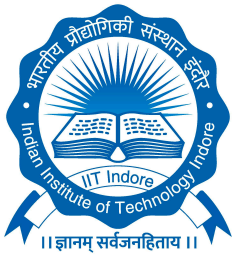
Counting Odd Length Cycles in Regular Graphs

M.Sc. Thesis

by

Visakha Goyal



DEPARTMENT OF MATHEMATICS

INDIAN INSTITUTE OF TECHNOLOGY INDORE

MAY 2025

Counting Odd Length Cycles in Regular Graphs

A THESIS

Submitted in partial fulfillment of the

requirements for the award of the degree

of

Master of Science

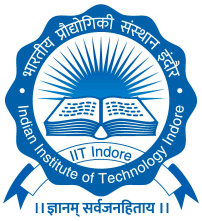
by

Visakha Goyal

(Roll No. **2303141019**)

Under the guidance of

Dr. Ranveer Singh and Dr. Priyamvada



DEPARTMENT OF MATHEMATICS

INDIAN INSTITUTE OF TECHNOLOGY INDORE

MAY 2025

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **Counting Odd Length Cycle in Regular Grpahs** in the partial fulfillment of the requirements for the award of the degree of **Master of Science** and submitted in the **Department of Mathematics**, **Indian Institute of Technology Indore**, is an authentic record of my own work carried out during the time period from August 2024 to May 2025 under the supervision of **Dr. Ranveer Singh**, Assistant Professor, Department of Computer Science & Engineering, and **Dr. Priyamvada**, Assistant Professor, Department of Mathematics, IIT Indore.

The matter presented in this thesis by me has not been submitted for the award of any other degree of this or any other institute. Visakha Goyal

28/05/2025

Signature of the student with date

(**Visakha Goyal**)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

28/05/2025 26/05/2025

Signature of Thesis Supervisor with date

(**Dr. Ranveer Singh and Dr. Priyamvada**)

Visakha Goyal has successfully given his M.Sc. Oral Examination held on 28th May, 2025.

Dr. Ranveer Singh

Signature of supervisor of M.Sc Thesis

Date: 28/05/2025

Signature of Convener, DPGC

Date: 28/05/2025

"Dedicated to My Parents for Everything"

Acknowledgements

I extend my deepest gratitude to my Master's Thesis supervisor,
Dr. Ranveer Singh and Dr. Priyamvada, for their invaluable guidance, unwavering support, and profound mentorship throughout this memorable journey. Dr. Ranveer Singh's & Dr. Priyamvada's expertise in the domain of graph theory has been instrumental in shaping the trajectory of our project. Their profound insights, meticulous attention to detail, and unwavering commitment to excellence have been a constant source of inspiration for me.

Furthermore, I offer my thanks to Prof. Swadesh Kumar Sahoo, Prof. V. Antony Vijesh, Dr. Charitha Cherugondi, Dr. Sourav Mitra, and Dr. Bibekananda Maji for their persistent assistance, motivation, and guidance during my stay at IIT Indore.

I would like to extend my heartfelt gratitude to Dr.Sanjeev Singh, Head of the Department of Mathematics, Dr. Vijay Kumar Sohani, DPGC, Department of Mathematics, and the entire Department of Mathematics for providing excellent facilities and support during my stay at IIT Indore.

I express my sincere appreciation to Mr. Haresh Kumar Jadav,
Mr. Priyanshu Pant members of AAGT Lab, for their significant contributions to this endeavor. As mentors, collaborators, and, most importantly, as elder brothers, their consistent backing, inspiration, and direction were crucial in the development of this research project. Their readiness to exchange thoughts, knowledge, and assets has been priceless. They have undeniably served as the cornerstones of my academic progress.

I would like to extend my thanks to the individuals at the Algorithmic and Algebraic Graph Theory (AAGT) Lab, specifically Mr. Aman Singh, Mr. Virendra Singh and Ms. Surabhi Chakraborty for their ongoing support and priceless advice. My journey in the AAGT Lab has been both exciting and transformative, profoundly enriching my academic and professional development. The motivation and support I have received here are unparalleled,

making my time in the AAGT Lab a pivotal turning point in my academic progress.

I am deeply grateful to my parents for their constant love, unwavering support, and belief in my abilities throughout my academic journey. Their encouragement has been a steadfast pillar, providing me with the strength and motivation to pursue my goals. Additionally, I extend my heartfelt thanks to the special persons I met here, Saurav, Pradeep, Akshat, Prince, Moksha, and all my friends, classmates, and juniors whose fellowship and encouragement have been a continuous source of inspiration. Their support and shared enthusiasm for learning have made this journey not only successful but also memorable and enriching.

Finally, I would like to express my utmost appreciation to the Divine for the countless blessings and opportunities that have been granted to me. It is only through His benevolence that I have been able to achieve all that I have.

Visakha Goyal

Abstract

This thesis explores the enumeration and structural analysis of cycles in graphs, with a particular focus on odd-length cycles in both regular and general graphs. The study is motivated by the combinatorial complexity inherent in cycle structures and their significance in various applications across graph theory and network analysis.

A review of existing literature is presented, covering methods for counting cycles in bipartite graphs and general graphs, as well as highlighting the challenges posed by isomorphism checks in large graph structures. Building on this foundation, the thesis develops and implements an algorithm to generate and extend cycle graphs systematically. This algorithm is applied to count and classify odd-length cycles in regular graphs. Although effective for smaller girth values, the algorithm's scalability is limited due to the exponential growth in the number of graph structures and the computational cost of isomorphism detection.

Further, the thesis investigates the enumeration of odd-length cycles in general graphs and identifies key limitations in using purely combinatorial approaches to derive closed-form expressions for the number of such structures. These challenges point to the need for alternative strategies, including more efficient algorithms or algebraic methods, to extend the current framework.

Overall, the thesis contributes both algorithmic techniques and theoretical insights to the field of cycle enumeration in graph theory, and it lays the groundwork for future research into more scalable and generalizable methods.

Contents

List of Figures

Chapter 1

Introduction

Graphs are fundamental structures in mathematics and computer science, widely used to model relationships in networks, biological systems, communication channels, and more. One of the central problems in graph theory is the identification and enumeration of *cycles*, especially *simple cycles*—closed walks that do not revisit any vertex except the starting point.

Among the many problems involving cycles, *counting cycles of a specific length* is both practically important and theoretically challenging. It is well established that the problem of counting the number of simple cycles of a given length in a general graph is #P-complete, a class of problems even more computationally difficult than those in NP. This complexity holds true even for restricted classes such as *bipartite*, *planar*, or *sparse graphs*, making *cycle counting a computationally intractable task in the general case.*

Despite this inherent hardness, significant progress has been made in *particular classes of graphs* where structure or symmetry can be exploited. One such powerful approach is rooted in *spectral graph theory*, which studies the properties of graphs through the eigenvalues and eigenvectors of matrices associated with them—most notably the *adjacency matrix*. The trace of powers of the adjacency matrix, for example, is closely related to the number of *closed walks* in the graph. In some cases, this connection can be extended to obtain counts of cycles, especially *short cycles* in *regular* or *biregular* graphs.

Notably, in *biregular bipartite graphs*, methods have been developed to compute the number of cycles of even length (equal to the girth and slightly above it) in terms of the degrees of the nodes and the *spectrum* (i.e., the eigenvalues) of the adjacency matrix. This is particularly relevant in *coding theory*, where the cycle structure of *Tanner graphs* directly affects the performance of *low-density parity-check (LDPC) codes*. In such contexts, cycles of length 4,

6, or 8 are often of primary interest because they influence the formation of trapping sets and, thus, the error floor behavior of iterative decoders.

However, the problem becomes even more subtle and complex when we shift focus to *odd-length cycles*. These cycles play a crucial role in various applications such as *detecting non-bipartiteness*, *graph coloring*, *network reliability*, and *social network analysis*. In *regular graphs*—graphs where each vertex has the same degree—odd cycles can be indicative of deeper structural features and often resist enumeration through traditional spectral approaches. This is because many known spectral techniques naturally lend themselves to even-length cycles, which arise more directly in the trace of even powers of the adjacency matrix.

In contrast, *odd-length cycles do not contribute in the same way to matrix powers*, and thus their enumeration poses unique challenges. Nevertheless, recent studies have shown that under certain constraints or with additional spectral information—such as knowledge of walk regularity or the behavior of non-backtracking walks—it is possible to derive estimates or exact counts for small odd-length cycles in *regular or walk-regular graphs*.

Building upon these ideas, the focus of this thesis is to *investigate the presence and enumeration of odd-length cycles in regular graphs* using spectral methods. By leveraging the structure of the adjacency matrix, its spectrum, and traces of its powers—along with insights from recent work in spectral graph theory and algebraic graph theory—this research aims to extend the frontier of tractable cycle counting. Specifically, this work examines how certain properties of regular graphs (such as degree uniformity and symmetry) can be exploited to isolate contributions from odd-length cycles.

This exploration is significant for several reasons:

- From a theoretical perspective, understanding the distribution of odd cycles enriches the structural characterization of regular graphs.

- From an algorithmic point of view, developing methods for efficiently estimating or counting odd cycles can contribute to faster heuristics in applications like *graph coloring*, *community detection*, or *scheduling problems*.

- From a spectral perspective, it deepens our understanding of how eigenvalues encode fine-grained cycle information—not just in general, but specifically with regard to parity and cycle length.

To this end, the thesis will begin with a detailed review of the theoretical foundations of cycle counting and spectral graph theory, followed by a survey of existing results on counting cycles using spectral techniques. The discussion then shifts toward specialized results for regular graphs and known difficulties in capturing odd-length cycles. Finally, original results and computational insights are presented that seek to bridge the gap between the theoretical hardness of general cycle counting and the analytical tractability offered by structured graphs and their spectra.

In summary, although counting cycles in general graphs is a computationally hard problem, this work aims to carve out new possibilities for analytical and semi-analytical methods for *odd-length cycle enumeration* in regular graphs—guided by the powerful lens of *spectral graph theory* and inspired by the promising partial results achieved in related contexts.

1.1 Background

In 2018, Blake and Lin studied the enumeration of short-length cycles in biregular bipartite graphs by constructing walk-generating functions using rooted trees. These rooted trees were constructed in such a way that they were isomorphic to the corresponding biregular bipartite graphs. Based on this isomorphism, walk-generating functions were developed, and a recurrence relation was established to facilitate the counting of short cycles.

In 2019, Dehghan and Banihashemi built upon the work of Blake and Lin (2018), drawing inspiration from their approach to counting short cycles in biregular bipartite graphs. While Blake and Lin focused on short-length cycles near the girth, Dehghan and Banihashemi extended the investigation to cycles of lengths larger than the girth. Instead of relying solely on walk-generating functions, they employed the graph's degree distribution and spectral properties to estimate the number of such cycles. Their work marked a significant step forward in understanding the interplay between a graph's structure and its spectrum, particularly in the context of counting longer cycles in biregular bipartite graphs.

In 2023, Barik and Reddy extended the ideas introduced by Dehghan and Banihashemi to count cycles of specific lengths—namely 8, 10, and 12—in general graphs. Building on the spectral and structural techniques used in earlier works, they developed a method that involved identifying and analyzing all possible subgraphs that could contribute to cycles of the desired lengths. By

examining these subgraphs, they computed the number of closed walks corresponding to those cycle lengths using a combination of degree distributions, adjacency matrices, and the spectral properties of the graphs. Their approach provided a general framework for cycle enumeration in arbitrary graphs and highlighted how algebraic and combinatorial properties can be effectively combined for this purpose.

1.2 Motivation

Cycle counting in graphs is a fundamental problem in graph theory with wide-ranging applications across computer science, communication networks, biology, and chemistry. At its core, the problem involves identifying and counting simple cycles (closed walks with no repeated vertices) of a given length in a graph. This problem is known to be **NP-hard**, meaning there is no known polynomial-time algorithm that can solve it for all graphs. The complexity arises because the number of possible cycles increases exponentially with the increase in size and density of the graph.

Despite its computational difficulty, solving the cycle counting problem is critically important. In *network science*, it helps in analyzing feedback loops and network resilience. In *bioinformatics*, it is used to study molecular structures such as ring compounds in chemical graphs or metabolic pathways in biological networks. In *coding theory*, especially in the analysis of LDPC (Low-Density Parity-Check) codes, short cycles affect decoding performance. Thus, accurately counting cycles of specific lengths (e.g., 4, 6, 8) plays a direct role in improving code design.

Over the years, several researchers have attempted to tackle this challenge through various mathematical and algorithmic approaches. In **2018**, Blake and Lin constructed walk-generating functions using rooted trees that were isomorphic to biregular bipartite graphs. Their method allowed the enumeration of short cycles by establishing recurrence relations. In **2019**, Dehghan and Banihashemi extended this work by using *degree distribution* and *spectral analysis* to estimate the number of longer cycles—those exceeding the girth—in biregular bipartite graphs. Most recently, in **2023**, Barik and Reddy generalized the approach to arbitrary graphs, focusing on cycles of length 8, 10, and 12. They achieved this by analyzing all relevant subgraphs and using *adjacency matrices*, *degree distributions*, and *graph spectra* to count the associated closed walks.

These efforts represent significant progress in an otherwise computationally intractable domain, offering both theoretical insights and practical tools for fields that rely heavily on graph structures.

1.3 Objectives

The central objective of this study is to develop and apply mathematical techniques for determining the number of cycles of **odd length** in **regular graphs**. While extensive research has been conducted on the enumeration of short and even-length cycles—particularly in bipartite and biregular graphs—less attention has been given to the characterization and counting of odd-length cycles in general regular graphs. This gap is significant, as odd cycles play a crucial role in several graph-theoretic properties and have implications in areas such as coloring problems, spectral graph theory, and network stability.

The specific aims of this research include:

- To investigate and formulate methods for detecting and counting odd-length cycles using algebraic and combinatorial tools such as adjacency matrices, walk-generating functions, and spectral techniques.
- To explore the relationship between the spectral properties of regular graphs and the existence of cycles of odd length.
- To analyze the structural constraints in regular graphs that allow or prohibit the presence of odd cycles, particularly in the context of girth and symmetry.
- To extend existing theoretical frameworks that are primarily suited for even-length or bipartite settings to encompass odd-length cycles in more general regular graphs.

1.4 Contributions of the Thesis

This thesis makes several contributions to the study of cycle enumeration in graphs, particularly focusing on counting odd-length cycles and understanding their structural properties in various types of graphs.

In Chapter 2, a comprehensive literature review is presented on the enumeration of cycles in bipartite graphs, including techniques for counting cycles of specific lengths in general graphs. This chapter lays the theoretical foundation and highlights existing approaches and their limitations, providing motivation for the work undertaken in this thesis.

In Chapter 3, we focus on the enumeration of odd-length cycles specifically within regular graphs. An algorithm is developed to generate unique cycle structures, and its computational limitations are discussed in relation to graph isomorphism complexity.

In Chapter 4, initial attempts to count odd-length cycles in general (non-regular) graphs are explored. These preliminary efforts help identify the challenges of applying combinatorial methods to general graphs and motivate the need for more efficient or algebraic approaches.

Together, these chapters contribute to a deeper understanding of the complexity involved in cycle enumeration and highlight potential directions for further research in combinatorial and algebraic graph theory.

Chapter 2

Literature Review

In this chapter, we present a comprehensive study of cycle counting in graphs using spectral graph theory. This is achieved through an extensive literature review of the research conducted to date in this area. We aim to highlight key theoretical developments, methodologies, and applications that have emerged, providing a solid foundation for further exploration in both theoretical and applied contexts.

2.1 Notation and Preliminaries

Let G be a simple, undirected graph with vertex set $V(G)$ and edge set $E(G)$; we write V and E when the graph is clear from context. An edge $e \in E$ connecting vertices u and w is denoted by $\{u, w\}$, or simply by uw. The degree of a vertex $v \in V$, denoted $d(v)$, is the number of edges incident to it.

A *walk* of length k in G is a sequence of vertices $v_1, v_2, \ldots, v_{k+1}$ such that $\{v_i, v_{i+1}\} \in E$ for all $i = 1, \ldots, k$. A walk is a *path* if all its vertices (except possibly the last) are distinct. A walk is a *closed walk* if $v_1 = v_{k+1}$. A *cycle* is a closed walk with all other vertices distinct.

We denote a path with n vertices by P_n, and a cycle of length k by C_k. The number of k-cycles in G is denoted by $N_k = |C_k|$. The *girth* of a graph, denoted g, is the length of its shortest cycle.

The adjacency matrix $A = [a_{ij}]$ of a simple undirected graph G is symmetric, with entries $a_{ij} \in \{0, 1\}$ and $a_{ii} = 0$ for all $i, j \in V$. The eigenvalues $\{\lambda_i\}$ of A form the spectrum of G. For disconnected graphs, the spectrum is the union of the spectra of its components.

A key property of A is that the number of walks of length k between nodes i and j equals the (i, j)-th entry of A^k, denoted $[A^k]_{ij}$. In particular,

$[A^k]_{ii}$ counts the number of closed walks of length k containing node i. The total number of closed walks of length k in G is given by the trace of A^k, $\text{tr}(A^k)$. Since, $\text{tr}(A^k) = \sum_{i=1}^{|V|} \lambda_i^k$, the spectrum enables counting closed walks of different lengths.

2.2 Short Cycle Counting in Biregular Bipartite Graphs

In 2018, Ian F. Blake and Shu Lin developed a formula to count short-length cycles in biregular bipartite graphs. Their work primarily focused on enumerating cycles whose lengths are equal to the girth of the graph. Although the initial objective was to derive expressions for cycles of length $g+2$ as well, the increased computational complexity of this task posed significant challenges. While they were unable to extend their results to such cycles, the authors expressed hope that their efforts would inspire further research in this direction, potentially leading to a more analytical framework for code design than was previously available. Additionally, they provided a closed-form generating function for the number of closed walks in a biregular tree associated with the biregular bipartite graph. This enumeration of closed walks in biregular trees was presented as a contribution of independent interest, offering insights that may have broader implications beyond the immediate scope of their study.

2.3 Cycle Counting in Bipartite Graphs Using the Degree Distribution and the Spectrum of the Graph

Since the walk generating function primarily facilitates the computation of shorter-length cycles, researchers sought more effective methods for identifying longer cycles. Building upon the foundational ideas introduced by Blake and Lin, in 2019, Ali Dehghan and Amir H. Banihashemi developed a novel formula for enumerating cycles of length k in bipartite graphs. Their method utilizes the degree distribution and the spectral properties of the graph, offering a more general and powerful framework for cycle enumeration. This advancement addresses the limitations of previous techniques and provides valuable tools for applications in coding theory and network analysis, where the structure

and frequency of longer cycles play a significant role.

In this section, we compute the multiplicity of k-cycles in bi-regular bipartite graphs with $g \geq 4$, for $g+2 \leq k \leq 2g-2$, in terms of the spectrum and the degree sequences of the graph. The results presented here complement those of Blake and Lin for g-cycles. These results are obtained by characterizing and counting closed walks of length k that are not cycles, and subtracting their multiplicity from the total number of closed walks of length k. The latter can be easily computed using the spectrum of the graph. In this section, we also provide a brief review of the main result of [1], and propose an alternate approach for calculating the number of closed cycle-free walks in a bipartite graph.

2.4 General Formula for $k-$ Length Cycles

For a given (d_v, d_c)-regular bipartite graph G, the number of k-cycles is given by:

$$N_k = \frac{\sum_{i=1}^{|V|} \lambda_i^k - \Omega_k(d_v, d_c, G) - \Psi_k(d_v, d_c, G)}{2k}, \tag{1}$$

where $\{\lambda_i\}_{i=1}^{|V|}$ is the spectrum of G, and $\Omega_k(d_v, d_c, G)$ and $\Psi_k(d_v, d_c, G)$ are the number of closed cycle-free walks of length k and closed walks with a cycle of length k in G, respectively.

The multiplicity $\Omega_k(d_v, d_c, G)$ of closed cycle-free walks of length k in a (d_v, d_c)-regular bipartite graph G was computed in [1].

2.5 Computing Ψ_k

To calculate Ψ_k, we employ the approach based on the degree distribution, as proposed by Ali Dehghan and Amir H. Banihashemi in their work on computing the multiplicity of cycles in bipartite graphs using the degree distribution and the spectrum of the graph [2].

For instance, to calculate the number of cycles of length $g+2$ in biregular bipartite graphs—where g denotes the girth of the graph—the following expression is used:

$$\Psi_{g+2}(d_v, d_c, G) = N_g \times \left[\binom{g}{2} (d_v + d_c) - g \right] \times 2(g+2) \tag{2.1}$$

Here, d_v and d_c represent the degrees of variable and check nodes, re-

spectively, N_g denotes the number of cycles of length g, and G is the graph under consideration.

Chapter 3

Counting Cycles of Odd Length in Regular Graph

In this chapter, we present techniques for counting odd-length cycles using the general formula given in equation (1). Noting that $\Omega_k = 0$ when k is odd [3], the computation simplifies significantly, then we are only required to evaluate the terms Ψ_k.

Let g be the **odd girth** of the graph. We are interested in finding cycles of length $g + l$, where l is even and $g + l < 2g$. In order to identify cycles of length $g + l$, we need to eliminate all closed walks of length $g + l$ that are not simple cycles. That is, we need to compute Ψ_{g+l}. In the following sections, we compute cycles of lengths N_{g+2} and N_{g+4} with the help of examples. Following the same approach, we then attempt to compute N_{g+l}.

3.1 Compute N_{g+2}

To compute the number of cycles of length $g+2$, we must first calculate Ψ_{g+2}.
Let us consider the Petersen graph as an example. It is a 3-regular graph with girth $g = 5$.
We need to compute N_{g+2}, i.e., the number of cycles of length 7 in the Petersen graph, since $g = 5$.
To calculate N_7, we must first compute Ψ_7, as we need to eliminate all closed walks of length 7 with to cycles of length less than 7.

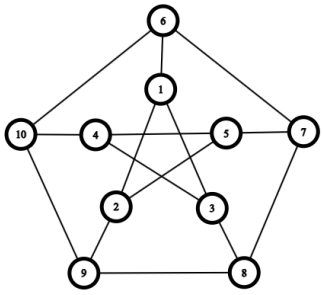


Figure 3.1: Petersen graph

To compute Ψ_7, we first need to identify the structures that account for all closed walks of length 7 which contain cycles of length less than 7. Since $g = 5$, a cycle of length 5 is one such structure that contributes to these walks.

Another structure can be formed by extending the 5-cycle: this is done by adding an extra vertex and connecting it to each vertex of the 5-cycle, one at a time. In each step, the extra vertex is connected to a single vertex of the 5-cycle, and any previous connections are removed.

After accounting for isomorphisms, we obtain a unique graph — a 5-cycle with a pendant edge.

Therefore, the two structures that cover all closed walks of length 7 with cycles of length less than 7 are:

1. The cycle of length 5, and

2. The cycle of length 5 with a pendant edge.

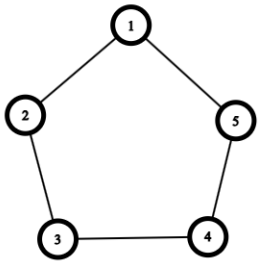


Figure 3.2: F_1

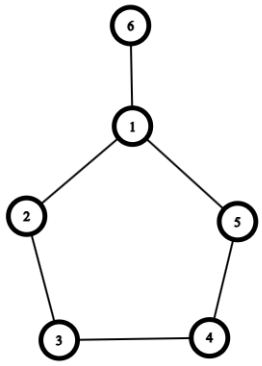


Figure 3.3: F_2

Now, to calculate the number of closed walks of length 7 in F_1, we use the adjacency matrix corresponding to F_1. Let $\text{tr}(F_i^7)$ denote the trace of the adjacency matrix of F_i raised to the power 7. The value of $\text{tr}(F_1^7)$ gives the total number of closed walks of length 7 corresponding to the structure F_1. Similarly, we can calculate the number of closed walks of length 7 corresponding to the structure F_2.

Now, we need to find the number of F_1 and F_2 structures in the graph. To do this, we use a combinatorial approach.
The number of F_1 structures is given by:

$$F_1 = N_g = \frac{\text{tr}(A^g)}{2g}$$

The number of F_2 structures is given by:

$$F_2 = gF_1(d-2)$$

Therefore, $\Psi_7 = F_1(tr(F_1^7)) + F_2(tr(F_2^7) - tr(F_1^7))$
Hence,

$$N_7 = \frac{\sum_{i=1}^{n} \lambda_i^7 - \Psi_7(G)}{2 \times 7}. \tag{3.1}$$

Hence from this example we can generalize and formulate:

$$N_{g+2} = \frac{\sum_{i=1}^{n} \lambda_i^{g+2} - \Psi_{g+2}(G)}{2(g+2)}$$

3.2 Compute N_{g+4}

Continuing with the same example, we now compute N_{g+4}, i.e., the number of cycles of length 9 in the Petersen graph, since $g = 5$.

To calculate N_9, we must first compute Ψ_9, as we need to eliminate all closed walks of length 9 that correspond to cycles of length less than 9.

To compute Ψ_9, we first need to identify the structures that account for all closed walks of length 9 which contain cycles of length less than 9.
The two structures obtained while computing N_{g+2} also contribute to the closed walks of length 9 that contain cycles of length less than 9.

Now, we need to find additional structures that account for the remaining closed walks of length 9. To do this, we extend the last structure used while computing N_{g+2} — the cycle of length 5 with a pendant edge — by adding an extra vertex and connecting it to each vertex of the existing structure, in the same manner as described earlier.

After considering isomorphisms, we obtain three unique structures:

1. A cycle of length 5 with two pendant edges from the same vertex,
2. A cycle of length 5 with two pendant edges from different vertices,
3. A cycle of length 5 with an L-shaped pendant edge.

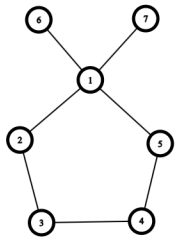


Figure 3.4: F_3

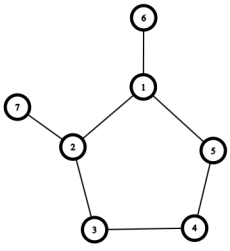


Figure 3.5: F_4

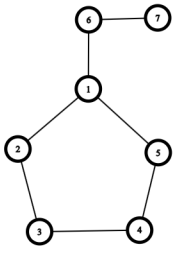


Figure 3.6: F_5

Now, we need to find the number of F_3, F_4, and F_5 structures in the graph. For this, we use a combinatorial approach:

$$F_3 = F_2(d-3)$$

$$F_4 = gF_2(d-2)$$

$$F_5 = F_2(d-1)$$

Since we have already computed N_{g+2}, i.e., N_7, the cycle of length 7 and the cycle of length 7 with a pendant edge will also contribute to covering all closed walks of length 9.

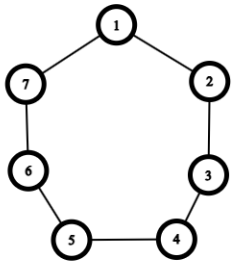


Figure 3.7: F_6

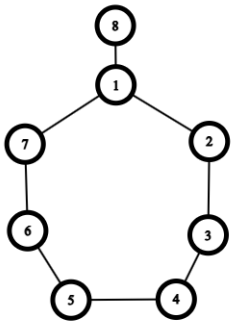


Figure 3.8: F_7

Now, we need to find the number of F_6 and F_7 structures in the graph. We define them as follows:

$$F_6 = N_{g+2}$$

$$F_7 = F_6(d-2)$$

Therefore, the expression for Ψ_9 is given by:

$$\begin{aligned}\Psi_9 = {} & F_1 \cdot \operatorname{tr}(F_1^9) + F_2 \cdot \big(\operatorname{tr}(F_2^9) - \operatorname{tr}(F_1^9)\big) \\ & + F_3 \cdot \big(\operatorname{tr}(F_3^9) - 2\operatorname{tr}(F_2^9) + \operatorname{tr}(F_1^9)\big) \\ & + F_4 \cdot \big(\operatorname{tr}(F_4^9) - 2\operatorname{tr}(F_2^9) + \operatorname{tr}(F_1^9)\big) \\ & + F_5 \cdot \big(\operatorname{tr}(F_5^9) - \operatorname{tr}(F_2^9)\big) \\ & + F_6 \cdot \operatorname{tr}(F_6^9) + F_7 \cdot \big(\operatorname{tr}(F_7^9) - \operatorname{tr}(F_6^9)\big)\end{aligned}$$

Hence,

$$N_9 = \frac{\sum_{i=1}^{n} \lambda_i^9 - \Psi_9(G)}{2 \times 9}. \tag{3.2}$$

Hence from this example we can generalize and formulate:

$$N_{g+4} = \frac{\sum_{i=1}^{n} \lambda_i^{g+4} - \Psi_{g+4}(G)}{2(g+4)}$$

3.3 Compute N_{g+l}

In the previous sections, we computed N_{g+2} and N_{g+4}. What we observed during this computation is that, in order to determine N_{g+2} and N_{g+4}, our main objective was to compute Ψ_{g+2} and Ψ_{g+4}, respectively.

Similarly, to compute N_{g+l}, we need to compute Ψ_{g+l}. For this, we must identify the structures that cover all closed walks of length $g + l$. To find these structures, we start with a cycle of length g, which serves as the base graph that covers closed walks of length $g + l$. From this base graph, we extend the graphs by adding extra vertices accordingly, which yields all the new structures.

Let the maximum number of extra vertices to be added be at most n, where $n = \frac{l}{2}$. So, when we compute structures for Ψ_{g+4}, we get $n = 2$, i.e., we have to add at most 2 extra vertices to the base graph for extension.

In the first step, we add one extra vertex and connect this extra vertex to each vertex of the base graph one at a time—that is, in each step, the extra vertex is connected to a single vertex of the base graph, with any previous connections removed. Then, isomorphism is applied to identify unique structures.

In the next step, the second extra vertex is added (since we have to add up to 2 extra vertices), and in exactly the same manner, connections are made with the structures obtained from the previous step. Again, isomorphism is applied to obtain new unique structures.

Thus, what we have done is: first, we determined the number of extra vertices needed for the extension, and in each step of the extension, we applied isomorphism to obtain unique structures.

Using this idea, we have developed an algorithm to generate unique structures for computing Ψ_{g+l}. The following is the pseudocode for the algorithm.

Algorithm: Generate and Extend Cycle Graphs

1. **Initialize parameters and data structures:**
   - Set initial value of g (e.g., $g = 5$)
   - Create empty lists: `all_extensions`, `new_extension`, `old_extension`
2. **Process base graphs for $g = 5$ with plotting:**

- For `cycle_len` from g to $2g-2$ with step size 2:
  (a) Generate a base cycle graph for the current cycle length
  (b) Update `old_extension` with the previous `new_extension` and the current base graph
  (c) Reset `new_extension` to an empty list
  (d) For each graph in `old_extension`:
    i. Generate all possible single-step extensions
    ii. Add these extensions to the `new_extension` list
  (e) Remove isomorphic duplicates from `new_extension`
  (f) Add the base graph and filtered extensions to `all_extensions`
- Plot all generated graphs

3. **Process graphs for $g = 11$ and print statistics:**
   - For $i = 11$ (with step size 2):
     (a) Set $g = i$
     (b) Reset all lists: `all_extensions`, `new_extension`, `old_extension`
     (c) Repeat steps 2(a)–2(f), using a different isomorphism filter
     (d) Print statistics: value of g, $2g-1$, and the number of graphs generated

Helper Functions:

- `create_cycle_graph(length)`: Returns a cycle graph of the specified length
- `extend_graph_once(graph)`: Returns a list of extended graphs
- `filter_nonisomorphic_original(graphs)`: Filters isomorphic graphs (version 1)
- `filter_nonisomorphic(graphs)`: Filters isomorphic graphs (version 2)

However, this algorithm has certain limitations. As g increases, the time complexity of checking graph isomorphism also increases. Therefore, this algorithm works effectively only up to $g = 9$ or 11. With each increment in g, the number of structures grows exponentially, significantly increasing the time required for isomorphism checks.

Chapter 4

Attempts to Count Odd Length Cycle in General Graphs

Let $G = (V, E)$ be a general graph with n vertices, where V is the vertex set and E is the edge set. For a given graph G, a *walk* of length k is a sequence of vertices $v_1, v_2, \ldots, v_{k+1}$ in V such that $(v_i, v_{i+1}) \in E$ for all $i = 1, 2, \ldots, k$. A walk is called a *closed walk* if $v_1 = v_{k+1}$. A walk $v_1, v_2, \ldots, v_{k+1}$ is a *path* if all the vertices $v_1, v_2, \ldots, v_{k+1}$ are distinct. A path is a *cycle* if $v_1 = v_{k+1}$.

The adjacency matrix of the graph G is $n \times n$ matrix $A = [a_{ij}]$, where $a_{ij} = 1$ if $(v_i, v_j) \in E(G)$, otherwise $a_{ij} = 0$. The set of eigenvalues $\{\lambda_i\}$ of A is called the spectrum of the graph G. The entry $[A^k]_{ij}$ of the matrix A^k gives the number of walks of length k between nodes i and j, while the diagonal entry $[A^k]_{ii}$ indicate the number of closed walks at vertex i. Thus, the trace of A^k, i.e., $\sum_{i=1}^{n} \lambda_i^k$, represents the total number of closed walks of length k in G.

In this chapter, we are interested in finding cycles of length $g + l$, where g is the odd girth of the graph G, l is even, and $g + l < 2g$. To identify cycles of length $g + l$, we need to eliminate all closed walks of length $g + l$ that are not simple cycles. In other words, we need to compute Ψ_{g+l}.

To calculate Ψ_{g+l}, we need to identify the structures in the graph that account for all closed walks of length $g+l$ containing cycles of length less than $g + l$.

Let us explain, with the help of an example, how we identify those structures that cover all closed walks of length $g + l$ which contain cycles of length less than $g + l$.

4.1 Counting $(g+2)$- length Cycle

Let G be a general graph having an odd girth g, then $g+2$ is also odd. Therefore, $\Omega_{(g+2)}(G) = 0$ [3]. Hence the Equation (1) becomes

$$N_{(g+2)} = \frac{\sum_{i=1}^{n} \lambda_i^{(g+2)} - \Psi_{(g+2)}(G)}{2(g+2)} \tag{4.1}$$

4.1.1 Identifying the structures for computing Ψ_{g+2}

To compute Ψ_{g+2}, we need to identify the structures that cover all closed walks of length $g+2$ that include cycles of length less than $g+2$.

Let $g = 5$ be the odd girth of the graph G. We know that the cycle of length g can cover all closed walks of length $g+2$ containing cycles of length less than $g+2$.

In order to identify additional structures that cover all closed walks of length $g+2$ containing cycles of length less than $g+2$, we extend the base graph by adding an extra vertex and connecting it to each vertex of the base graph one at a time—that is, in each step, the extra vertex is connected to a single vertex of the base graph, with any previous connections removed. This process is repeated for all vertices of the base graph, and isomorphism is then used to identify the unique resulting structures that cover all such closed walks of length $g+l$ containing cycles of length less than $g+l$ (in this case, $g+2$). Here, the base graph refers to the structure obtained in the previous extension step: when $g+2k$ with $k=1$, the base graph is the cycle of length g; when $k=2$, the base graph is the cycle of length g with one pendant edge.

4.1.2 Computing Ψ_{g+2}

Following the approach mentioned in subsection 4.1.1 we get the following structures.

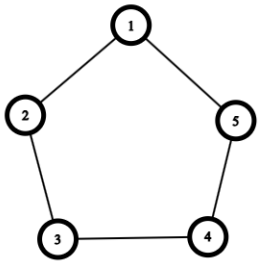


Figure 4.1: F_1

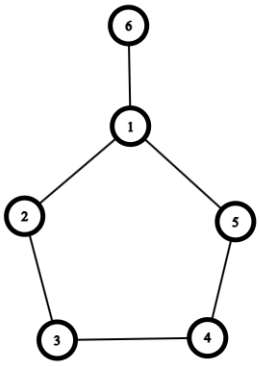


Figure 4.2: F_2

Let $F_1 = N_g = \frac{\text{tr}(A^g)}{2g}$, and let $F_2 = gN_g(d_i - 2)$ represent the number of such structures in the graph, where d_i denote the degree of the i^{th} vertex in graph G. Let $\text{tr}(F_i^{g+l})$ denote the trace of the corresponding adjacency matrix raised to the power $g+l$, which gives the total number of closed walks of length $g + l$ corresponding to the structure F_i.

Therefore, $\Psi_{g+2} = F_1(tr(F_1^{g+2})) + F_2(tr(F_2^{g+2}) - tr(F_1^{g+2}))$.

4.2 Counting of $(g + 4)$-length Cycle

Let G be a graph having an odd girth g, then $g + 4$ is also odd. Therefore, $\Omega_{(g+4)}(G) = 0$. Hence the Equation (1) becomes

$$N_{(g+4)} = \frac{\sum_{i=1}^{n} \lambda_i^{(g+4)} - \Psi_{(g+4)}(G)}{2(g+4)}. \tag{4.2}$$

Now we are left to calculate $\Psi_{(g+4)}(G)$. To calculate $\Psi_{(g+4)}(G)$, we follow the approach as mentioned in subsection 4.1.1 which results in extra three structures.

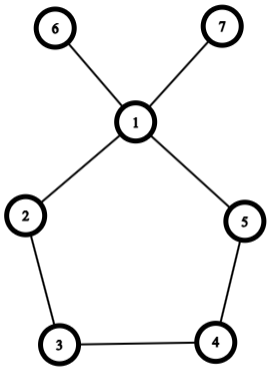


Figure 4.3: F_3

To find a structure like F_3, we can formulate it using $F_3 = F_2(d_i - 3)$ and can calculate closed walk of length $g + 4$ using $tr(F_3^{g+4})$.

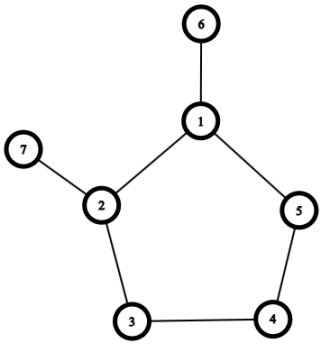


Figure 4.4: F_4

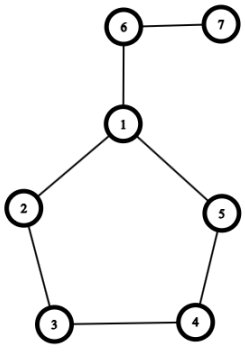


Figure 4.5: F_5

In order to find structures like F_4 and F_5 in a general graph, we face difficulties because their calculation requires the use of formulas $F_2(d_j - 2)$ and $F_2(d_j - 1)$ respectively. However, the adjacency matrix is not sufficient to provide information about the actual position of the j^{th} vertex within the graph.

Therefore, we are unable to calculate cycles of length $g + 4$ using this approach in general graphs. Hence, in the next chapter, we use this approach to find odd-length cycles in regular graphs.

4.3 Future Direction and Approach

As we have observed in Chapter 3, the algorithm has certain limitations due to the increasing time complexity as g increases. This is primarily because checking isomorphism among graphs becomes more time-consuming with the exponential growth in the number of structures. Therefore, a more efficient algorithm needs to be developed for handling isomorphism checks more quickly as g increases.

Furthermore, in Chapter 4, we were able to find the structures using the algorithm described in Chapter 3 and Section 3.3. However, formulating a closed-form expression for the number of such structures using a purely combinatorial approach proves to be challenging. Consequently, we either need to develop an algorithm capable of deriving such a formula or explore an alternative algebraic approach.

Bibliography

[1] Ian F. Blake and Shu Lin. On short cycle enumeration in biregular bipartite graphs. *IEEE Transactions on Information Theory*, IT-25(6):693–701, 1979.

[2] Ali Dehghan and Amir H. Banihashemi. On computing the multiplicity of cycles in bipartite graphs using the degree distribution and the spectrum of the graph. *IEEE Transactions on Signal Processing*, 69:1127–1142, 2021.

[3] Douglas B. West. *Introduction to Graph Theory*. Prentice Hall, 2001.