithm looks through all of the edges currently not in the graph and adds the one which increases algebraic connectivity the most at each iteration.

```
% Initialize variables
alg_conn_plot = [];

% Specify the number of vertices
n = 50;

% Create an n x n empty graph
A = zeros(n);
```

```
% Loop until the desired number of edges is in the graph
for i=1:49

    % Find the algebraic connectivity
    D = diag(sum(A));
    L = D - A;
    [V, D] = eig(L);
    e = diag(D);
    [o1, o2] = sort(e);
    alg_conn = o1(2);

    % Store the algebraic connectivity
    alg_conn_plot(end+1,:) = alg_conn;

    % Find and sort the Fiedler vector
    fiedler = V(:, 2);
    [o1, o2] = sort(fiedler);

    % Find the maximum and minimum values
    maximum = o1(end);
    minimum = o1(1);

    % Find their indices
    index_max = o2(end);
    index_min = o2(1);

    % Add the edge to the graph
    A(index_max, index_min) = 1;
    A(index_min, index_max) = 1;

end
```

```
for i=50:250

    % Initialize and reset variables
    M = [];

    % Find the original algebraic connectivity
    D = diag(sum(A));
    L = D - A;
    [V, D] = eig(L);
    e = diag(D);
    [o1,o2] = sort(e);
    og_alg_conn = o1(2);

    % Store the original algebraic connectivity
    alg_conn_plot(end+1,:) = og_alg_conn;

    % Test each edge
    for i=1:n
        for j=i+1:n
            % If the edge is not in the graph, add it
            if A(i,j) == 0
                A(i,j) = 1;
                A(j,i) = 1;
                % Find the new algebraic connectivity
                D = diag(sum(A));
                L = D - A;
                [V, D] = eig(L);
                e = diag(D);
                [o1,o2] = sort(e);
                alg_conn = o1(2);
                % Find their difference
                M(end+1,:) = [abs(og_alg_conn - alg_conn), i, j];
                % Delete the edge
                A(i,j) = 0;
                A(j,i) = 0;
            end
        end
    end

    M = sortrows(M,1);

    % Add strongest edge
    for i=size(M,1):-1:2
         A(M(i,2), M(i,3)) = 1;
         A(M(i,3), M(i,2)) = 1;
         break;
    end
end

% Return the final algebraic connectivity
final_alg_conn = alg_conn_plot(end)
```