Ph.D. THESIS MATHEMATICS

Metric Dimension: Contributions and Algorithms for $BG_1(G)$ and $BG_2(G)$ Graphs

A thesis submitted to the

University of Calicut

for the award of the degree of

DOCTOR OF PHILOSOPHY
in
MATHEMATICS

under the Faculty of Science

by

SAMEERALI C.P

(Reg. No. 342/2019/Admn)

Under the supervision of

Dr. SAMEENA KALATHODI

PG and Research Department of Mathematics,
M.E.S Mampad College (Autonomous)

Kerala, India 676542.

APRIL 2025

DECLARATION

I hereby declare that the work presented in the thesis entitled "**Metric Dimension: Contributions and Algorithms for $BG_1(G)$ and $BG_2(G)$ Graphs**" is based on the original work done by me under the guidance of **Dr.Sameena Kalathodi, Assistant Professor, M.E.S Mampad College(Autonomous), Malappuram, Kerala** and has not been included in any other thesis submitted previously for the award of any degree. The contents of the thesis are undergone plagiarism check using **iThenticate** software at C.H.M.K. Library , University of Calicut, and the similarity index found within the permissible limit. I also declare that the thesis is free from AI generated contents. Further, all the corrections and suggestions recommended by the adjudicators have been carefully incorporated in the final version of the thesis.

Sameerali C.P

(Research Scholar)

Dr. Sameena Kalathodi

(Research Supervisor)

Dr. SAMEENA KALATHODI
Asst. Professor, HoD & Research Guide
P G & Research Department of Mathematics
M. E. S. Mampad College (Autonomous)
Mampad, Malappuram - 676 542

M.E.S Mampad College,

15-04-2025.

CERTIFICATE

This is to certify that the thesis titled **"Metric Dimension: Contributions and Algorithms for $BG_1(G)$ and $BG_2(G)$ Graphs"** submitted by **Mr. Sameerali C.P** to the University of Calicut in partial fulfillment of the requirements for the award of the Degree of Doctor of Philosophy in Mathematics is a record of original research work carried out by him under my supervision. The content of this thesis, in full or in parts, has not been submitted by any other candidate to any other University for the award of any degree or diploma. Further, all the corrections and suggestions recommended by the adjudicators have been carefully incorporated in the final version of the thesis.

Dr. Sameena Kalathodi

(Research Supervisor)

Dr. SAMEENA KALATHODI
Asst. Professor, HoD & Research Guide
P G & Research Department of Mathematics
M. E. S. Mampad College (Autonomous)
Mampad, Malappuram - 676 542

M.E.S Mampad College,

15-04-2025.

Certificate

This is to certify that all the corrections and suggestions recommended by the adjudicators have been carefully incorporated in the final version of the Thesis entitled "**Metric Dimension: Contributions and Algorithms for $BG_1(G)$ and $BG_2(G)$ Graphs**" submitted by **Mr. Sameerali C.P.** It has also been certified that the thesis and the soft copy are the same.

Dr. Sameena Kalathodi

Research Supervisor

Dr. SAMEENA KALATHODI
Asst. Professor, HoD & Research Guide
P G & Research Department of Mathematics
M. E. S. Mampad College (Autonomous)
Mampad, Malappuram - 676 542

Mampad

10.04.2025

ACKNOWLEDGEMENTS

I am deeply grateful to my research supervisor, *Dr. Sameena Kalathodi*, Research Guide and Head, Department of Mathematics, M.E.S Mampad College, for her invaluable guidance, unwavering support, and insightful suggestions throughout this research journey. Her expertise and encouragement have been instrumental in the completion of this thesis.

I extend my sincere thanks to the subject Expert, *Dr. Preethi Kuttippilakkal*, Associate Professor and Head, Department of Mathematics, University of Calicut, Kerala, for her expert advice and constructive feedback, which significantly shaped my research work. My gratitude also goes to the University Nominee, *Dr. Muhammed Ali N*, Associate Professor, Department of Mass Communication, University of Calicut, Kerala, for his valuable input and encouragement.

I would like to express my heartfelt appreciation to *Dr. Manzur Ali P.P*, Principal, M.E.S Mampad College, for his continuous support and for providing a conducive environment for my research.

I am profoundly thankful to my grandmother, *Mrs. Kayyumma K*, whose care and guidance supported me throughout my studies. Though she is no longer with us, her memory continues to inspire and encourage me. I am equally grateful to my father, *Mr. Mohammed C.P*, and my uncle, *Mr. Yahkoob C.P*, who, along with my mother, *Mrs. Ayisha A.P*, and my aunt, *Mrs. Subaida M.T*, supported me selflessly during my education. I also wish to acknowledge my beloved wife, *Mrs. Thasni K*, and my children, for their patience and understanding, especially during the times I had to dedicate myself to this work. Their sacrifices have not gone unnoticed, and

I am deeply grateful for their unwavering support.

I also want to thank every member of my family for their love, patience, and encouragement. Your steadfast support has been my strength.

I would like to express my heartfelt gratitude to my colleagues at Government Arts and Science College Mankada, for their support throughout this journey. Your encouragement and assistance have been invaluable.

Special thanks are due to *Mrs. Asha P*, Assistant Professor of Computer Science, Sri Achutha menon Govt. College, Thrissur and *Dr. Sadiquali A*, Associate Professor, M.E.A.E.C Perinthalmanna, for their valuable suggestions and guidance. I also fondly remember all my teachers, whose teachings and encouragement have greatly influenced my academic and professional journey.

I would like to thank my co-researchers at the M.E.S Mampad College, whose collaboration and discussions enriched my research experience.

Finally, I extend my gratitude to *Dr. Ranjini M.C*, RAC Member, Department of Mathematics, M.E.S Kalladi College, and *Dr. Shahida A.T*, RAC Member, Department of Mathematics, M.E.S Mampad College, for their constructive feedback and suggestions during my research work.

LIST OF FIGURES

LIST OF SYMBOLS

V	The vertex set for graph G
E	The edge set for graph G
$G = (V, E)$	A graph G with vertex set V and edge set E
$\|V\|$	Number of vertices in the graph G
$\|E\|$	Number of edges in the graph G
$\text{diam}(G)$	The diameter of the graph G
K_n	Complete graph with n vertices
C_n	Cycle with n vertices
P_n	Path graph with n vertices
$K_{1,n}$	Star graph with n leaves
$\Delta(G)$	Maximum vertex degree in the graph G
$\delta(G)$	Minimum vertex degree in the graph G
$BG_1(G)$	Boolean graph of the first kind
$BG_2(G)$	Boolean graph of the second kind
v'	Point vertex corresponding to the vertex v of G
e'	Line vertex corresponding to the edge e of G
$d(u, v)$	The distance between vertices u and v
$s(v)$	Status of the vertex v
$M(G)$	Median of the graph G
$G_1 \cong G_2$	G_1 is isomorphic to G_2
u''	The vertex in $BG_1^2(G)$ or $BG_2^2(G)$ corresponding to the vertex u in G
$\overline{G}$	The complement of the graph G
$\mathcal{O}(n)$	Big-O notation for algorithm complexity

LIST OF TABLES

Preface

Background and Motivation

The subject Graph theory is a branch of Mathematics that studies the structures which are used to model the association between objects. These geometric structures, called graphs, contain vertices (also called nodes) and edges (or lines) joining pairs of vertices. Over the years, the subject graph theory has expanded and been used in various fields like Computer Science, Chemistry, Biology, and Social Sciences, making it a good tool for finding solutions to complex problems, especially in networks.

P.J. Slater in 1975, introduced the idea of *Metric dimension* in the graph theory. The concept of metric dimension is the identification of the location of vertices in a graph concerning their distances to a set of selected vertices, called a *Resolving set*. The size of the smallest resolving set is called the metric dimension of the graph. The metric dimension has important roles in areas such as robotics, networks, and chemical graph theory, where it assists in locating and identifying all nodes in a network or system.

In this thesis, the structural properties of the expanded graphs $BG_1(G)$ and $BG_2(G)$ are studied, with a focus on their metric dimension. The study of metric dimensions in these expanded graphs opens new doors for innovation, particularly in the development of algorithms and real-world applications such as robotics and network systems.

We are living in a world of rapidly growing networks. The expansion of social platforms like WhatsApp and Facebook are very good examples of expanding networks. As the networks become more complex, it is crucial to analyze and manage their growth effectively. So, a study of growing networks is very suited for this age of technological advancements, where network structures play an increasingly central role in daily life. In this aspect, this study is highly relevant as it offers insights into the mathematical frameworks of two expanding networks.

Problem Statement

The metric dimension is a well-studied area in traditional graph theory, but its application to expanded graphs like $BG_1(G)$ and $BG_2(G)$ is relatively new. While Thusleem Furjana[4] has made some contributions by studying the metric dimension of $BG_2(G)$. The metric dimension of $BG_1(G)$ and the algorithmic aspects of determining the exact metric dimension, particularly for $BG_1(G)$ and $BG_2(G)$, remain unexplored. The main challenges lie in addressing this gap by developing efficient algorithms to compute the metric dimension of these graphs. This thesis aims to address the following research questions:

- What are the structural properties of $BG_1(G)$ and $BG_2(G)$, and how do these properties relate to the metric dimension of the graphs?
- Can bounds be established for the metric dimension of $BG_1(G)$ and $BG_2(G)$ for various base graphs, such as complete graphs, star graphs, path graphs, and cycle graphs?
- How can efficient algorithms be developed to compute the metric dimension of these expanded graphs?

Objectives of the Study

The primary objective of this thesis is to explore and analyze the metric dimension of $BG_1(G)$ and $BG_2(G)$ graphs, focusing on the following key areas:

1. To review the existing literature and basic results related to graph expansions and network analysis.

2. To study the fundamental properties of $BG_1(G)$ and $BG_2(G)$ graphs, including their composition and powers.

3. To investigate the metric dimensions of $BG_1(G)$ and $BG_2(G)$ graphs.

4. To develop and analyze algorithms for determining the metric dimensions and metric bases of $BG_1(G)$ and $BG_2(G)$.

5. To explore the algorithmic complexity of metric dimensions in the complements of $BG_1(G)$ and $BG_2(G)$ graphs.

6. To demonstrate practical applications of the metric dimensions and metric bases of $BG_1(G)$ and $BG_2(G)$ in real-world scenarios.

Thesis Outline

This thesis is organized into nine chapters, each of which addresses different aspects of the study:

- **Chapter 1: Preliminary Concepts and Essential Background**: Provides the foundational concepts and a review of relevant literature, setting the stage for the main contributions.

- **Chapter 2: Structural Properties of $BG_1(G)$ and $BG_2(G)$ Graphs**: Discusses the composition, powers, and structural properties of $BG_1(G)$ and $BG_2(G)$ graphs.

- **Chapter 3: Metric Dimension: a case study of $BG_1(G)$ Graph**, Presents the main theoretical results related to the metric dimension of these expanded graphs, including exact values for certain graph families.

- **Chapter 4: Metric Dimension of $BG_1\overline{(G)}$ and $BG_2\overline{(G)}$ Graphs:** This chapter Presents theoretical results related to the metric dimension of expanded graphs with complement of a given graph as the base graph.

- **Chapter 5: Algorithmic Perspectives on Metric Dimensions in $BG_1(G)$ and $BG_2(G)$ graphs**: Develops and explains algorithms for computing the metric dimension.

- **Chapter 6: Algorithmic Complexity of Metric Dimension in $BG_1\overline{(G)}$ and $BG_2\overline{(G)}$ Graphs** : Analyzes the complexity of the proposed algorithms and compares them with existing methods.

- **Chapter 7: Real-World Applications for Metric Dimension of $BG_1(G)$ and $BG_2(G)$ graphs** : Gives the applications of the metric dimension of $BG_1(G)$ and $BG_2(G)$ graphs in real-world scenarios, focusing on robotics and collaboration networks.

- **Chapter 8: Conclusion**: Summarizes the research findings and discusses the limitations and possible future extensions of the work.

- **Chapter 9: Recommendations**: This chapter Guides future research in this area.

UNIVERSITY OF CALICUT

CERTIFICATE ON PLAGIARISM CHECK

1.	Name of the Research Scholar	Sameerali C.P	
2.	Title of thesis / dissertation	Metric Dimension: Contributions and Algorithms for BG1(G) and BG2(G) Graphs	
3.	Name of the Supervisor	Dr. SAMEENA KALATHODI	
4.	Department/Institution	Reg No:342/2019/admn Research Scholar, MES Mampad College	
5.	Similar content (%) identified	Non Core **Introduction/ Theoretical overview/Review of literature/ Materials & Methods/ Methodology**	Core **Analysis/Result/Discussion / Summary/Conclusion/ Recommendations**
		9	**3**
	Acceptable maximum limit (%)	10	10
6.	Software used	iThenticate	
7.	Date of verification	Thu Jan 11 2024 00:00:00 GMT+0530 (India Standard Time)	

**Report on plagiarism check, specifying included/excluded items with % of similarity to be attached.*

Checked by (with name , designation & signature) Dr. Nasirudheen. T
Assistant Librarian
University of Calicut, Kerala.

Name and signature of the Researcher Sameerali cp

Name and signature of the Supervisor. Sameena Kalathodi

The Doctoral Committee* has verified the report on plagiarism check with the contents of the thesis, as summarized above and appropriate measures have been taken to ensure originality of the Research accomplished herein.

Name & Signature of the HoD/HoI (Chairperson of the Doctoral Committee)

Dr. MANZUR ALI PP
PRINCIPAL
PEN: 468239
M.E.S. Mampad College
(Autonomous)

**In case of languages like Malayalam, Tamil etc..on which no software is available for plagiarism check, a manual check shall be made by the Doctoral Committee, for which an additional certificate has to be attached.*

TABLE OF CONTENTS

Chapter 1

Preliminary Concepts and Essential Background

Introduction

In this chapter, basic definitions and concepts in the subject of graph theory are presented, with a focus on the metric dimension of graphs. The chapter begins by giving basic terms which are needed for understanding graph structures. The concept of metric dimension is then introduced, it acts as a parameter for determining vertices uniquely. The chapter proceeds to review existing research on $BG_1(G)$, which reveals its properties and relevance. Furthermore, it investigates recent studies on $BG_2(G)$ in the context of its metric dimension. This chapter is a foundation for the coming chapters.

1.1 Basic Definitions and graph Terminology

Definition 1.1.1. ***(Graph)***

A graph $G = (V, E)$ is a structure consisting of two sets V and E:

- *V is the set of vertices (or nodes), and*
- *E is the set of edges, where each edge joins at most two vertices.*

In a real-world scenario, the set V represents the objects(nodes) in the scenario, while a member in E represents a pair of nodes having a specific relationship between them. A graph can be visualized as a collection of dots connected by line segments. The number of vertices in a graph is called the *Size* of the graph and a graph with a finite number of vertices is called a *Finite graph.*

Definition 1.1.2. ***(Degree of a Vertex)***

The degree of a vertex v in a graph G is the number of edges incident to v. It is denoted as $deg(v)$ or $d(v)$.

Definition 1.1.3. ***(Isolated Vertex)***

A vertex v in a graph G is called an isolated vertex if it has no edges incident to it, i.e., $deg(v) = 0$.

Definition 1.1.4. ***(Cut-vertex)***

A vertex v in a graph G is called a cut-vertex if its removal increases the number of components of G.

Definition 1.1.5. ***(Path)***

A path in a graph G is a sequence of vertices and edges where neither vertices (except possibly the start and end vertices in the case of a closed path) nor edges are repeated. Each consecutive pair of vertices in the sequence is connected by an edge.

Definition 1.1.6. ***(Connected Graph)***

A graph G is said to be connected if there exists a path between any two vertices of G. In other words, for every pair of vertices v_i and v_j, there is a sequence of edges connecting them.

Definition 1.1.7. ***(Girth)***

The girth of a graph G is the length of the shortest cycle in G. If G has no cycles, its girth is defined as infinity.

Definition 1.1.8. ***(Median)***

In a graph G, the median of a set of vertices is a vertex that minimizes the sum of distances to all other vertices in the set.

Definition 1.1.9. ***(Status of a Vertex)***

The status of a vertex v in a graph G is the sum of the distances from v to all other vertices in G. It is denoted as $status(v)$.

Definition 1.1.10. ***(Planar Graph)***

A graph G is called a planar graph if it can be drawn on a plane without any of its edges crossing each other.

Definition 1.1.11. ***(Isomorphic Graphs)***

Two graphs G_1 and G_2 are said to be isomorphic if there exists a one-to-one correspondence between their vertex sets and their edge sets that preserves the adjacency relation.

Definition 1.1.12. ***(Simple Graph)***

A graph G is called a simple graph if it has no loops (edges from a vertex to itself) and no multiple edges between two vertices.

Definition 1.1.13. ***(Complete Graph)***

A graph G is called a complete graph if every two vertices are connected by an edge. A complete graph with n vertices is denoted as K_n.

Definition 1.1.14. ***(Star Graph)***

A graph G with $n+1$ vertices is called a Star graph, when there is a vertex with degree n and the remaining vertices are of degree one. A Star graph with $n+1$ vertices is denoted as $K_{1,n}$.

In addition to the definitions provided here, for definitions of basic graph-theoretic terms and related terminology, one can refer *Introduction to Graph Theory* [13] by Douglas B. West and *Distance in Graphs* [6] by F. Buckley and F. Harary, they serve as a comprehensive resource for foundational definitions in graph theory.

Theorem 1.1.15. *[13][Kuratowski's Theorem] A graph G is a planar graph if and only it is free from a subdivision of the complete graph K_5 or the complete bipartite graph $K_{3,3}$.*

1.2 Metric Dimension: Definitions and Basic Results

The metric dimension of a graph is a very relevant idea in graph theory. It helps to determine the position of any vertex based on its distances to a set of selected vertices. This concept finds many applications in various areas like network theory, robotics, and biology. In this section, the formal definition of the metric dimension is presented together with some major results from the literature, revealing its impacts on different types of graphs.

Definition 1.2.1. ***(Metric Code)***

Let G be a graph with V as its vertex set and let $B = \{v_1, v_2, \ldots, v_k\}$ be a subset of V. The metric code of a vertex v in V with respect to the subset B, denoted by $C_B(v)$, is defined as:

$$C_B(v) = (d(v, v_1), d(v, v_2), \ldots, d(v, v_k))$$

where $d(v, v_i)$ denotes the distance between the vertices v and v_i in the graph G.

Definition 1.2.2. ***(Resolving Set)***

A subset S is termed a resolving set of the graph G if it uniquely determines each

vertex in V based on their metric codes. Specifically, for any two distinct vertices u and v in V, their metric codes must be different, that is,

$C_S(u) \neq C_S(v)$ *whenever* u *and* $v \in V$

Definition 1.2.3. ***(Metric Dimension)***

The metric dimension of the graph G is defined as the cardinality of the smallest resolving set. This value quantifies the smallest number of vertices required in S to ensure that S serves as a resolving set for G.

Example 1.2.4. *The figure 1.1 illustrates the metric codes of the vertices with respect to the set $S = \{A, B\}$. These codes confirm that S is a resolving set, as each vertex is uniquely identified by its code. Moreover, S is the smallest possible resolving set, since no singleton subset of S can distinguish all vertices through unique codes.*

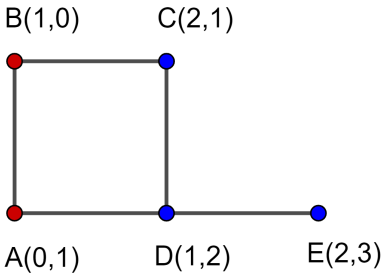


Figure 1.1: Metric code and Resolving set

Theorem 1.2.5. *[10] If G is a connected graph of order $n \geq 2$ and diameter d, then*

$$f(n, d) \leq \beta(G) \leq n - d.$$

where, $f(n, d)$ is the least positive integer k for which $k + d^k \geq n$.

Theorem 1.2.6. *[10] Let G be a connected graph of order n. Then, $\beta(G) = 1$ if and only if $G = P_n$.*

Theorem 1.2.7. *[10] A connected graph G of order $n \geq 2$ has metric dimension $n - 1$ if and only if $G = K_n$.*

Name	**Graph**	$\lvert V(G)\rvert$	$\beta(G)$	
Path	P_n	$n \geq 1$	1	
Cycle	C_n	$n \geq 3$	2	
Complete	K_n	$n \geq 2$	$n-1$	[12]
Bicomplete	$K_{r,s}$	$r+s \geq 3$	$n-2$	
Wheel	$W_{1,r}$	$r+1 = 4, 7$	3	
Hypercube	$Q_r = [K_2]^r$	2^r	$r\,(r \leq 4)$	

Table 1.1: Precise Metric Dimensions in Selected Graph Families

Theorem 1.2.8. *[17] Let the graph G is connected and simple. Then,* $\log_3(\Delta+1) \leq \beta(G) \leq n - diam(G)$.

1.3 $BG_1(G)$ Graphs: A Literature Overview

T.N. Janakiraman, M. Bhanumathi, and S. Muthammai introduced and studied the graphs $BG_1(G)$ and $BG_2(G)$. They have identified several properties of these expanded graph classes. This section presents the results regarding the girth, connectivity, eccentricity, and subgraph of these expanded structures. The theorems and propositions included here, state the conditions under which $BG_1(G)$ and $BG_2(G)$ have the properties such as regularity, Hamiltonicity, and Eulerian cycles. These findings are very helpful in the study of the metric dimensions of these expanded graphs.

Definition 1.3.1. *[3] (*$BG_1(G)$***-Boolean Graph of the First Kind)***
Let G be a simple graph with vertex set V and edge set E. The Boolean graph of the first kind, denoted by $BG_1(G)$, is defined as follows:

The vertex set of $BG_1(G)$ is the union of the vertex set V and the edge set E of the original graph G, that is, $V \cup E$. Two vertices in $BG_1(G)$ are adjacent if and only if they either correspond to adjacent vertices in the graph G or correspond to a vertex and an edge of G such that the edge is not incident with that vertex in G. The vertices corresponding to the vertices of G are called point vertices and those corresponding to edges of G are called line vertices of $BG_1(G)$.

Example 1.3.2. *The figure 1.2 gives the graph P_3 and $BG_1(P_3)$.*

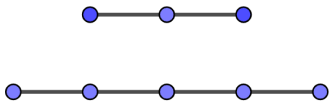


Figure 1.2: P_3 and $BG_1(P_3)$

Proposition 1.3.3. *[3] The girth of $BG_1(G)$ is 3 if and only if the girth of G is three or G has at least two non-adjacent edges.*

Proposition 1.3.4. *[3] The girth of $BG_1(G)$ is four if and only if $G = K_{1,n}$ or $G = K_{1,n} \cup mK_1$, $n > 2$.*

Proposition 1.3.5. *[3] The girth of $BG_1(G)$ is six if and only if $G = K_{1,2} \cup K_1$.*

Proposition 1.3.6. *[3] Let G be a connected non-trivial (p, q) graph with $p > 3$. Then $BG_1(G)$ is connected.*
$BG_1(G)$ is disconnected if and only if $G = K_2$, $2K_2$, nK_1, or $K_2 \cup nK_1$.

Proposition 1.3.7. *[3] Let G be a connected (p, q) graph such that $BG_1(G)$ is connected. If $BG_1(G)$ has a cut point, then it is a point vertex only.*

Proposition 1.3.8. *[3] Let G and $BG_1(G)$ be connected graphs. Then $BG_1(G)$ has a cut vertex if and only if $G = K_3$ or $G = K_{1,2}$.*

Lemma 1.3.9. *[3] Let G be a connected graph, and let $u \in V(G)$ and $e \in E(G)$. Then in $BG_1(G)$, e and u are connected by at least $p - 2$ edge-disjoint paths.*

Theorem 1.3.10. *[3] $BG_1(G)$ is free from induced C_5 if and only if G is isomorphic to any one of the following: K_2, K_3, $K_{1,n}$, P_4, or $K_{1,n} \cup mK_1$.*

Theorem 1.3.11. *[3] $BG_1(G)$ has C_e as an induced subgraph if and only if G has $K_{1,2} \cup K_1$ as an induced subgraph or G has three mutually non-adjacent vertices with degree at least one.*

Theorem 1.3.12. *[3] $BG_1(G)$ contains C_n, $n > 7$ as an induced subgraph if and only if G contains C_n, $n > 7$ as an induced subgraph.*

Proposition 1.3.13. *[3] If $BG_1(G)$ is connected, the eccentricity of a point vertex is 2, 3, or 4.*

Lemma 1.3.14. *[3] If G is a non-trivial (p, q) graph with more than four vertices and has no pendant vertices, then the eccentricity of every line vertex is two.*

Theorem 1.3.15. *[3] If G is connected with more than four vertices and has no pendant vertex, then $BG_1(G)$ is self-centered with diameter two.*

Proposition 1.3.16. *[3] If G is connected with more than three vertices and has pendant vertices, then the eccentricity of a pendant vertex u' and pendant edge e' in $BG_1(G)$ is three.*

Theorem 1.3.17. *[3] In $BG_1(G)$, the eccentricity of every point vertex is two and that of a line vertex is three if and only if $G = K_3$, K_4, or C_4.*

Theorem 1.3.18. *[3]*

1. *The eccentricity of a line vertex in $BG_1(G)$ is four if and only if $G = K_{1,2}$.*
2. *The eccentricity of a point vertex in $BG_1(G)$ is four if and only if $G = 2K_2 \cup nK_1$.*

Theorem 1.3.19. *[3] $BG_1(G)$ is self-centered with diameter three if and only if $G = K_{1,2} \cup K_2$, $K_{1,2} \cup mK_1$, or nK_2, $n > 3$.*

1.4 Existing Results on $BG_2(G)$ Graphs and its Metric Dimension

M. Thusleem Furjana and M. Bhanumathi have made contributions to the study of the metric dimension of $BG_2(G)$ graphs. Their work gives the bounds for the

metric dimension of $BG_2(G)$ graphs. The propositions and theorems presented in this section are about connected and disconnected graphs, as well as paths and cycle graphs. These findings are strong enough to provide insight into the properties and metric dimensions of $BG_2(G)$.

Definition 1.4.1. *[3]($BG_2(G)$-**Boolean Graph of the Second Kind**)*

Let G be a simple connected graph with vertex set V and edge set E. The Boolean graph of the second kind, denoted by $BG_2(G)$, is defined as follows:

The vertex set of $BG_2(G)$ is given by the union $V \cup E$. Two vertices in $BG_2(G)$ are adjacent if and only if they satisfy one of the following conditions:

1. *They correspond to adjacent vertices in the original graph G.*
2. *They represent an edge incident to a vertex in G.*
3. *They represent two non-adjacent edges in G.*

Example 1.4.2. *The figure 1.3 shows P_4 and $BG_2(P_4)$*

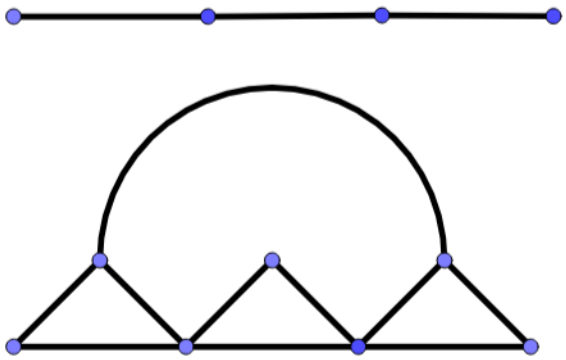


Figure 1.3: P_4 and $BG_2(P_4)$

Proposition 1.4.3. *[3] (1) Every vertex of $BG_2(G)$ lies in a triangle if G has no isolated vertices. (2) If $v_i \in V(G)$, and $deg(v_i) = d_i$ in G, then v_i lies on d_i triangles with distinct edges in $BG_2(G)$. (3) $BG_2(G)$ has at least q triangles and the girth of $BG_2(G)$ is three.*

Proposition 1.4.4. *[3] (1) $BG_2(G)$ has an isolated vertex if and only if G has an isolated vertex. (2) $BG_2(G)$ is regular if and only if G is regular and $deg(u) = \frac{q+3}{4}$, $p = \frac{4q}{q+3}$. (3) $BG_2(G)$ is connected if and only if G has no isolated vertices.*

Proposition 1.4.5. *[3] If G is a self-centered graph with radius r, then every vertex of $BG_2(G)$ lies on cycles of length 3, 4, 5, ..., $4r$ in $BG_2(G)$.*

Theorem 1.4.6. *[3] Let G be a graph without isolated vertices. Then $BG_2(G)$ is free from P_4 if and only if $G = K_{1,n}$.*

Lemma 1.4.7. *[3] $BG_2(G)$ has C_4 as an induced subgraph if and only if G has C_4, P_4, $K_{1,2} \cup K_2$, $K_3 \cup K_2$, $K_3 \cup K_{1,2}$, $2K_3$ or $2K_{1,2}$, $K_4 - e$, $K_{1,3} + e$ or K_4 as induced subgraphs.*

Proposition 1.4.8. *[3] Eccentricity of every line vertex is two in $BG_2(G)$ if $G \neq K_2$.*

Proposition 1.4.9. *[3] Let G be a graph without isolated vertices. If $BG_2(G)$ is Hamiltonian, then each vertex of G is incident to at most two pendant edges.*

Theorem 1.4.10. *[3] Let G be a regular graph. Then $BG_2(G)$ is Eulerian if and only if q is odd.*

Proposition 1.4.11. *[4] If G is a connected graph with $p \geq 4$, then $\beta(BG_2(G)) \leq p - 1$.*

Proposition 1.4.12. *[4] If G is a connected graph, then $2 \leq \beta(BG_2(G)) \leq p$.*

Proposition 1.4.13. *[4] Let G be a disconnected graph. Let $W \subseteq E(G)$. If G has K_2 as a component, then W is not a minimum resolving set of $BG_2(G)$.*

Theorem 1.4.14. *[4] Let G be any connected graph, then $\beta(BG_2(G)) = p$ if and only if $G = P_2$.*

Theorem 1.4.15. *[4] If $G = P_n$ $(n \geq 6)$, then $\beta(BG_2(P_n))$ is given by:*

$$\beta(BG_2(P_n)) = \begin{cases} 2m-1 & \text{when } n = 3m, \\ 2m & \text{when } n = 3m+1, \\ 2m+1 & \text{when } n = 3m+2. \end{cases} \quad m > 1.$$

Theorem 1.4.16. *[4] If $G = BG_2(C_n)$ $(n \geq 6)$, then $\beta(BG_2(C_n))$ is given by:*

$$\beta(BG_2(C_n)) = \begin{cases} 2m-1 & \text{when } n = 3m, \\ 2m & \text{when } n = 3m+1, \\ 2m+1 & \text{when } n = 3m+2. \end{cases} \quad m > 1.$$

Convention. *Unless otherwise specified, the symbol u' denotes the vertex in the Boolean graph of the graph G corresponding to the vertex u of the graph G. Similarly, the symbol e' represents the vertex in the Boolean graph corresponding to the edge e of the graph G.*

Chapter 2

Structural Properties of $BG_1(G)$ and $BG_2(G)$ Graphs

Introduction

This chapter focuses on the basic properties of the graphs $BG_1(G)$ and $BG_2(G)$. Through strict analysis, the chapter establishes fundamental results concerning the powers of $BG_1(G)$ and the composition $BG_1(BG_2(G))$, which will help to understand these complex structures. The results established in this chapter played a vital role in setting the content of the subsequent chapters. The chapter mainly discusses the connectivity, girth, and distances between vertices in the graph expansions. The knowledge of expansion properties of growing graph structures is essential to use them effectively. The study of expansions also offers valuable information in designing algorithms and protocols that can help in controlling and making beneficial use of these evolving graph structures.

2.1 Characteristics of the Graph $BG_1(G)$

Theorem 2.1.1. *Given a graph G with diam$(G) > 4$, the graph $BG_1(G)$ is non-planar.*

Proof. Kuratowski's theorem [20] gives a characterization of planar graphs; it states "a graph G is planar if and only if it does not contain K_5 or $K_{3,3}$, or a subdivision of K_5 or $K_{3,3}$ as a subgraph."
In the case where the diameter of graph G is 5, the graph $BG_1(G)$ contains a subgraph isomorphic to P_6. Let $v_1, v_2, \ldots, v_6$ represent the vertices and $e_i = v_i v_{i+1}$ denote the edges of P_6.

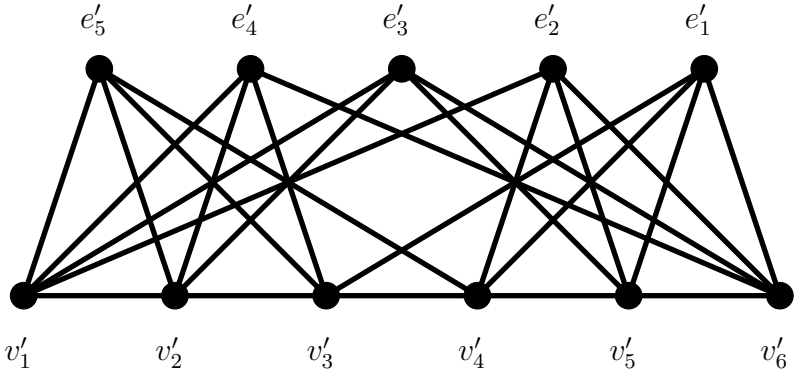


Figure 2.1: The graph $BG_1(P_6)$

In $BG_1(P_6)$, edges e'_5 and e'_4 are adjacent to each of the vertices v'_1, v'_2, and v'_3. Edge e'_3 is adjacent to v'_1 and v'_2 but not v'_3. There exists a path $e'_3 v'_5 v'_4 v'_3$ in $BG_1(P_6)$. By appending this path to the subgraph formed by the aforementioned vertices and edges in $BG_1(P_6)$, we obtain a subdivision of $K_{3,3}$. Therefore, according to Kuratowski's theorem [20], $BG_1(G)$ is not planar.

In the case where the diameter of G is greater than 5, $BG_1(G)$ contains a subgraph isomorphic to P_7. Using the same vertex and edge notation as in the previous case, each of the vertices e'_6, e'_5, and e'_4 is adjacent to v'_1, v'_2, and v'_3. This implies that $K_{3,3}$ is a subgraph of $BG_1(G)$. Therefore, by Kuratowski's theorem

[20], $BG_1(G)$ is not planar.

□

Theorem 2.1.2. *If G is a connected graph with more than 4 vertices, then the distance in $BG_1(G)$, denoted as $d(u', v')$, will be as:*

$$d(u', v') = \begin{cases} 1 & \text{if } d(u, v) = 1 \text{ or } u' \text{ is a point vertex and } v' = e' \\ & \quad \text{is a line vertex, where edge } e \text{ does not meet } u, \\ 3 & \text{if } u \text{ is a pendant vertex and } v' = e' \\ & \quad \text{is a line vertex, where edge } e \text{ meets } u, \\ 2 & \text{otherwise.} \end{cases}$$

Proof. If u and v are adjacent vertices of G, then they are adjacent in $BG_1(G)$ as well; therefore, $d(u', v') = 1$.

If u is a vertex of G and is not adjacent to the edge e, then u' and $e' = v'$ are adjacent in $BG_1(G)$, hence $d(u', v') = 1$.

If u is a pendant vertex and $e = uw$ is the edge adjacent to u, then e' is adjacent to neither u' nor w'. Since G is a connected graph with more than 4 vertices, $deg(w) \geq 2$. Let z be a vertex of G adjacent to w other than u. Then $e'z'w'u'$ is a shortest path from u' to $e' = v'$, i.e., $d(u', v') = 3$.

Now, if u and v are two non-adjacent vertices in a connected graph G with $d(u, v) = 2$, then by definition of $BG_1(G)$, $d(u', v') = 2$. If $d(u, v) > 2$, then it is possible to find an edge e in G not incident with u or v, and then $u'e'v'$ is a path in $BG_1(G)$, therefore $d(u', v') = 2$.

Assume u is a non-pendant vertex of G and the edge $e = uw$ meets u, then u' is non-adjacent to e'. Take a vertex z of G adjacent to u but not to w, then $e'z'u'$ is a path in $BG_1(G)$, hence $d(u', v') = 2$.

Finally, if u' and v' are line vertices in $BG_1(G)$. If they do not share a common vertex, then by assumption there is a fifth vertex z in G not adjacent to any of

these edges. So $u'z'v'$ is a path in $BG_1(G)$, so $d(u', v') = 2$. Now if u' and v' share a common vertex, there is a fourth vertex in G not an endpoint of these edges. Let it be z, then $u'z'v'$ is a path in $BG_1(G)$, so $d(u', v') = 2$.

□

Theorem 2.1.3. *Let G be a connected graph with n vertices and e edges ($n > 4$). Then, for a vertex v' in $BG_1(G)$, its status, $s(v')$ will be as:*

$$s(v') = \begin{cases} n + 2e & \text{if } v' \text{ is a line vertex corresponding to a non-pendent edge,} \\ n + 2e + 1 & \text{if } v' \text{ is a line vertex corresponding to a pendant edge,} \\ 2n + e - 2 & \text{if } v' \text{ is corresponding to a non-pendant vertex,} \\ 2n + e - 1 & \text{if } v' \text{ is associated to a pendant vertex.} \end{cases}$$

Proof. The distances between the vertices of $BG_1(G)$ considered throughout this proof are derived from Theorem 2.1.2. If v' is a line vertex corresponding to a non-pendant edge of G, then, in $BG_1(G)$, v' is adjacent to every point vertices except two. The distance from v' to the two remaining vertices is 2. Additionally, the distance from v' to each of the $e - 1$ point vertices is 2. Therefore,

$$s(v') = n - 2 + 2 + 2 + (e - 1) \times 2 = n + 2e.$$

If v' is a line vertex corresponding to a pendant edge of G, then, in $BG_1(G)$, v' is adjacent to every other point vertex except two. The distance from v' to one of the remaining vertices is 2, and the distance to the other remaining vertex is 3. Additionally, the distance from v' to each of the $e - 1$ point vertices is 2. Consequently,

$$s(v') = n - 2 + 2 + 3 + (e - 1) \times 2 = n + 2e + 1.$$

If v' is a point vertex associated with a non-pendant vertex of G, then, in $BG_1(G)$,

the distance from v' to $d(v)$, the degree of the vertex v in G, point vertices is 1 and the remaining $n - d(v) - 1$ vertices are at a distance of 2 from v'. Additionally, v' is adjacent to $e - d(v)$ line vertices, and it is at a distance of 2 from the remaining $d(v)$ line vertices. Therefore,

$$s(v') = d(v) \times 1 + (n - d(v) - 1) \times 2 + d(v) \times 2 + (e - d(v)) \times 1 = 2n + e - 2.$$

Finally, if v' is a point vertex associated with a pendant vertex of G, then the distance from v' to one of the point vertices is 1, and the remaining $n - 2$ point vertices are at a distance of 2 from v'. Additionally, v' is adjacent to $e - 1$ line vertices, and it is at a distance of 3 from the remaining line vertex. Thus,

$$s(v') = 1 + (n - 2) \times 2 + (e - 1) + 3 = 2n + e - 1.$$

□

Corollary 2.1.4. *Let G be a connected graph with n vertices and e edges ($n > 4$). Then, the median, $M(BG_1(G))$ =$\{$v': v' is a point vertex of $BG_1(G)$ and v' is not corresponding to a pendent vertex of G $\}$.*

Proof. Theorem 2.1.3 lists the different possible values of status in $BG_1(G)$. The first and second, and similarly, the third and fourth values for $s(v)$ suggested by Theorem 2.1.3 differ by 1. Given that G is connected, so $e \geq n - 1$. Now, let's consider

$$\begin{aligned}(n + 2e) - (2n + e - 2) &= e - n + 2 \\ &\geq n - 1 - n + 2 \\ &= 1\end{aligned}$$

That is, $2n + e - 2$ is the minimum value for $s(v)$, which is the status of a point vertex that does not correspond to a pendant vertex of G. Then by definition,

$M(BG_1(G))$ ={v': v' is a point vertex of $BG_1(G)$ and v is not corresponding to a pendent vertex of G }.

□

2.2 Some Features of the Graph $BG_2(G)$

Theorem 2.2.1. *[3] Let G be a graph with n vertices and e edges then, $BG_2(G)$ has $\frac{e^2+7e-\sum d_i^2}{2}$ edges.*

Proof. The edges of graph G are retained in $BG_2(G)$, directly contributing e edges to $BG_2(G)$. Additionally, each edge in G corresponds to a line vertex in $BG_2(G)$. This line vertex connects to the two point vertices in $BG_2(G)$ via two additional edges, contributing another $2e$ edges to $BG_2(G)$. There are e line vertices in the graph $BG_2(G)$. Let q' be a line vertex of $BG_2(G)$ corresponding to the edge $q = v_iv_j$ of G. Let the vertex v_i is adjacent to d_i vertices, and v_j is adjacent to d_j vertices in G; the edge v_iv_j is common to both classes. Therefore, the total number of non-adjacent edges to q is $e-(d_i+d_j-1)$. Taking into account that the edges v_iv_j and v_jv_i are identical, the total number of edges between line vertices is equal to $\frac{1}{2}\sum(e-(d_i+d_j-1))$, where the $\sum$ is taken over all line vertices of $BG_2(G)$. Hence the total number of edges in $BG_2(G)$ is,

$$\begin{aligned} e+2e+\frac{1}{2}\sum(e-(d_i+d_j-1)) &= 3e+\frac{1}{2}\left(e+e+\cdots+e \text{ terms}\right) \\ &\quad -\sum(d_i+d_j)+(1+1+\cdots+1 \text{ terms}) \\ &= \frac{1}{2}\left(e^2+6e-\sum(d_i+d_j)+e\right) \end{aligned} \tag{1}$$

when the sum is taken over all end vertices of edges v_iv_j,

$$\sum(d_i+d_j) = (d_1+d_1+\cdots+d_1 \text{ terms})+(d_2+d_2+\cdots+d_2 \text{ terms})+\ldots$$

$$= \sum d_i^2 \tag{2}$$

Using (2) in (1), the number of edges in $BG_2(G)$,

$$= \frac{1}{2}(e^2 + 7e - \sum d_i^2)$$

□

Remark 2.2.2. *M.Bhanumathi has made an observation in her thesis [3] that the number of edges of $BG_2(G)$ will be $\frac{e^2 + 7e - \sum d_i^2}{2}$, but she did not provide a proof. Here, a detailed proof of this observation is presented.*

Theorem 2.2.3. *Let G be a graph with with diam$(G) > 5$, the graph $BG_2(G)$ is non-planar.*

Proof. Assume that the diameter of graph G is 6. This implies that graph G contains a subgraph isomorphic to P_7. Let the vertices be $v_1, v_2, \ldots, v_7$ and the edges be $e_i = v_i v_{i+1}$.

In graph $BG_2(G)$, the vertex e'_1 is adjacent to e'_4 and e'_6, and there is a path $e'_1 v'_2 e'_2$.

Similarly, vertex e'_3 is adjacent to e'_6, and there are two paths $e'_3 v'_4 e'_4$ and $e'_3 v'_3 e'_2$ within $BG_2(G)$.

Additionally, vertex e'_5 is adjacent to e'_2, and the paths $e'_5 v'_5 e'_4$ and $e'_5 v'_6 e'_6$ present in $BG_2(G)$.

When all the mentioned edges, vertices, and paths are combined, they form a subdivision of $K_{3,3}$. Thus, according to Kuratowski's theorem [20], $BG_2(G)$ is non-planar.

When the diameter of G is greater than 6, the graph contains a subgraph isomorphic to P_7, which also includes a subgraph P_6. Therefore, the arguments

provided for $\text{diam}(G) = 6$ apply in this case as well, leading to the conclusion that $BG_2(G)$ is non-planar. □

Theorem 2.2.4. *If G is a connected graph with more than 2 vertices, then the distance in $BG_2(G)$, denoted as $d(u', v')$, will be as follows:*

$$d(u', v') = \begin{cases} 1 & \text{if } d(u, v) = 1 \text{ or } u' \text{ is a point vertex and } v' = e' \\ & \text{is a line vertex, where edge } e \text{ is adjacent to } u \\ & \text{or when, } u \text{ and } v \text{ are non-adjacent edges in } G \\ 3 & \text{if } u \text{ and } v \text{ are vertices of } G \text{ with } d(u, v) \geq 3 \\ 2 & \text{otherwise} \end{cases}$$

Proof. If $d(u, v) = 1$, then $d(u', v') = 1$.

If v is an edge meeting with the vertex u, then by definition of $BG_2(G)$, $d(u', v') = 1$.

When u and v are two non-adjacent edges of G, then by definition of $BG_2(G)$, $d(u', v') = 1$.

Now, let u and v be two vertices of G with $d(u, v) \geq 3$. Let $ux \dots yv$ be a shortest path from u to v in G. Take the edges $ux = e_1$ and $yv = e_2$ of G, then $u'e_1'e_2'v'$ is a shortest path from u' to v'. Therefore, $d(u', v') = 3$.

Assume u and v are two edges sharing a common vertex w, then $u'w'v'$ is a shortest path in $BG_2(G)$. Hence, $d(u', v') = 2$.

Next, assume that u is a vertex and v is a non-adjacent edge of u in G. Let $v = xy$. If x or y is adjacent to u, then $u'x'v'$ or $u'y'v'$ is a shortest path in $BG_2(G)$. Then, $d(u', v') = 2$.

If neither x nor y is adjacent to u, take an edge z with u as one end vertex, then $u'z'v'$ is a shortest path in $BG_2(G)$.Consequently, $d(u', v') = 2$.

□

2.3 Powers of the Graph $BG_1(G)$

Let G be a graph with n vertices and e edges. Then $BG_1^2(G)$ is the graph $BG_1(BG_1(G))$, it has $n(e+1)$ vertices and $e(n-1)(n+e-1)$ edges.The adjacency in $BG_1^2(G)$ emerges from the definition of adjacency in $BG_1(G)$. In similar fashion, one can define the higher powers of $BG_1(G)$, denoted as $BG_1^k(G)$ for any positive integer k. If the sequence, $n_1, e_1, n_2, e_2, \ldots$, represents the sequence of vertices and edges in $BG_1^k(G)$, $k = 1, 2, 3 \ldots$ then,

$$n_k = n_{k-1} + e_{k-1} \quad \text{and} \quad e_k = e_{k-1}(n_{k-1} - 1)$$

Theorem 2.3.1. *Let G be a simple connected graph with more than 2 vertices. Then, the girth of $BG_1^2(G)$ is 3.*

Proof. Given that the number of vertices, $n > 2$, there will be at least two edges, denote them as e_1 and e_2, in G. Now, consider two cases:

Case 1: (The edges e_1 and e_2 share a common vertex)

Let the edges be $e_1 = uv$ and $e_2 = vz$ in G. Then, in $BG_1(G)$, there exists a path $e_2'u'v'z'e_1'$. Let h be the edge $h = z'e_1'$ in $BG_1(G)$. Then, $h'e_2''u''h'$ forms a cycle in $BG_1^2(G)$. Consequently, girth of $BG_1^2(G)$ is 3.

Case 2: (The edges e_1 and e_2 does not share a common vertex)

Let the edges be $e_1 = uv$ and $e_2 = xy$ in G. Then, $e_2'u'v'e_2'$ is cycle in $BG_1(G)$ and this cycle is preserved in $BG_1^2(G)$. Therefore, the girth of $BG_1^2(G)$ is 3. □

Corollary 2.3.2. *Let G be a simple connected graph with more than two vertices, and let $k \geq 2$. Then, the girth of $BG_1^k(G)$ is 3.*

Proof. One of the cycles mentioned in the proof of Theorem 2.3.1, depending on the cases discussed, will persist in the higher powers of $BG_1(G)$. Therefore, the girth of $BG_1^k(G)$ is 3.

□

Theorem 2.3.3. *Let G be a simple connected graph with more than 2 vertices then $BG_1^2(G)$ has no cut vertex.*

Proof. The vertices of $BG_1^2(G)$ can be classified into three categories:

1. The vertices corresponding to vertices of G.
2. The vertices corresponding to edges of G.
3. The vertices corresponding to an edge of $BG_1(G)$.

Case 1: (The vertex of $BG_1^2(G)$ corresponding to a vertex of G)
Let u'' be a vertex of the specified type within $BG_1^2(G)$. According to the assumption, $BG_1(G)$ has at least 5 vertices, with 3 of them belonging to this type and it has at least 4 edges. Now, we choose another vertex v'' of the same type as u'', and adjacent to u'' in $BG_1^2(G)$ then,u' and v' are adjacent in $BG_1(G)$. Let h be an edge where neither u' nor v' is an endpoint in $BG_1(G)$. Such an edge must exist because $BG_1(G)$ has P_5 as a subgraph with 3 of its interior vertices belonging to the class of u'. Now, consider the cycle formed by the vertices $h'u''v''h'$ in $BG_1^2(G)$. In this cycle, u'' is a part of the cycle or u'' is not a cut vertex.
Case 2: (The vertex of $BG_1^2(G)$ corresponding to an edge of G)
Let e'' be a vertex of this type in $BG_1^2(G)$.choose a vertex v' of $BG_1(G)$ such that v' is an end vertex of e' in $BG_1(G)$. Let f be any edge of $BG_1(G)$ such that neither e' nor v' is an end vertex of f. Then, $e''v''f'e''$ is a cycle in $BG_1^2(G)$ and hence, e'' is not a cut-vertex.
Case 3: (The vertex corresponding to an edge of $BG_1(G)$)
Let f be an edge of $BG_1(G)$) and $e = u'v'$ be an edge not adjacent to f in $BG_1(G)$. The choice of such an edge, e, is possible as P_5 is a subgraph of $BG_1(G)$. Then, $f'u''v''f'$ is a cycle in $BG_1^2(G)$ containing f' and hence, f' is not a cut vertex.
Thus, $BG_1^2(G)$ has no cut vertex.

□

Corollary 2.3.4. *Let G be a simple connected graph with more than 2 vertices then $BG_1^k(G)$, $k \geq 2$, has no cut vertex.*

Proof. The Theorem 2.3.3 gives the proof for $k = 2$. When $k > 2$, $BG_1^k(G)$ $=BG_1^2(BG_1^{k-2}(G))$. Here, $BG_1^{k-2}(G))$ is a graph that fulfills our assumption of the Theorem 2.3.3. Therefore, by Theorem 2.3.3 $BG_1^k(G)$ has no cut vertex. □

Theorem 2.3.5. *Let G be a graph with n vertices and e edges and v be a vertex of $BG_1^2(G)$. Then,*

$$deg_{BG_1^2(G)}(v) = \begin{cases} n+e-2, & \text{If } v \text{ is a line vertex of } BG_1^2(G) \\ e(n-1), & \text{if } v \text{ is a vertex corresponding to a line vertex of } BG_1(G) \\ e(n-1), & \text{if } v \text{ is a point vertex corresponding to a vertex of } G. \end{cases}$$

Proof. Let v be a line vertex of $BG_1^2(G)$. $BG_1(G)$ is a graph with $n+e$ vertices, so in $BG_1(BG_1(G))$, every line vertex is adjacent to $n+e-2$ point vertices and not adjacent to any other vertices. Therefore, $deg_G(v) = n+e-2$.
Let v be a point vertex of $BG_1^2(G)$ corresponding to a line vertex of $BG_1(G)$. Then, $deg_{BG_1(G)}(v) = n-2$. There are $e+e(n-2)$ edges in $BG_1(G)$. Hence, There are $e+e(n-2) = e(n-1)$ line vertices in $BG_1^2(G)$ and all but except $n-2$ of them are adjacent to v. Consequently, $deg_{BG_1^2(G)}(v) = (n-2)+(e(n-1)-(n-2)) = e(n-1)$.
Finally, let v be a point vertex of $BG_1^2(G)$, corresponding to a vertex of G. $deg_G(v)$ edges are incidenting the vertex v in G.In $BG_1(G)$, $e - deg_G(v)$ edges will additionally incident with v. Out of the $e(n-1)$ line vertices of $BG_1^2(G)$, $e(n-1) - (deg_G(v) + e - deg_G(v))$, will incident with v. Therefore,
$deg_{BG_1^2(G)}(v) = deg_G(v)+(e-deg_G(v))+(e(n-1)-(deg_G(v)+e-deg_G(v)) = e(n-1)$.

□

Theorem 2.3.6. *Let G be any graph with at least two edges. Then, $BG_1^2(G)$ is connected.*

Proof. Consider a graph G comprising two disjoint edges, $e = uv$ and $f = ab$. Then, $BG_1(G)$ contains two cycles, $e'a'b'e'$ and $f'u'v'f'$. Let h be the edge $h = a'e'$ in $BG_1(G)$. Then,$b''h'u''$ is a path connecting the two cycles in $BG_1^2(G)$. Thus, the subgraph of $BG_1^2(G)$, induced by the point vertices and line vertices of the above cycles, is connected. Upon examination of any two other vertices x and y in $BG_1^2(G)$, the following cases arise.

Case 1: (x and y are point vertices of $BG_1^2(G)$)

$xh'y$ is a path in $BG_1^2(G)$ from x to y.

Case 2: (x is a point vertex and y is a line vertex of $BG_1^2(G)$)

If u' is not an end vertex of the edge corresponding to the line vertex y, then $xh'u''y$ is a path in $BG_1^2(G)$ from x to y.

If u' is an end vertex of the edge corresponding to the line vertex y, then $xh'v''y$ is a path in $BG_1^2(G)$ from x to y.

Case 3: (x and y are line vertices of $BG_1^2(G)$)

Here, $xu''y$ is a path from x to y.

Next assume that the two edges of G are adjacent, and they are $e = uv$ and $f = vw$. Then, $BG_1(G)$ contains a path $f'u'v'w'e'$. The subgraph of $BG_1^2(G)$ induced by the point and line vertices, associated with the above path in $BG_1(G)$, is connected. Examining any other pair of vertices s and t within $BG_1^2(G)$, there are the following distinct possibilities.

Case 1: (s and t are point vertices of $BG_1^2(G)$)

Let g be the edge $g = u'v'$ in $BG_1(G)$. Then, $sg't$ is a path from s to t.

Case 2: (s is a point vertex and t is a line vertex of $BG_1^2(G)$)

If v' is not an end vertex of the edge associated to the line vertex t, then $sg'w''t$ is a path in $BG_1^2(G)$ from s to t and the same path exists when v is an end vertex of that edge.

Case 3: (x and y are line vertices of $BG_1^2(G)$)

Here, $su''t$ is a path from s to t.

Finally, Let s be any vertex from the induced subgraphs mentioned above and t be any other vertex of $BG_1^2(G)$. If t is a point vertex, then t is adjacent to at least one of the line vertex of the mentioned connected induced subgraph containing s. Therefore, there is a path from s to t. When t is the line vertex, t is adjacent to at least one of the point vertex of the connected subgraph containing s.
Consequently, in all cases, a path can be traced between any pair of vertices in$BG_1^2(G)$), affirming the connectivity of the graph.
Similarly, these arguments hold true even when the graph G has more than two edges. □

Corollary 2.3.7. *If G is a graph with at least one edge, then for all $k > 3$, $BG_1^k(G)$ is connected.*

Proof. Consider the graph G with at least one edge. Then, $K_2 \cup K_1$ is a subgraph of $BG_1(G)$. Consequently, $BG_1^2(G)$ contains the subgraph $2K_2$, implying $BG_1^2(G)$ has at least two vertices. By Theorem 2.3.6, $BG_1^2(BG_1^2(G)) = BG_1^4(G)$, is connected. Since $BG_1(G)$ inherits connectedness from the connectedness of G, we conclude that $BG_1^k(G)$ is connected for all $k > 3$.

□

Corollary 2.3.8. *if $k > 3$, $BG_1^k(G)$ is disconnected if and only if $G \cong nK_1$*

Proof. Assume $k > 3$ and $BG_1^k(G)$ is disconnected. If possible assume that $G \ncong nK_1$. Then, G has at least one edge. Consequently, Corollary 2.3.7 implies that $BG_1^k(G)$ is connected, which is a contradiction. Therefore, $G \cong nK_1$.
Next, let $G \cong nK_1$. $BG_1^k(nK_1) \cong nK_1$, which is disconnected for $k > 1$.

□

Corollary 2.3.9. *If G is a graph with at least one edge, then for all $k > 3$, $BG_1^k(G)$ contains P_8.*

Proof. Let G be a graph with at least one edge. Then, the argument in the proof for Corollary 2.3.7 shows that $2K_2$ is a subgraph of $BG_1^2(G)$. Consequently, $2C_3$ is a subgraph of $BG_1^3(G)$. Let the cycles be $u_1u_2u_3$ and $v_1v_2v_3$. Let e_1, e_2, e_3 and f_1, f_2, f_3 respectively denote the edges in the cycles. Then, $f_1'u_3'u_1'u_2'e_3'v_1'v_3'v_2'f_3'$ is a path in $BG_1^4(G)$. A copy of this path persists within $BG_1^k(G)$ for $k > 4$.

□

Theorem 2.3.10. *The graph $BG_1^2(G)$ is disconnected if and only if G is isomorphic to one of the graphs K_2, $K_2 \cup K_1$ or nK_1.*

Proof. If $BG_1^2(G)$ is disconnected, Theorem 2.3.6 suggests that G can have at most one edge, leading to three potential graphs: K_2, $K_2 \cup nK_1$, or nK_1. $BG_1^2(K_2) \cong 2K_2$ i.e., $BG_1^2(K_2)$ is disconnected.

$BG_1^2(K_2 \cup K_1) \cong 2C_3$ i.e., $BG_1^2(K_2 \cup K_1)$ is disconnected.

When $n > 1$, $BG_1(K_2 \cup nK_1) \cong K_2 \cup K_{1,n}$. Thus, $BG_1^2(K_2 \cup nK_1) = BG_1(K_2 \cup K_{1,n})$, which is a connected graph.

$BG_1^2(nK_1) \cong nK_1$ i.e., $BG_1^2(nK_1)$ is disconnected.

Consequently, The choices for G are K_2, $K_2 \cup K_1$ or nK_1.

The proof of the converse part of the theorem follows from the preceding discussion.

□

Corollary 2.3.11. *The graph $BG_1^3(G)$ is disconnected if and only if G is isomorphic to K_2 or nK_1.*

Proof. Let $BG_1^3(G)$ be disconnected, *i.e.*, $BG_1^2(BG_1(G))$ is disconnected. then Theorem 2.3.6 implies that $BG_1(G)$ can have at most one edge, leading to three possibilities for $BG_1(G)$: K_2, $K_2 \cup nK_1$, or nK_1.

No graph G generates K_2 or $K_2 \cup nK_1$, where $n > 1$, as its $BG_1(G)$. Thus, the possibilities for $BG_1(G)$ are narrowed down to $K_2 \cup K_1$ and nK_1. In other words, the choices for G are either K_2 or nK_1.

conversely, assume that G is isomorphic to K_2 or nK_1.

$BG_1^3(K_2) \cong 2C_3$, which is disconnected.

$BG_1^3(nK_1) \cong nK_1$, which is also disconnected.

□

Theorem 2.3.12. *Let G be a graph containing at least two edges and α , β denote two vertices of $BG_1^2(G)$. Then, $d(\alpha, \beta)$ is either 1 or 2.*

Proof. Let u_1'' and u_2'' denote the point vertices of $BG_1^2(G)$ corresponding to the point vertices u_1 and u_2 of G respectively. Similarly, let e_1' and e_2' be the point vertices of $BG_1^2(G)$ corresponding to the line vertices e_1 and e_2 of $BG_1(G)$). Now, let f_1 and f_2 represent two line vertices of $BG_1^2(G)$. Given various possibilities, the cases for $d(\alpha, \beta)$ can be described as follows:

Case 1: ($\alpha = u_1''$ and $\beta = u_2''$)

if u_1 and u_2 are adjacent in G then, u_1'' and u_2'' are adjacent in $BG_1^2(G)$ and hence, $d(\alpha, \beta)$=1. If u_1 and u_2 are non-adjacent in G, then either 2C3 or P5 is a subgraph of $BG_1(G)$. Consequently, it's possible to select an edge g in $BG_1(G)$ that is non-adjacent to both u_1' and u_2'. As a result, $u_1''g'u_2''$ forms a path in $BG_1^2(G)$. Therefore, $d(\alpha, \beta)$=2.

Case2 : ($\alpha = u_1''$ and $\beta = e_1'$)

If u_1' and e_1 are adjacent in $BG_1(G)$ then, u_1'' and e_1' are adjacent in $BG_1^2(G)$ and hence, $d(\alpha, \beta)$=1. otherwise, the choice of an edge, nonadjacent to both u_1' and e_1' in $BG_1(G)$, is possible because of the reason mentioned in Case 1. Let that edge be g. Then, $u_1''g'e_1'$ is a path in $BG_1^2(G)$. Therefore, $d(\alpha, \beta)$=2.

Case3 : ($\alpha = e_1'$ and $\beta = e_2'$)

Here, e_1 and e_2 are line vertices of $BG_1(G)$. Therefore, they are neither adjacent in $BG_1(G)$ nor in $BG_1^2(G)$. There are at least 5 vertices in $BG_1(G)$, so it is able to select a vertex v, of $BG_1(G)$, which is neither an end vertex of e_1 nor that of e_2. Consequently, $e_1'v'e_2'$ is a path in $BG_1^2(G)$. Thus, $d(\alpha, \beta)$=2.

Case4 : ($\alpha = u_1''$ and $\beta = f_1$)

If u_1'' and f_1 are adjacent in $BG_1^2(G)$ then, $d(\alpha, \beta)$=1. Otherwise, f_1 corresponds to some edge $u_1'v_1$ in $BG_1(G)$.Now, $deg_{BG_1(G)}(u_1') = e$, the number of edges of G, here it is minimum 2, so it is able to choose a vertex v_2 adjacent to u_1' other than v_1. Consequently, $f_1v_2'u_1''$ is a path in $BG_1^2(G)$. Thus, $d(\alpha, \beta)$=2.

Case5 : ($\alpha = f_1$ and $\beta = f_2$)

f_1 and f_2 are not adjacent in $BG_1^2(G)$ then, $d(\alpha, \beta) \neq 1$.

Let f_1 and f_2 are corresponding to the edges $u_1'u_2'$ and $v_1'v_2'$ respectively. Since the minimum number of vertices in $BG_1(G)$ is 5, it is able to choose a different vertex w_1 from $BG_1(G)$ other than u_1', u_2', v_1' and u_2'. Therefore, $f_1w_1'f_2$ is a path in $BG_1^2(G)$. Hence, $d(\alpha, \beta)$=2.

Case6 : ($\alpha = e_1'$ and $\beta = f_1$)

If e_1' and f_1 are adjacent in $BG_1^2(G)$ then, $d(\alpha, \beta)$=1. Otherwise, f corresponds to an edge of $BG_1(G)$ with e_1 as one of its end vertex. Let that edge be e_1v_1. $deg_{BG_1(G)}(e_1) = n + e - 2$, where n is the number of vertices in G and e is the number of edges of G.

Here, $deg_{BG_1(G)}(e_1)$ is minimum 3.

so it is able to choose a vertex v_2 of $BG_1(G)$, adjacent to e_1 and different from v_1. Hence, $fv_2'e_1'$ is a path in $BG_1^2(G)$ and $d(\alpha, \beta)$=2.

□

Corollary 2.3.13. *Let G be a graph with at least one edge and three vertices. If $k \geq 3$ and α, β denote two vertices of $BG_1^k(G)$, then $d(\alpha, \beta)$ is either 1 or 2.*

Proof. Here, $2K_2$ is a subgraph of $BG_1(G)$. *i.e.*, $BG_1(G)$ has at least two edges. Therefore, by Theorem 2.3.12, in $BG_1^3(G) = BG_1^2(BG_1(G))$, the distance $d(\alpha, \beta)$ is either 1 or 2. These two edges of $2K_2$ will remain in every power $BG_1^k(G)$. Consequently, the fact $BG_1^k(G) = BG_1^2(BG_1^{k-2}(G))$ and Theorem 2.3.12, together implies the result. □

2.4 The Graph $BG_1(BG_2(G))$

Let G be a graph with n vertices and e edges. Then $BG_1(BG_2(G))$ is the graph with $n + e$ vertices and the adjacency is defined in accordance with the definitions of the boolean graphs of first and second kind.

Theorem 2.4.1. *Suppose G is a simple graph with at least one edge. Then, the girth of $BG_1(BG_2(G))$ is 3.*

Proof. Let $e = uv$ be an edge of G, then $e'u'v'e'$ is a cycle in $BG_2(G)$. This cycle is preserved in $BG_1(BG_2(G))$. Given that the graph is simple and thus contains no cycles of length 2, it follows that the girth of $BG_1(BG_2(G))$ is 3.

□

Theorem 2.4.2. *If G is a graph with at least one edge, then $BG_1(BG_2(G))$ is connected.*

Proof. Given a graph G with at least one edge, consider its associated graph $BG_2(G)$, which contains at least three edges that form a cycle C. Let the cycle be $v_1v_2v_3$, with the corresponding edges e_1, e_2, and e_3. Let's examine the existence of a path between two vertices x' and y' in $BG_1(BG_2(G))$, considering the following scenarios:

Case 1 (G is connected):

In this case, $BG_2(G)$ is also connected.

- **Subcase 1** (x' and y' are both point vertices of $BG_1(BG_2(G))$):

Since $BG_2(G)$ is connected, there is a path from x to y in $BG_2(G)$, and therefore a corresponding path exist from x' to y' in $BG_1(BG_2(G))$.

- **Subcase 2** (x' and y' are both line vertices of $BG_1(BG_2(G))$):

If x' and y' are associated with edges of cycle C in $BG_2(G)$, they connect directly through C', forming a path in $BG_1(BG_2(G))$. If neither x' nor y' are associated with the edges of C, then at least one vertex v_i from the cycle C is not an endpoint

of the edges associated with x' and y'. In that case, a path from x' to y' can be established via v'_i. If x' is associated with the edge of C and y' is not, then both x' and y' are connected to C' in $BG_1(BG_2(G)))$, thus there is a path from x' to y'.

- **Sub-case 3** (x' is a line vertex and y' is a point vertex of $BG_1(BG_2(G)))$:

If y is not an endpoint of the edge associated with x', then $x'y'$ is a path in $BG_1(BG_2(G))$. Now, if y is an endpoint, both y and x are part of some cycle in $BG_2(G)$. Choose a vertex z from the cycle that is not an endpoint of x and is adjacent to y. This choice is possible because every vertex of $BG_2(G)$ is a part of some cycle. this choice ensures the formation of a path $x'z'y'$ in $BG_1(BG_2(G))$.

Case 2 (G is disconnected): Given the scenario where G is a graph that is disconnected, the following situation is presented: If x' and y' are vertices in $BG_1(BG_2(G))$ and are associated with the same component of G, then the same arguments from **Case 1** can be applied to prove that a path exists from x' to y'. Therefore, we will focus on the situation where x' and y' are associated with different components of G.

- **Subcase 1** (both x' and y' are point vertices of $BG_1(BG_2(G)))$:

In this situation, at least one edge e_j in the cycle C of $BG_2(G)$ will exist such that neither x nor y is an endpoint of e_j. Consequently, the path $x'e'_jy'$ can be established in $BG_1(BG_2(G))$.

- **Subcase 2** (both x' and y' are line vertices of $BG_1(BG_2(G)))$:

Here, x and y are two edges in $BG_2(G)$. Since they relate to different components of G, their end vertices differ. Additionally, x and y belong to two different cycles of $BG_2(G)$. Choose a vertex z of $BG_2(G)$ from the cycle of x such that z is not an endpoint of x. This allows the creation of a path $x'z'y'$ from x' to y' in $BG_1(BG_2(G))$.

- **Subcase 3** (x' is a point vertex and y' is a line vertex of $BG_1(BG_2(G)))$:

In this scenario, x represents a vertex, and y represents an edge in $BG_2(G)$, with x not being an endpoint of y. This allows for the creation of the path $x'y'$ in

$BG_1(BG_2(G))$.

In every case discussed, a path can be found from x' to y', demonstrating that $BG_1(BG_2(G))$ is connected.

□

Theorem 2.4.3. *The graph $BG_1(BG_2(G))$ is disconnected if and only if $G \cong nK_1$, $n > 1$.*

Proof. If $BG_1(BG_2(G))$ is disconnected, then the theorem 2.4.2 implies that G has no edges and hence $G \cong nK_1$. Conversely, if $G \cong nK_1$, then $BG_2(G) \cong nK_1$, leading to $BG_1(BG_2(G)) \cong nK_1$. Thus, when $n > 1$, $BG_1(BG_2(G))$ is disconnected.

□

Theorem 2.4.4. *If G is a simple graph with n vertices, e edges, and d_i denotes the degree of vertex v_i, then $BG_1(BG_2(G))$ is a graph with $\frac{e^2+2n+9e-\sum d_i^2}{2}$ vertices and $\frac{e^2+7e-\sum d_i^2}{2}(n+e-1)$ edges.*

Proof. Given a graph G with n vertices and e edges, the graph $BG_2(G)$ has $n+e$ vertices. According to Theorem 2.2.1 , $BG_2(G)$ has $\frac{e^2+7e-\sum d_i^2}{2}$ edges. Therefore, graph $BG_1(BG_2(G))$ has

$$n+e+\frac{e^2+7e-\sum d_i^2}{2} = \frac{e^2+2n+9e-\sum d_i^2}{2}$$

vertices. If G is a graph with p vertices and q edges then $BG_1(G)$ has p+q vertices and $q(p-1)$ edges. Consequently, $BG_1(BG_2(G))$ has $\frac{e^2+7e-\sum d_i^2}{2}(n+e-1)$ edges.

□

Theorem 2.4.5. *If G is a non-trivial simple connected graph, then the graph $BG_1(BG_2(G))$ has a cut vertex if and only if $G \cong P_2$.*

Proof. Let $G = P_2$, $BG_2(P_2) = C_3$ and $BG_1(BG_2(P_2)) = BG_1(C_3)$.

$BG_1(C_3)$ has 3 pendant vertices and hence have cut vertices.

To prove the next part, let G be a graph with more than two vertices and at least two edges, let $v_1, v_2, \ldots, v_n$ be the vertices of G, and $e_1, e_2, \ldots, e_m$ be the edges of G. Let $e_1 = v_iv_j$ and $e_2 = v_pv_q$. The vertices of $BG_1(BG_2(G))$ can be classified into three categories:

1. Line vertices of $BG_1(BG_2(G))$
2. Vertices corresponding to line vertices of $BG_2(G)$
3. Vertices corresponding to point vertices of $BG_2(G)$

There are at least two cycles $v_i'v_j'e_1'v_i'$ and $v_p'v_q'e_2'v_p'$ in $BG_2(G)$, indicating that there are at least six edges in $BG_2(G)$.

Let v be a vertex from the first category (line vertices of $BG_1(BG_2(G))$). Suppose e_k is the corresponding edge in $BG_2(G)$. We can find an edge $e = u_iu_j$ among the six mentioned earlier such that it does not share a common vertex with e_k, where u_i and u_j be any of v_x', v_y' and e_z'. Then, $u_i'u_j'vu_i'$ forms a cycle in $BG_1(BG_2(G))$, showing that v is not a cut vertex.

If v belongs to the second category (vertices corresponding to line vertices of $BG_2(G)$), each line vertex in $BG_2(G)$ is part of a cycle in $BG_2(G)$, and the same cycle is present in $BG_1(BG_2(G))$.

Finally, if v belongs to the third category (vertices corresponding to point vertices of $BG_2(G)$), the point vertex of $BG_2(G)$ also forms part of a cycle in $BG_2(G)$, and the same cycle exists in $BG_1(BG_2(G))$. Therefore, v is not a cut vertex. Thus, if $BG_1(BG_2(G))$ has a cut vertex, then G either has 2 or fewer vertices, or it has 1 or 0 edges. Since G is simple connected, it has only one choice, 2 vertices, and one edge or $G \cong P_2$.

□

Chapter 3

Metric Dimension: A Case Study of $BG_1(G)$ Graph

Introduction

This chapter presents a comprehensive collection of theorems and corollaries on the metric dimension of Boolean graphs, $BG_1(G)$. The metric dimension $\beta(BG_1(G))$ for various types of graphs is explored in detail. Specifically, results are developed for finding $\beta(BG_1(G))$ for complete graphs, star graphs, cycle graphs, and paths. The graphs with $\beta(BG_1(G)) = 1$ have been identified, and upper and lower bounds for $\beta(BG_1(G))$ are established in terms of both the number of vertices and edges in the graph G. The results contained in this chapter refine existing bounds and offer practical insights for applying metric dimension techniques in areas such as network theory and graph-based algorithms. These results lead the way for developing algorithmic approaches to computing the metric dimension of Boolean graphs in subsequent chapters.

3.1 Bounds and Results on the Metric Dimension of $BG_1(G)$

Theorem 3.1.1. *For any connected undirected non trivial Graph* $\beta(BG_1(G))$ *=1 if and only if* $G = P_3$.

Proof. Let $\beta(BG_1(G)) = 1$. If possible, assume that G is not a path. Then, by the definition of $BG_1(G)$, it is also not a path. Hence, by Theorem 1.2.6, $\beta(BG_1(G)) > 1$, which is a contradiction, and so G is a path. Assume $G = P_n$ and consider the following cases for n.

Case 1: $n > 3$,

Let e be an edge meeting a pendant vertex. Then there will be at least two vertices v_1 and v_2 in G that are not meeting the edge e. Since G is connected, there is at least one path $v_1u_1u_2\cdots u_kv_2$ from v_1 to v_2 in G. Then $v_1'u_1'u_2'\cdots u_k'v_2'e'v_1'$ is a cycle in $BG_1(G)$. Hence, by Theorem 1.2.6, $\beta(BG_1(G)) > 1$, which contradicts $\beta(BG_1(G)) = 1$.

Case 2: $n = 3$,

The Figure 3.1 shows that $BG_1(P_3)$ is P_5. Therefore, by Theorem 1.2.6, $\beta(BG_1(G)) = \beta(P_5) = 1$.

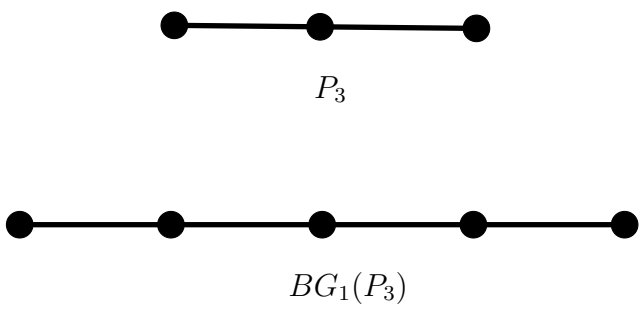


Figure 3.1: The graphs P_3 and $BG_1(P_3)$

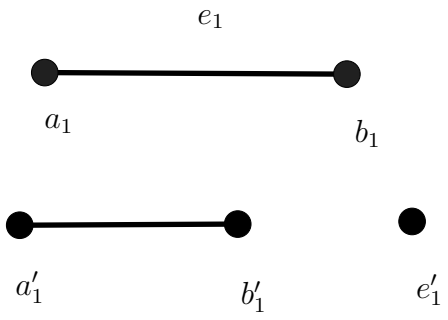


Figure 3.2: The graphs P_2 and $BG_1(P_2)$

Case 3: $n = 2$,

$BG_1(P_2)$ is disconnected. Therefore, $\beta(BG_1(P_2)) \neq 1$.

Thus, if $\beta(BG_1(G)) = 1$, then $G = P_3$.

Conversely, assume $G = P_3$. The Figure 3.1 shows that $BG_1(P_3)$ is P_5. Therefore, by Theorem 1.2.6, $\beta(BG_1(P_3)) = 1$.

□

Theorem 3.1.2. *Let G be a graph with more than two vertices. Then, the set of all point vertices in the Boolean Graph $BG_1(G)$ forms a resolving set.*

Proof. Let $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_p\}$ and $E = \{e_1, e_2, \ldots, e_q\}$. Let V' and E' be the sets of point vertices and line vertices of $BG_1(G)$. Let $V' = \{v'_1, v'_2, \ldots, v'_p\}$ and $E' = \{e'_1, e'_2, \ldots, e'_q\}$.

$C_{V'}(v'_i) \neq C_{V'}(v'_j)$, since the i^{th} component of $C_{V'}(v'_i)$ is zero, and that of $C_{V'}(v'_j)$ is the distance from v_i to v_j, which is at least one. Thus, the codes of point vertices from V' are all different.

Let e' be a line vertex of $BG_1(G)$. Assume e joins v_m and v_n in G. Then, in $BG_1(G)$, e' is neither adjacent to v'_m nor to v'_n, but is adjacent to all the remaining point vertices. Then $C_{V'}(e')$ will have $p - 2$ of its components as 1's, and the

remaining two components are greater than 1, i.e.,

$$C_{V'}(e') = (1, 1, \ldots, a, 1, 1, \ldots, b, 1, \ldots, 1),$$

where a and b occur respectively in the m^{th} and n^{th} positions.

If possible, assume e'_i and e'_j are two line vertices of $BG_1(G)$ with $C_{V'}(e'_i) = C_{V'}(e'_j)$. This means the components that are not 1 occur in the same positions. That is, e_i and e_j join the same vertices in G, which is a contradiction to the fact that G is simple. Thus, $C_{V'}(e'_i) \neq C_{V'}(e'_j)$. Moreover, $C_{V'}(v'_i) \neq C_{V'}(e'_j)$, since the i^{th} component of $C_{V'}(v'_i) = 0$, and none of the components of $C_{V'}(e'_j)$ is zero.

Hence, all the vertices of $BG_1(G)$ have different codes, so V' is a resolving set.

□

Corollary 3.1.3. *If $G = (V, E)$ and $BG_1(G)$ is connected, then*

$$\beta(BG_1(G)) \leq |V|.$$

Proof. Theorem 3.1.2 says V' is a Resolving set. Then by definition, the Metric dimension will not exceed the cardinality of V. □

Theorem 3.1.4. *If G=(V,E) and $BG_1(G)$ is connected, then*
$Log_3(\mid E \mid +1) \leq \beta(BG_1(G) \leq \mid E \mid + \mid V \mid - diam(BG_1(G))$.

Proof. The degree of a point vertex in $BG_1(G)$ is $\mid E \mid$ and that of a line vertex is $\mid V \mid -2$, which implies that $\Delta = \mid E \mid$. Theorem 1.2.8 gives

$$\log_3(\Delta + 1) \leq \beta(G) \leq n - \text{diam}(G).$$

$BG_1(G)$ has $\mid E \mid + \mid V \mid$ vertices, and thus

$$\log_3(\mid E \mid +1) \leq \beta(BG_1(G)) \leq \mid E \mid + \mid V \mid - \text{diam}(BG_1(G)).$$

□

Theorem 3.1.5. *Let G=(V,E) and $BG_1(G)$ are connected graphs. Then,* $Log_3(\mid E \mid +1) \leq \beta(BG_1(G)) \leq \mid V \mid$.

Proof. Theorem 3.1.4 gives

$$Log_3(\mid E \mid +1) \leq \beta(BG_1(G)).$$

Corollary 3.1.3 gives

$$\beta(BG_1(G)) \leq \mid V \mid .$$

Combining these results,

$$Log_3(\mid E \mid +1) \leq \beta(BG_1(G)) \leq \mid V \mid .$$

□

Theorem 3.1.6. *Let $G = (V, E)$ and $BG_1(G)$ be connected. Then*

$$\beta(G) < 3^{\beta(BG_1(G))}.$$

Proof. For any connected graph G, corollary 3.1.3 gives,

$$\beta(G) \leq \mid V \mid -1. \tag{1}$$

Since G is connected,

$$\mid V \mid -1 \leq \mid E \mid . \tag{2}$$

Now, since Log_3 is an increasing function, (1) and (2) gives,

$$Log_3(\beta(G)) \leq Log_3(\mid V \mid -1) \leq Log_3(\mid E \mid).$$

$$< Log_3(\mid E \mid) + 1.$$

$$\leq \beta(BG_1(G)).$$

$$\text{i.e., } Log_3(\beta(G)) < \beta(BG_1(G)).$$

Thus,

$$\beta(G) < 3^{\beta(BG_1(G))}.$$

□

3.2 Metric Dimension of $BG_1(G)$ for Complete and Star Graphs

Theorem 3.2.1. *If G is complete and $\mid V \mid > 2$, then any $\mid V \mid -1$ point vertices form a resolving set of $BG_1(G)$.*

Proof. Let $S = \{v'_1, v'_2, \ldots, v'_{i-1}, v'_{i+1}, \ldots, v'_n\}$ be a set of $\mid V \mid -1$ point vertices. v'_i and all line vertices are outside S. The code $C_S(v'_i) = (1, 1, \ldots, 1)$. In $C_S(v'_k), k \neq i$, the component corresponding to v'_k is zero, and all other components are non-zero. Therefore, the codes of every vertex in S are different.

Let e' be a line vertex. There are two cases:

Case 1: e is incident with v'_i.

Let $e = v_iv_j$. In this case, the distance from e' to a point vertex other than v'_i and v'_j is 1, and the distance from e' to v'_i and v'_j is 2. Hence, $C_S(e') = (1, 1, \ldots, 1, 2, 1, \ldots, 1)$. The position of 2 varies for each edge of this type, so the codes of such vertices are different.

Case 2: e is not incident with v_i.

Let $e = v_kv_h$. Here, the distances from e' to v'_k and v'_h are 2, and the distances from e' to other point vertices are 1. Hence, $C_S(e') = (1, 1, \ldots, 2, \ldots, 2, 1, \ldots)$. The 2's

occur in positions corresponding to v'_k and v'_h. The position of the 2's will vary uniquely from line vertex to line vertex, so the metric codes of all line vertices of this type are different.

Thus, the metric codes of all point and line vertices are different, which means S is a resolving set.

□

Theorem 3.2.2. *If G is complete and $\mid V \mid > 2$, then $\beta(BG_1(G)) = \mid V \mid -1$.*

Proof. Let's consider two possibilities for $\mid V \mid$.

Case 1: ($\mid V \mid = 3$)

Theorem 3.2.1 implies that $\beta(BG_1(G)) \leq 2$. Here, $BG_1(G)$ is not a path. Therefore, by theorem 1.2.6 $\beta(BG_1(G)) > 1$. Thus, $\beta(BG_1(G)) = 2 = \mid V \mid -1$.

Case 2: ($\mid V \mid > 3$)

Using Theorem 3.2.1, $\beta(BG_1(G)) \leq \mid V \mid -1$. If possible, assume that S is a resolving set of $BG_1(G)$ with fewer than $\mid V \mid -1$ vertices. The following sub-cases are to be discussed.

Subcase 1: (S contains point vertices only)

Here, at least two point vertices lie outside S, say v'_i and v'_j.

$C_S(v'_i) = (1, 1, \ldots, 1)$ and $C_S(v'_j) = (1, 1, \ldots, 1)$, which contradicts the fact that S is a resolving set. Hence, there must be at least one line vertex in S.

Subcase 2: (S contains line vertices only)

Choose two line vertices e'_i and e'_j outside S. Then,

$C_S(e'_i) = (2, 2, \ldots, 2)$ and $C_S(e'_j) = (2, 2, \ldots, 2)$, which is not possible. Hence, there must be at least one point vertex in S.

Subcase 3: (S contains at least one point vertex and one line vertex)

In this case, there must be at least 3 point vertices outside S. Consider the $\binom{3}{2}$ line vertices associated with these point vertices. If at least two of these line vertices lie outside S, the metric codes of those line vertices will be the same and of the

form $(1, 1, \ldots, 1, 2, \ldots, 2)$, where the 1's correspond to point vertices in S and 2's correspond to line vertices of S. This is a contradiction, so two or more line vertices among the $\binom{3}{2}$ must lie in S. Then, at least four point vertices lie outside S. Now, consider the $\binom{4}{2}$ line vertices corresponding to these point vertices. If any two of them lie outside S, their codes will be the same, which is not possible, so there must be at least 7 point vertices outside S. Repeating the same arguments for a finite number of times, the number of line vertices in S will be greater than or equal to the cardinality of S. This is a contradiction, as S cannot include line vertices alone.

Therefore, there is no resolving set for $BG_1(G)$ with fewer than $\mid V \mid -1$ elements. That is, $\beta(BG_1(G)) = \mid V \mid -1$.

□

Corollary 3.2.3. *Let G be a complete graph and $\mid V \mid > 3$. Then, $\beta(BG_1(G)) = \mid V \mid -1 = \beta(G)$.*

Proof. G is a complete graph implies $\beta(G) = \mid V \mid -1$, by Theorem 1.2.7. Now the result follows from the Theorem 3.2.2. □

Theorem 3.2.4. *Let G be a star graph with $\mid V \mid > 3$. Then, the set of all point vertices, except any one of the leaves, forms a resolving set of $BG_1(G)$.*

Proof. Let $V = \{v_1, v_2, \ldots, v_n\}$ and v_1 be the center vertex of G. Let $S = \{v'_1, v'_2, \ldots, v'_{i-1}, v'_{i+1}, \ldots, v'_n\}$. Here, $C_S(v'_i) = (1, 2, 2, \ldots, 2)$, where the first position corresponds to v'_1 and all other components are 2. Let $e = v_1v_k$, with $k \neq i$. Then, $C_S(e') = (2, 1, \ldots, 2, \ldots, 1)$, where the positions of the 2's correspond to v'_1 and v'_k, while the other components are 1. As k varies, the position of the second 2 in $C_S(e')$ changes uniquely, distinguishing each line vertex associated with different v_k. Thus, the metric code of each line vertex is distinct, ensuring that S forms a resolving set.

□

Corollary 3.2.5. *If G is a star Graph and $\mid V \mid > 3$, then $\beta(BG_1(G)) \leq \mid V \mid -1$.*

Theorem 3.2.6. *If G is a star Graph and $\mid V \mid > 3$, then $\beta(BG_1(G)) = \mid V \mid -1$.*

Proof. From Corollary 3.2.5, $\beta(BG_1(G)) \leq \mid V \mid -1$. If possible, assume that $BG_1(G)$ has a resolving set S with fewer elements than $\mid V \mid -1$. Without loss of generality, assume that S has $\mid V \mid -2$ elements. Let e_i' be the line vertex corresponding to the edge $e_i = v_1v_i$, where v_1 is the center vertex of G. Now consider the following $\mid V \mid$ pairs of vertices of $BG_1(G)$:

$$(v_1', v_1'), (v_2', e_2'), (v_3', e_3'), \ldots, (v_n', e_n'),$$

where, in the first pair, both components are equal. Eliminate the pairs if at least one component is in S. Then, there remain at least two pairs, and we have the following cases:

Case 1: v_1' is in S

In this case, there must be two pairs, (v_i', e_i') and (v_j', e_j'), whose components are not in S. Also,

$$C_S(e_i') = (2, 1, 1, \ldots, 2, \ldots, 2) = C_S(e_j')$$

where the 2's correspond to v_1' and other line vertices in S, and the 1's correspond to point vertices in S. This is not possible.

Case 2: v_1' is not in S

Here, v_1' and the components of at least one pair (v_i', e_i') are not in S. Then

$$C_S(v_1') = (1, 1, \ldots, 2, 2, \ldots, 2) = C_S(e_i'),$$

where the 1's correspond to point vertices in S, and the 2's correspond to line vertices in S. This is a contradiction.

Hence, S is not a resolving set, and therefore, $\beta(BG_1(G)) = \mid V \mid -1$.

□

Corollary 3.2.7. *Let G_1 be a complete Graph and G_2 is a Star Graph with an equal number of vertices. If $\mid V \mid > 3$, then $\beta(BG_1(G_1)) = \mid V \mid -1 = \beta(BG_1(G_2)$.*

Proof. Result follows from Theorem 3.2.2 and Theorem 3.2.6 . □

Corollary 3.2.8. *If G is a Star Graph with $\mid V \mid > 3$, then $\beta(BG_1(G))$ exceeds $\beta(G)$ by one.*

Proof. Given, G is a Star Graph. Then,
$\beta(G) = \mid V \mid -2$ (set of $\mid V \mid -2$ leaves is a metric base).
From Theorem 3.2.6 , $\beta(BG_1(G)) = \mid V \mid -1$.

□

3.3 Metric Dimension of $BG_1(G)$ for Cycles and Paths

Theorem 3.3.1. *Let G be a simple connected graph with $\mid V \mid < 4$, then $\beta(BG_1(G)) = 2$ if and only if $G = C_3$.*

Proof. Let $G = C_3$. The fact that $\beta(G) = 1$ if and only if $G = P_n$ implies that $\beta(BG_1(C_3)) \geq 2$. It is clear from Figure 3.3 that the two red-colored vertices form a resolving set for $\beta(BG_1(C_3))$. Hence, $\beta(BG_1(C_3)) = 2$.

Conversely, assume that G is a simple connected graph with $\beta(BG_1(G)) = 2$ and $|V| < 4$. The possibilities for G are: P_2, P_3, C_3, and $K_{1,2}$. Figure 3.2 shows that $BG_1(P_2)$ is disconnected, and so the metric dimension is not defined. Figure 3.1 shows that $BG_1(P_3) = P_5$. Therefore, $\beta(BG_1(P_3)) = 1$. The graph $K_{1,2} \cong P_3$, and so $\beta(BG_1(K_{1,2})) = 1$.

□

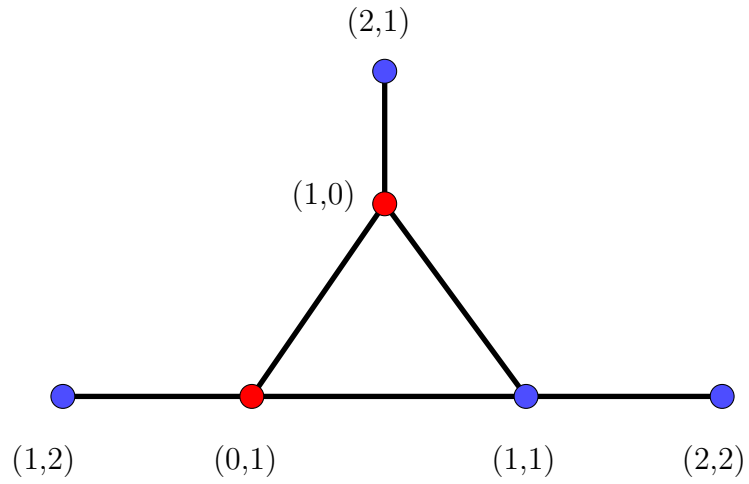


Figure 3.3: The graph $BG_1(C_3)$ with metric code

Theorem 3.3.2. *If G is a disconnected simple graph with $|V| < 5$ and $BG_1(G)$ is connected, then G must have an isolated vertex.*

Proof. For the proof, let us consider the following cases for $|V|$.

Case 1: $|V| \leq 3$,

There are no disconnected simple graphs of order less than or equal to 3 without the presence of an isolated vertex.

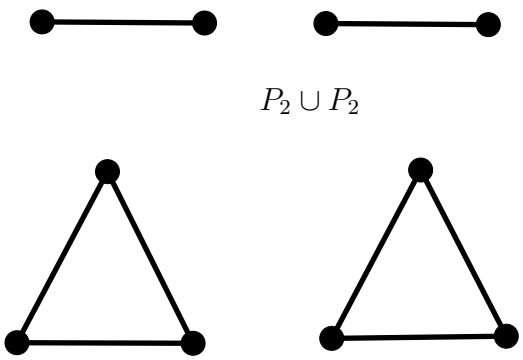
Case 2: $|V| = 4$,

There is only one disconnected graph of order 4 without an isolated vertex, and it is $P_2 \cup P_2$. Figure 3.4 shows that $BG_1(P_2 \cup P_2)$ is disconnected. Hence, the metric dimension of this graph is not defined.

□

Theorem 3.3.3. *Let G be a simple connected graph with $\beta(G) > 1$ and $|V| \leq 4$. Then, $\beta(BG_1(G)) = 3$ if and only if $|V| = 4$.*

Proof. To establish the proof, possible values of $|V|$ are divided into two distinct



$P_2 \cup P_2$

$BG_1(P_2 \cup P_2)$

Figure 3.4: The graphs $P_2 \cup P_2$ and $BG_1(P_2 \cup P_2)$

cases. Subsequently, each case is individually examined.

Case 1: $|V| \leq 3$,

There are precisely three graphs with an order less than or equal to 3: P_2, P_3, and C_3. Notably, $\beta(P_2) = 1 = \beta(P_3)$, and so P_2 and P_3 do not fulfill the specified hypothesis of this theorem. The figure 3.3 shows that the metric dimension of $BG_1(C_3)$ is 2.

Case 2: $|V| = 4$,

There are six possibilities for the graph G: K_4, P_4, C_4, $K_{1,3}$, $K_4 - e$, and $L_{3,1}$. Theorem 3.2.1 implies $\beta(BG_1(K_4)) = 3$. $\beta(P_4) = 1$, so P_4 fails to satisfy the hypothesis of the theorem. In the case of C_4, in Figure 3.5, C_4 and $BG_1(C_4)$ are depicted, and it is evident that $BG_1(C_4)$ does not form a path graph. Consequently, $\beta(BG_1(C_4)) > 1$.

The following matrix, in Table 3.1, shows the distance between each pair of vertices in $BG_1((C_4)$.

Upon examination of the matrix, it becomes evident that the set $\{V_0, V_1, V_2\}$ constitutes a resolving set, and no pair of vertices forms a resolving set. As a result,

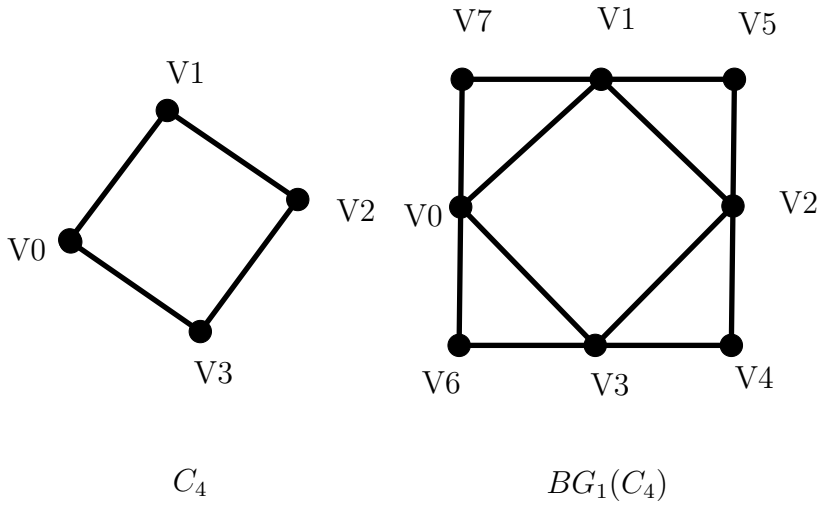


Figure 3.5: The graphs C_4 and $BG_1(C_4)$

	V0	V1	V2	V3	V4	V5	V6	V7
V0	0	1	2	1	2	2	1	1
V1	1	0	1	2	2	1	2	1
V2	2	1	0	1	1	1	2	2
V3	1	2	1	0	1	2	1	2
V4	2	2	1	1	0	2	2	2
V5	2	1	1	2	2	0	2	2
V6	1	2	2	1	2	2	0	2
V7	1	1	2	2	2	2	2	0

Table 3.1: The distance matrix of $BG_1(C_4)$

$\beta(BG_1(C_4)) = 3$.

Theorem 3.2.6 implies that $\beta(BG_1(K_{1,3})) = 3$.

Theorem 3.3.5 shows that $\beta(BG_1(K_4 - e)) = 3$.

Finally, in the case of the lollipop graph $L_{3,1}$, Figure 3.6 shows the graphs, and Table 3.2 displays the distance matrix. Upon scrutinizing the table, it becomes evident that the set of vertices $\{V_0, V_1, V_2\}$ constitutes a resolving set, while none of the vertex pairs achieves the same. Furthermore, the graph is a non-linear structure. Therefore, $\beta(BG_1(L_{3,1})) > 1$.

Consequently, $\beta(BG_1(L_{3,1})) = 3$.

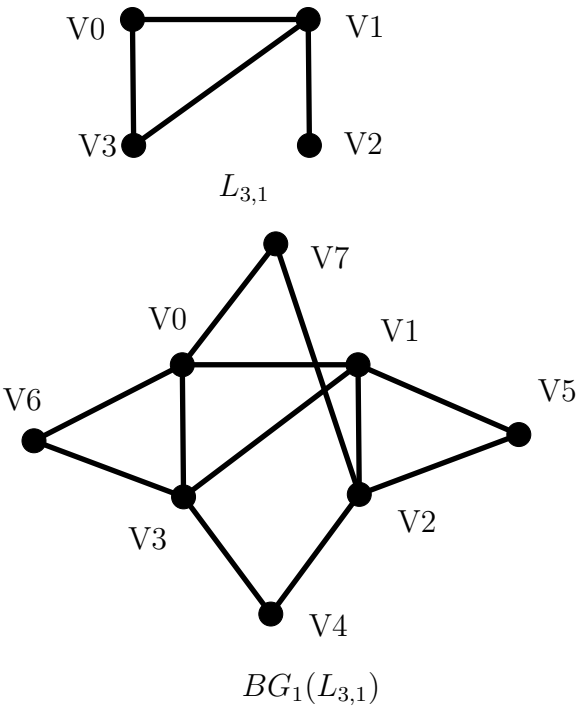


Figure 3.6: The graphs $L_{3,1}$ and $BG_1(L_{3,1})$

□

Theorem 3.3.4. *If* $| V | > 3$, *then* $\beta(BG_1(K_n - e)) \leq | V | -1$.

Proof. Let $G = K_n$ with $|V| = n$ and the vertex set $V = \{V_1, V_2, \ldots, V_n\}$. Assume V_1 and V_2 are non-adjacent in G, while all other pairs of vertices are adjacent. Let $S = \{V_1', V_2', \ldots, V_{n-1}'\}$. Consequently, $C_S(V_n') = (1, 1, \ldots, 1)$. To establish S as a resolving set, we'll examine various cases of line vertices. Let q be a line vertex.

Case 1: (q intersects with V_n' but not with either V_1' or V_2')
When $q = V_p'V_n'$, where $p \notin \{1, 2, n\}$, $C_S(q') = (1, 1, \ldots, 1, 2, 1, \ldots, 1)$, with 2 occurring in the pth position. Therefore, the code varies with the change in p, resulting in unique metric codes for these line vertices.

Case 2: (q intersects one of V_1' or V_2', but not with V_n')
Let $q = V_1'V_p'$. Then, $C_S(q') = (2, 1, 1, \ldots, 2, 1, \ldots, 1)$, where the 2's occur in the first and pth positions. Consequently, the code changes with variations in the vertex. A similar argument applies for $q = V_2'V_p'$.

	V0	V1	V2	V3	V4	V5	V6	V7
V0	0	1	2	1	2	2	1	1
V1	1	0	1	1	2	1	2	2
V2	2	1	0	2	1	1	3	1
V3	1	1	2	0	1	2	1	2
V4	2	2	1	1	0	2	2	2
V5	2	1	1	2	2	0	2	2
V6	1	2	3	1	2	2	0	2
V7	1	2	1	2	2	2	2	0

Table 3.2: The distance matrix of $BG_1(L_{3,1})$

Case 3: ($q = V_1'V_n'$ or $q = V_2'V_n'$)

If $q = V_1'V_n'$, then $C_S(q') = (2, 1, \ldots, 1)$.

If $q = V_2'V_n'$, then $C_S(q') = (1, 2, 1, \ldots, 1)$.

Case 4: (q does not intersect with V_1', V_2', or V_n')

Let $q = V_p'V_q'$, where $p, q \notin \{1, 2, n\}$. Then,

$$C_S(q') = (1, \ldots, 2, 1, \ldots, 2, 1, \ldots, 1)$$

where the 2's occur in the p^{th} and q^{th} positions, respectively. The code changes when the vertex is altered.

Hence, the codes for all vertices are distinct, indicating that S is a resolving set.

Then, $\beta(BG_1(K_n - e)) \leq \mid V \mid -1$. □

Theorem 3.3.5. *If* $|V| > 3$*, then* $\beta(BG_1(K_n - e)) = |V| - 1$.

Proof. Theorem 3.3.4 states that $\beta(BG_1(K_n - e)) \leq |V| - 1$.

Let $V = \{V_1, V_2, \cdots, V_n\}$, where V_1 and V_2 are not adjacent, but all other pairs of vertices are adjacent in G. We assume that there is a resolving set S with $|V| - 2$ elements. Let's look at all the possible cases.

Case 1: (S contains only point vertices)

Subcase 1: (V_1' and V_2' are not in S)

In this situation, the code for V_1' is $C_S(V_1') = (1, 1, \ldots, 1)$, and the code for V_2' is

exactly the same: $C_S(V_2') = (1, 1, \ldots, 1)$. Since both have the same code, S cannot be a resolving set.

Subcase 2: (V_1' and V_2' are both in S)

Here, take two vertices, V_i' and V_j', that are not in S. The codes for these two vertices are $C_S(V_i') = (1, 1, \ldots, 1)$ and $C_S(V_j') = (1, 1, \ldots, 1)$, meaning they are identical. This contradicts the definition of a resolving set because each vertex should have a different code.

Subcase 3: (S contains only one of V_1' and V_2')

Suppose V_1' is in S, but V_2' is not. Now, S will be of the form,

$S = \{V_1', V_3', \ldots, V_{i-1}', V_{i+1}', \ldots, V_n'\}$. The code for V_i' will be $C_S(V_i') = (1, 1, \ldots, 1)$, and the code for the edge $q = V_2V_i$ will also be $C_S(q') = (1, 1, \ldots, 1)$. This means $C_S(V_i') = C_S(q')$, which contradicts the idea that S is a resolving set.

The same argument holds if V_1' is not in S but V_2' is. So, in this case, there can't be a resolving set of size $|V| - 2$ that only includes point vertices.

Hence, S is not a resolving set.

Case 2: (S contains only line vertices)

Here, let's assume that S only has line vertices. There are $\binom{n}{2} - n + 1$ line vertices that are outside S. If we pick two of these line vertices, e_i' and e_j', their codes are $C_S(e_i') = (2, 2, \ldots, 2)$ and $C_S(e_j') = (2, 2, \ldots, 2)$, which are the same. This means S is not a resolving set because the codes of e_i' and e_j' should be different.

So, combining Case 1 and Case 2, we can conclude that there is no resolving set of size $|V| - 2$ that consists only of point vertices or line vertices.

Case 3: (S contains at least one point vertex and one line vertex)

In this case, there are at least three point vertices not in S. Then, there are at least $\binom{3}{2} - 1$ line vertices associated with these point vertices. A similar argument to the one made in Subcase 3 of Theorem 3.2.2 applies here, leading to a contradiction.

Therefore, $\beta(BG_1(K_n - e)) = |V| - 1$. □

Theorem 3.3.6. *If* $n \leq 6$

$$\beta(BG_1(P_n)) = \begin{cases} 1, & \textit{if } n = 1, \\ \textit{doesnotexist}, & \textit{when } n = 2, \\ n-2, & \textit{when } 3 \leq n \leq 6. \end{cases}$$

Proof. When $n = 1$, the graph $BG_1(P_1)$ consists of a single vertex, P_1. In this case, the single vertex itself acts as the metric base, resulting in $\beta(BG_1(P_1)) = 1$.

For $n = 2$, $BG_1(P_2) = P_2 \cup P_1$, which is a disconnected graph. Consequently, the metric dimension $\beta(BG_1(P_2))$ is not defined.

When $n = 3$, the Figure 3.1 shows that $BG_1(P_3)$ is P_5. Therefore, by Theorem 1.2.6, $\beta(BG_1(G)) = \beta(P_5) = 1$.

When $n = 4$, as indicated in Table 3.3, the set $S = \{V_0, V_1\}$ emerges as a metric

	V0	V1	V2	V3	V4	V5
V0	0	1	2	2	3	1
V1	1	0	1	2	2	2
V2	2	1	0	1	1	2
V3	2	2	1	0	1	1
V4	3	2	1	1	0	2
V5	1	2	2	1	2	0

Table 3.3: Distance matrix of $BG_1(P_4)$

base. Consequently, $\beta(BG_1(P_4)) = 2$.

In the context of $n = 5$, as illustrated in Table 3.4, the set $S = \{V_0, V_1, V_2\}$ stands

	V0	V1	V2	V3	V4	V5	V6	V7	V8
V0	0	1	2	2	2	3	1	1	1
V1	1	0	1	2	2	2	2	1	1
V2	2	1	0	1	2	1	2	2	1
V3	2	2	1	0	1	1	1	2	2
V4	2	2	2	1	0	1	1	1	3
V5	3	2	1	1	1	0	2	2	2
V6	1	2	2	1	1	2	0	2	2
V7	1	1	2	2	1	2	2	0	2
V8	1	1	1	2	3	2	2	2	0

Table 3.4: Distance matrix of $BG_1(P_5)$

as a metric base. This observation leads to, $\beta(BG_1(P_5)) = 3$.

	V0	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
V0	0	1	2	2	2	2	3	1	1	1	1
V1	1	0	1	2	2	2	2	2	1	1	1
V2	2	1	0	1	2	2	1	2	2	1	1
V3	2	2	1	0	1	2	1	1	2	2	1
V4	2	2	2	1	0	1	1	1	1	2	2
V5	2	2	2	2	1	0	1	1	1	1	3
V6	3	2	1	1	1	1	0	2	2	2	2
V7	1	2	2	1	1	1	2	0	2	2	2
V8	1	1	2	2	1	1	2	2	0	2	2
V9	1	1	1	2	2	1	2	2	2	0	2
V10	1	1	1	1	2	3	2	2	2	2	0

Table 3.5: Distance matrix of $BG_1(P_6)$

When $n = 6$, as depicted in Table 3.5, the selection $S = \{V_0, V_1, V_2, V_3\}$ emerges as a metric base. Thus $\beta(BG_1(P_6)) = 4$.

□

Theorem 3.3.7. *If $n > 6$, then*

$$\beta(BG_1(P_n)) = \begin{cases} 2m-1, & \text{if } n = 3m, \\ 2m, & \text{otherwise.} \end{cases}$$

Proof. Consider a path graph P_n with vertices $V = \{v_0, v_1, \ldots, v_{n-1}\}$ and edges $e_i = v_{i-1}v_i$ for $i = 1, 2, \ldots, n-1$.

case 1:(n = 3m)

In this case, first define the set $S = \{v'_0, v'_2, v'_3, v'_5, v'_6, \ldots, v'_{n-4}, v'_{n-3}\}$, containing $2m-1$ vertices. Notably, $C_S(v'_{3x-2}) = (2, 2, \ldots, 1, 1, 2, \ldots, 2)$, where the 1's occur at positions $2x-1$ and $2x$, the indices less than n. Also, $C_S(v'_{n-1}) = (2, 2, \ldots, 2)$. Considering the line vertices e'_i, $C_S(e'_i) = (1, 1, \ldots, 2, 1, \ldots, 1)$ or $(1, 1, \ldots, 2, 2, 1, \ldots, 1)$, depending on whether one or both end vertices associated to the edge are in S. The positions of 2's vary from edge to edge, ensuring distinct codes, thus confirming S as a resolving set.

Now, removing v'_0 from S, we find $C_S(v'_{n-1}) = (2, 2, \ldots, 2) = C_S(v'_0)$. Similarly, if we remove v'_2, $C_S(e'_3) = (1, 2, 1, \ldots, 1) = C_S(e'_4)$. This reasoning extends to show that removing any vertex from S renders it a non-resolving set. Therefore, S is a minimal resolving set of point vertices.

Next, if possible, consider a resolving set S with $2m - 2$ line vertices. This leaves, $3m - 1 - 2m + 2 = m + 1$ line vertices outside S. Select any two, say e'_i and e'_j, from this group. Since their codes are identical, $C_S(e'_i) = (2, 2, \ldots, 2) = C_S(e'_j)$. it implies there's no such resolving set.

Now, if possible, consider a resolving set of $2m - 2$ elements containing both point and line vertices. This implies there are at least $m + 3$ point vertices and $m + 2$ line vertices outside S. In this scenario, one can select two vertices, at least from one of the following three categories: 1. Two line vertices e'_i and e'_j from the complement of S, both having endpoints also in the complement:

$$C_S(e'_i) = (1, 1, \ldots, 1, 2, 2, \ldots, 2) = C_S(e'_j)$$

2. Two line vertices e'_i and e'_j having a common end in S and the remaining ends in the complement of S:

$$C_S(e'_i) = (1, 1, \ldots, , 2, 1, \ldots, 2, \ldots, 2) = C_S(e'_j)$$

3.Two point vertices in the complement of S that are not adjacent to any members of S:

$$C_S(v'_i) = (2, 2, \ldots, 2, 1, 1, \ldots, 1) = C_S(v'_j)$$

Where the 2's correspond to point vertices and 1's correspond to line vertices of S. As a consequence, S fails to qualify as a resolving set. Thus, there exists no resolving set of order $2m-2$, leading to the conclusion that $\beta(BG_1(P_n)) = 2m-1$.

Case 2: $(n = 3m + 1)$

Here, first introduce the set $S = \{v'_0, v'_1, v'_3, v'_4, v'_6v'_7, \ldots, v'_{3m-3}, v'_{3m-2}\}$, comprising $2m$ vertices.

The codes of point vertices: $C_S(v_{3i-1}) = (2, 2....1, 1, 2...)$, the occurrence of 1's corresponds to vertices v_{3i-2} and v_{3i}, exhibiting variation with i. Meanwhile, $C_S(v_{3m}) = (2, 2...., 2)$.

Now, the codes of line vertices:

- When both ends reside outside S, the code is $(1, 1, ...1)$.
- If only one end is within S, the code becomes $(1, 1, ...2, 1, ...1)$.
- When both ends are within S, the code is $(1, 1, ...2, 2, 1, ...1)$.
- Finally, $C_S(e_1) = (3, 2, 1....1)$.

  the list shows a dynamic code change from vertex to vertex.

Evidently, S is a resolving set with $2m$ elements.
The removal of any vertex from S results in an edge e_i, where both ends are not within S. Consequently,

$$C_S(e'_i) = (1, 1, \ldots, 1) = C_S(e_{3m})$$

. This uniformity in codes signifies a failure to meet the criteria of a resolving set. Hence, S is a minimal resolving set of this type.
There exists no resolving set S consisting solely of line vertices, as the codes of any two line vertices outside S are identical, represented as $(2, 2, ...2)$.
Now, let's consider a resolving set S comprising both point and line vertices. Employing the same arguments as in Case 1, it becomes evident that no resolving set of this type, with $2m - 1$ elements, can exist.
Consequently, $\beta(BG_1(P_n)) = 2m$

case 3:(n = 3m-1)

Consider the set $S = \{v_1, v_2, v_4, v_5, \ldots, v_{3m-5}, v_{3m-4}\}$. Arguing as in Case 1, it can be demonstrated that S is a resolving set. Moreover, none of the vertices within S can be removed while preserving its resolving property.

Furthermore, it's apparent that there is no resolving set consisting of $2m - 1$ elements comprising solely line vertices, as the codes of any two line vertices outside S become identical. Applying analogous reasoning to Case 1, it follows that there exists no resolving set of $2m-1$ elements containing both line and point vertices. Therefore, $\beta(BG_1(P_n)) = 2m$.

□

Theorem 3.3.8. *If $n \leq 5$, then $\beta(BG_1(C_n)) = n - 1$.*

Proof. When $n = 3$, $C_3 = K_3$, implying, by Theorem 3.2.2, $\beta(BG_1(C_3)) = 2$.

	V0	V1	V2	V3	V4	V5	V6	V7
V0	0	1	2	1	2	2	1	1
V1	1	0	1	2	2	1	2	1
V2	2	1	0	1	1	1	2	2
V3	1	2	1	0	1	2	1	2
V4	2	2	1	1	0	2	2	2
V5	2	1	1	2	2	0	2	2
V6	1	2	2	1	2	2	0	2
V7	1	1	2	2	2	2	2	0

Table 3.6: Distance matrix of $BG_1(C_4)$

For $n = 4$, the distance matrix of $BG_1(C_4)$ is illustrated in Table 3.6. It's apparent from the table that $S = \{V_0, V_1, V_3\}$ forms a metric base, leading to $\beta(BG_1(C_4)) = 3$.

In the case of $n = 5$, table 3.7 gives the distance matrix of $BG_1(C_5)$. a simple analysis leads to , $S = \{V_0, V_1, V_2, V_5\}$ is a metric base, hence $\beta(BG_1(C_5)) = 4$.

□

Chapter 4

Metric Dimension of $BG_1(\overline{G})$ and $BG_2(\overline{G})$ Graphs

Introduction

This chapter addresses the metric dimension of $BG_1(\overline{G})$ and $BG_2(\overline{G})$. A series of theorems are established for these expanded graphs in the context of well-known graph structures such as path graphs, cycle graphs, and star graphs. The facts derived from these theorems have been used in constructing algorithms to determine the metric dimension and related parameters of $BG_1(\overline{G})$ and $BG_2(\overline{G})$.

4.1 Analysis of $\beta(\mathbf{BG}_1(\overline{G}))$ for Path, Cycle, and Star Graphs

Theorem 4.1.1. *The set of all point vertices, except one of the pendent vertices of P_n, $n \geq 4$, constitutes a resolving set for $BG_1(\overline{P}_n)$.*

Proof. Let $S = \{v'_0, v'_1, \ldots, v'_{n-2}\}$ be the set of all point vertices except the point vertex corresponding to the pendent vertex v_{n-1} of P_n. In $BG_1(\overline{P}_n)$, the code for

v'_{n-1} is $C_S(v'_{n-1}) = (1, 1, \ldots, 1, 2)$.

Let e' be any line vertex of $BG_1(\overline{P}_n)$. Assume that e' corresponds to the edge v_iv_j in $\overline{P}_n$, where $i < j$, and note that $j \neq i + 1$. We now consider two cases for i and j:

Case 1: $j \neq n - 1$

In this case, the shortest paths from e' to v'_i and v'_j are $e'v'_0v'_i$ and $e'v'_{n-1}v'_j$, respectively. Therefore, the code for e' is $C_S(e') = (1, 1, \ldots, 1, 2, 1, \ldots, 1, 2, 1, \ldots, 1)$, where the 2's appear in the i^{th} and j^{th} positions. Since the positions of the 2's vary based on i and j, the codes will be different for different line vertices.

Case 2: $j = n - 1$

In this case, the code for the line vertex e', corresponding to the edge $e = v_iv_{n-1}$ in G, in $BG_1(\overline{P}_n)$, is $C_S(e') = (1, 1, \ldots, 1, 2, 1, \ldots, 1)$, where the 2 occurs in the i^{th} position. The code will vary as i varies.

In each case, the metric code differs, and thus S constitutes a resolving set. □

Corollary 4.1.2. *If $n \geq 4$, then $\beta(BG_1(\overline{P}_n)) \leq n - 1$.*

Proof. The theorem 4.1.1 establishes that the graph has a resolving set comprising n−1 elements. Since the metric dimension of a graph is defined as the cardinality of the minimal resolving set, the result follows obviously. □

Theorem 4.1.3. *If P_n is the path graph,*

$$\beta(BG_1(\overline{P}_n)) = \begin{cases} 1 & \text{If } n = 1 \\ \text{Does not exist} & \text{when } n = 2 \text{ or } 3 \\ n - 1 & \text{if } n > 3. \end{cases}$$

Proof. When n=1, there is only one vertex in $\overline{P}_n$, which can be identified with the code 0, hence $\beta(\mathrm{BG}_1(\overline{P}_n))$ =1.

When n=2 or 3, $BG_1(\overline{P}_n)$ is disconnected, therefore $\beta(\mathrm{BG}_1(\overline{P}_n))$ does not exist.

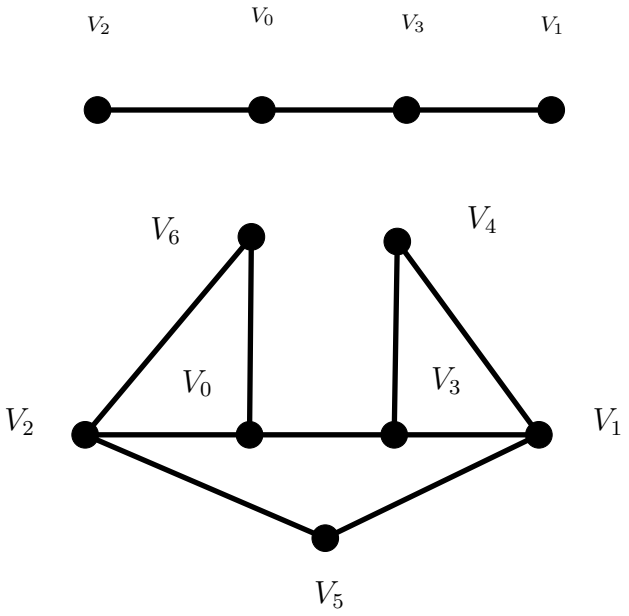


Figure 4.1: The graphs $\overline{P}_4$ and $BG_1(\overline{P}_4)$

	V0	V1	V2	V3	V4	V5	V6
V0	0	2	1	1	2	2	1
V1	2	0	2	1	1	1	2
V2	1	2	0	2	2	1	1
V3	1	1	2	0	1	2	2
V4	2	1	2	1	0	2	2
V5	2	1	1	2	2	0	2
V6	1	2	1	2	2	2	0

Table 4.1: The distance matrix of $BG_1(\overline{P}_4)$

When n = 4, the figure 4.1 gives $BG_1(\overline{P}_n)$ and table 4.1 gives the distance matrix of $BG_1(\overline{P}_n)$). The table specifies that $\{ V_0, V_1, V_2\}$ is metric base. Therefore, $\beta(\mathrm{BG}_1(\overline{P}_4) = 3$.

Now consider the case when $n \geq 5$.

The corollary 4.1.2 establishes,

$$\beta(BG_1(\overline{P}_n)) \leq n - 1 \tag{1}$$

Theorem 4.1.1 gives a resolving set, S=$\{V_0', V_1'..., V_{n-2}'\}$. If we remove one vertex V_i', $0 < i < n - 2$, from S to get a new vertex set S', then $C_{S'}(V_i') =$

$(1, 1, ..., 12, 2, 1, ..., 1)$. Where the 2's occur in $i - 1^{th}$ and $i + 1^{th}$ positions. Let e' be the line vertex corresponding to the edge $V_{i-1}V_{i+1}$ in $\overline{P}_n$, then $C_{S'}(e') = (1, 1, ..., 12, 2, 1, ..., 1)$. Therefore, $C_{S'}(V_i') = C_{S'}(e')$. Hence, S' cannot fulfill the role of a resolving set.

Let S'' be the set obtained from S by removing V_0, then $C_{S''}(V_0') = (2, 1, 1, ..., 1)$. Let f' be the line vertex corresponding to the edge V_1V_{n-1} of $\overline{P}_n$, then $C_{S''}(f') = (2, 1, 1, ..., 1) = C_{S''}(V_0')$. Hence,$S''$ is not a resolving set.

Let V_{n-2} is removed from S to obtain S'''. Let h' be the line vertex corresponding to the edge $V_{n-3}V_{n-1}$ of $\overline{P}_n$, then $C_{S'''}(h') = (1, 1, ..., 1, 2) = C_{S'''}(V_{n-2}')$. Therefore, S''' fails to be a resolving set.

The role of V_0 and V_{n-1} in $\overline{P}_n$ are identical.

Thus, there is no resolving set with $n - 2$ point vertices.

Next examine the scenario where S comprises $n - 2$ line vertices. The total count of edges in $\overline{P}_n$ amounts to $\binom{n}{2} - (n - 1) = \frac{n(n-1)}{2} - (n - 1) = \frac{(n-1)(n-2)}{2}$. Consequently, $V' - S$ is expected to contain $\frac{(n-1)(n-2)}{2} - (n - 2) = \frac{(n-2)(n-3)}{2}$ line vertices.Given that $n \geq 5$, it follows that $V' - S$ must contain a minimum of three line vertices. choose any two such line vertices e_1' and e_2', then $C_S(e_1') = (2, 2, ..., 2) = C_S(e_2')$. Therefore S cannot be a resolving set.

Now assume that S is a resolving set with $n - 2$ elements and contains both point vertices and line vertices.

In this case, there should be at least 3 point vertices in $V' - S$, choose two such point vertices V_p' and V_q'. Let V_i' be any point vertex in S. Choose p, q and i in such a way that the line vertices $f_1' = V_i'V_p'$ and $f_2' = V_i'V_q'$ exist. Then, $C_S(f_1') = (1, 1, ..., 2, 1, 1, .., 1, 2, 2..., 2) = C_S(f_2')$. Where the first 2 is in the i^{th} position and the other $2'$s correspond to the line vertices of S. Therefore, S is not a resolving set.

Hence, there is no resolving set S with $n - 2$ elements.

It follows from the property that any super set of a resolving set is itself a resolving

set, that a resolving set with fewer than $n-1$ elements cannot exist.

Thus,

$$\beta(BG_1(\overline{P}_n)) \geq n-1 \tag{2}$$

From (1) and (2),

$$\beta(\mathrm{BG}_1(\overline{P}_n)) = n-1.$$

□

Theorem 4.1.4. *If C_n is the cycle graph with n vertices,*

$$\beta(BG_1(\overline{C}_n)) = \begin{cases} 1 & \text{If } n = 1 \\ \text{Does not exist} & \text{when } 2 < n < 5 \\ n\text{-}1 & \text{if } n = 5 \text{ or } n = 6 \\ n & \text{if } n > 6. \end{cases}$$

Proof. Consider the graph C_1, which is a loop. Its complement is P_1, consisting of a single vertex. This sole vertex is identifiable by the code 0 and hence, $\beta(BG_1(\overline{C}_1))) = 1$.

In case of the graphs C_2, C_3, and C_4; $BG_1(\overline{C}_2)$,$BG_1(\overline{C}_3)$ and $BG_1(\overline{C}_4)$ are all

	V0	V1	V2	V3	V4	V5	V6	V7	V8	V9
V0	0	2	1	1	2	2	2	1	1	1
V1	2	0	2	1	1	1	1	2	2	1
V2	1	2	0	2	1	2	1	1	1	2
V3	1	1	2	0	2	1	2	2	1	1
V4	2	1	1	2	0	1	1	1	2	2
V5	2	1	2	1	1	0	2	2	2	2
V6	2	1	1	2	1	2	0	2	2	2
V7	1	2	1	2	1	2	2	0	2	2
V8	1	2	1	1	2	2	2	2	0	2
V9	1	1	2	1	2	2	2	2	2	0

Table 4.2: The distance matrix of $BG_1(\overline{C}_5)$

disconnected. Consequently, the metric dimension cannot be defined.

If $n = 5$, then $\overline{C}_5$ has 5 point vertices and $\binom{5}{2} - 5 = 5$ line vertices. The distance matrix of this graph with 10 vertices is given in Table 4.2. It's evident from Table 4.2 that the set $\{V_0, V_1, V_3, V_6\}$ forms a resolving set, while no set of 3 vertices serves as a resolving set. Therefore, $\beta(BG_1(\overline{C}_5))) = 4$.

. When $n = 6$, the graph $\overline{C}_6$ has 6 point vertices and $\binom{6}{2} - 6 = 9$ line vertices. The distance matrix of this graph with 14 vertices is given in Table 4.4. The Table 4.4 tells that the set $\{V_0, V_1, V_3, V_4, V_7\}$ forms a metric base, while no set of 4 is a resolving set. Consequently, $\beta(BG_1(\overline{C}_5))) = 5$.

	V0	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14
V0	0	2	1	1	1	2	2	2	2	1	1	1	1	1	1
V1	2	0	2	1	1	1	1	1	1	2	2	2	1	1	1
V2	1	2	0	2	1	1	2	1	1	1	1	1	2	2	1
V3	1	1	2	0	2	1	1	2	1	2	1	1	1	1	2
V4	1	1	1	2	0	2	1	1	2	1	2	1	2	1	1
V5	2	1	1	1	2	0	1	1	1	1	1	2	1	2	2
V6	2	1	2	1	1	1	0	2	2	2	2	2	2	2	2
V7	2	1	1	2	1	1	2	0	2	2	2	2	2	2	2
V8	2	1	1	1	2	1	2	2	0	2	2	2	2	2	2
V9	1	2	1	2	1	1	2	2	2	0	2	2	2	2	2
V10	1	2	1	1	2	1	2	2	2	2	0	2	2	2	2
V11	1	2	1	1	1	2	2	2	2	2	2	0	2	2	2
V12	1	1	2	1	2	1	2	2	2	2	2	2	0	2	2
V13	1	1	2	1	1	2	2	2	2	2	2	2	2	0	2
V14	1	1	1	2	1	2	2	2	2	2	2	2	2	2	0

Table 4.3: The distance matrix of $BG_1(\overline{C}_6)$

In the case when $n > 6$, according to corollary 3.1.3,

$$\beta(BG_1(\overline{C}_n)) \leq n \tag{1}$$

If possible, assume that S is a resolving set of $n-1$ point vertices. Let V_i be the only point vertices left outside S. Then, $C_S(V_i') = (1, ..., 1, 2, 2, 1..., 1)$. The $2's$ appear in $i - 1^{th}$ and $i + 1^{th}$ positions. Consider the line vertex e' of $BG_1(\overline{C}_n)$, corresponding to the edge $V_{i-1}V_{i+1}$ of $\overline{C}_n$. Then, $C_S(e') = (1, ..., 1, 2, 2, 1..., 1)$. Consequently, $C_S(V_i') = C_S(e')$ and hence there is no resolving set of point vertices

with order $n-1$.

Next, assume that there is a resolving set of $n-1$ line vertices. There are $\binom{n}{2}-n$ line vertices in $BG_1(\overline{C}_n)$ and $\frac{n(n-1)}{2}-n-(n-1)=\frac{n^2-5n+2}{2}$ line vertices lie in $V'-S$. Let e_1',$e_2' \in V'-S$. Then,

$C_S(e_1')=(2,2....2)=C_S(e_2')$.

This contradicts the fact that S is a resolving set.

Now assume that S is a resolving set containing both line vertices and point vertices. Let there be p point vertices and l line vertices in S, then $p \leq n-2$. Choose two point vertices $V_x', V_y' \in V'-S$. Choose a point vertex V_z' with the following properties,

1. $V_z' \in S$
2. $z \not\equiv i+1 \pmod{n}$ and $z \not\equiv i-1 \pmod{n}$, $i=x,y$
3. $e_1', e_2' \in V'-S$, where e_1' and e_2' are the line vertices corresponding to the edges V_xV_z and V_yV_z of $\overline{C}_n$, respectively.

Then, $C_S(e_1')=(1,...,1,2,1,1...1,2,2...,2)=C_S(e_2')$. where the 2's corresponding to V_z' and the line vertices in S. The 1's correspond to the point vertices of S other than V_z'. This contradicts the assumption that S is a resolving set.

Thus, there is no resolving set with $n-1$ vertices. Therefore,

$$\beta(BG_1(\overline{C}_n)) \geq n \tag{2}$$

From (1) and (2) $\beta(BG_1(\overline{C}_n)))=n$.

□

Theorem 4.1.5. *If K_n is the complete graph with n vertices,*

$$\beta(BG_1(\overline{K}_n)) = \begin{cases} 1 & \text{If } n=1 \\ \text{Does not exist} & \text{when } n \geq 2 \end{cases}$$

Proof. When n=1, $\overline{K_1}$ has a single vertex and no edges.Therefore, $BG_1(\overline{K}_1)$ is isomorphic to K_1. Hence, $\beta(BG_1(\overline{K}_1))) = 1$.

When $n \geq 2$, $\overline{K_n}$ has no edges. Consequently, $BG_1(\overline{K}_n)$ is disconnected and hence $\beta(BG_1(\overline{K}_1)))$ is not defined. □

Theorem 4.1.6. *If $n > 3$, then the set of all point vertices except the center vertex and one of the leaves of $K_{1,n}$ forms a resolving set for $BG_1(\overline{K}_{1,n})$.*

Proof. Let $\{V_0, V_1..., V_n\}$ be the vertices of $K_{1,n}$, where V_0 is the center vertex. Assume the vertex V_0' and V_k' are removed to form a subset S of V'. In $BG_1(\overline{K}_{1,n})$, rename the vertices V_k' as V_n' and V_n' as V_k'.

Then, S=$\{V_1', V_2'..., V_{n-1}'\}$.

$C_S(V_0') = (2, 2..., 2)$ and $C_S(V_n') = (1, 1..., 1)$.

Therefore the metric codes of every point vertices are all different.

Next assume that e' be any line vertex of $BG_1(\overline{K}_{1,n})$.

Let $e = V_iV_j$ in $\overline{K}_{1,n}$.

Case 1: $(i, j \neq n-1)$

In this case, $C_S(e') = (1, ..., 1, 2, 1...1, 2, 1..., 1)$, where the 2's occur in i^{th} and j^{th} positions. The position of 2's will vary in accordance with i and j, that is codes of different line vertices are different.

Case 2:(Either $i = n-1$ or $j = n-1$)

In this case, when $i = n-1$, $C_S(e') = (1, ..., 1, 2, 1..., 1)$, where the 2 occur in i^{th} position. The position of 2 will vary in accordance with i, that is codes of different line vertices of this kind are different.

The case of $j = n-1$ is similar.

Thus, codes of every vertices in $BG_1(\overline{K}_{1,n})$ are different.

So S is a resolving set.

□

Theorem 4.1.7.

$$\beta(BG_1(\overline{K}_{1,n})) = \begin{cases} \textit{Does not exist} & \textit{If } n \leq 2 \\ n & \textit{if } n = 3 \\ n-1 & \textit{if } n > 3. \end{cases}$$

Proof. $BG_1(\overline{K}_{1,1})$ and $BG_1(\overline{K}_{1,2})$ are both disconnected. Consequently, $\beta(\mathrm{BG}_1(\overline{K}_{1,1}))$ and $\beta(\mathrm{BG}_1(\overline{K}_{1,2}))$ does not exist.

When $n = 3$, the Figure 4.2 gives a plot of $BG_1(\overline{K}_{1,3})$ and Table 4.4 gives its

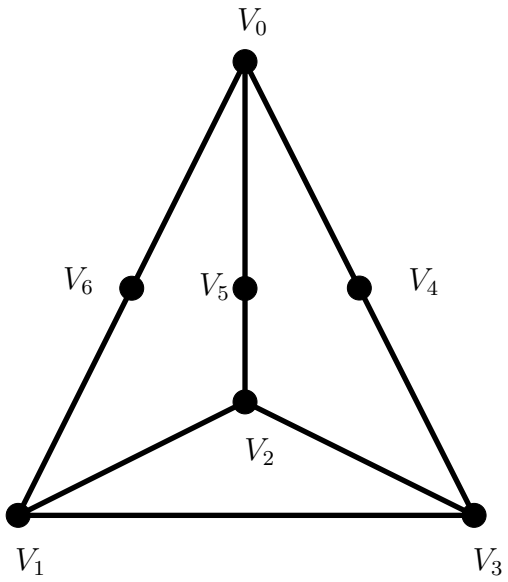


Figure 4.2: The graph $BG_1(\overline{K}_{1,3})$

distance matrix. It's evident from Table 4.4 that the set $\{V_0', V_1', V_2'\}$ is a resolving set, and no subset of the set of vertices with fewer members can act as a resolving set. Consequently, $\beta(\mathrm{BG}_1(\overline{K}_{1,3})) = 3$.

When $n > 3$, theorem 4.1.6 implies that,

$$\beta(BG_1(\overline{K}_{1,n})) \leq n - 1 \tag{1}$$

Now to prove the claim that there is no resolving set with $n - 2$ vertices, first

we eliminate the possibility of a resolving set comprising $n-2$ point vertices.

If possible assume that S is a resolving set of this type. Here exactly 3 point vertices V_i', V_j' and V_k' in $V'-S$, where $i,j \neq 0$. If $V_0' \in V'-S$ then,

$C_S(V_i') = (1,1...,1) = C_S(V_j')$

If $V_0' \in S$ then,

$C_S(V_i') = (2,1...,1) = C_S(V_j')$.

The 2 corresponds to V_0'.This contradicts the fact that S is a resolving set.

Secondly, assume that S is a resolving set with $n-2$ line vertices. There are

	V_0'	V_1'	V_2'	V_3'	V_4'	V_5'	V_6'
V_0'	0	2	2	2	1	1	1
V_1'	2	0	1	1	2	2	1
V_2'	2	1	0	1	2	1	2
V_3'	2	1	1	0	1	2	2
V_4'	1	2	2	1	0	2	2
V_5'	1	2	1	2	2	0	2
V_6'	1	1	2	2	2	2	0

Table 4.4: The distance matrix of $BG_1(\overline{K}_{1,3})$

$\frac{(n)(n+1)}{2} - n$ line vertices in $BG_1(\overline{K}_{1,n})$ and so, $V'-S$ contains $\frac{(n)(n+1)}{2} - 2n + 2$ line vertices, which is at least 4 when $n > 3$. Choose any two line vertices, $e_1', e_2' \in V'-S$ then,

$C_S(e_1') = (2,2...,2) = C_S(e_2')$.

This is not possible in a resolving set S.

Finally, assume that S is a resolving set with $n-2$ elements, containing both line vertices and point vertices. Let's discuss the following possible cases.

Case 1: (S contains exactly one line vertex)

Let the line vertex in S be the line vertex corresponding to the edge $V_p'V_q'$ of $\overline{K}_{1,n}$.

Subcase 1: ($V_p', V_q' \in S$)

Choose V'_r and V'_s, $r \neq 0$ and $s \neq 0$

$$C_S(V'_r) = C_S(V'_s) = \begin{cases} (2,1,1,...,1) & \text{If } V'_0 \in S \\ (1,1,1,...,1) & \text{If } V'_0 \notin S. \end{cases}$$

Subcase 2: $(V'_p \in S, V'_q \notin S)$

Choose the point vertex $V'_r \notin S$, $r \neq 0, q$ and the line vertex $e' = V'_r V'_q$. Then,

$$C_S(V'_q) = C_S(e') = \begin{cases} (2,1,1,...,1,2) & \text{If } V'_0 \in S \\ (1,1,1,...,1,2) & \text{If } V'_0 \notin S. \end{cases}$$

The case $V'_p \notin S, V'_q \in S$ is similar to this Subcase.

Subcase 3: $(V'_p \notin S, V'_q \notin S)$

In this case,

$$C_S(V'_p) = C_S(V'_q) = \begin{cases} (2,1,1,...,1,2) & \text{If } V'_0 \in S \\ (1,1,1,...,1,2) & \text{If } V'_0 \notin S. \end{cases}$$

Case 2:(S contains more than one line vertex)

In this case, there will be at least 5 point vertices in $V' - S$, so there will be at least 4 point vertices other than V'_0 in $V' - S$. Consider the $\binom{4}{2} = 6$ line vertices between them. If at least two of them,e'_1 e'_2, is in $V' - S$ then,

$C_S(e'_1) = (1, 1, ..., 1, 2, 2....2) = C_S(e'_2)$, where the 1's correspond to the point vertices of S and 2's correspond to the line vertices of S.

If no pair of these 6 line vertices lies in $V' - S$ then, S contains 5 line vertices and hence there will be at least 7 point vertices in S other than V'_0. Consider the $\binom{7}{2} = 21$ line vertices formed by these point vertices. Repeating the above

arguments, either we reach an S consisting of line vertices only or
$C_S(e'_1) = (1, 1, ..., 1, 2, 2....2) = C_S(e'_2)$, which is a contradiction.
Thus, there is no resolving set of order $n-2$, therefore

$$\beta(BG_1(\overline{K}_{1,n})) \geq n-1 \tag{2}$$

From (1) and (2),
$\beta(\mathrm{BG}_1(\overline{K}_{1,n})) = n-1$.

□

4.2 The Basic Properties of $\beta(BG_2(\overline{G}))$

Theorem 4.2.1. *Let G be a graph with $n+1$ vertices, where $K_{1,n}$ is a subgraph of G. Then, $\beta(BG_1(\overline{G}))$ does not exist.*

Proof. Assume G is a graph with $n+1$ vertices and $K_{1,n}$ is a subgraph of G. There exists a vertex, denoted v_1, in G with degree n. This means v_1 is adjacent to all other vertices in the graph. Since the total number of vertices is $n+1$, v_1 is an isolated vertex in the complement graph $\overline{G}$.

By the definition of $BG_2(G)$, none of the vertices in $BG_2(\overline{G})$ are adjacent to v'_1 (the point vertex corresponding to v_1).
Therefore, $BG_2(\overline{G})$ is disconnected.

As a result, the metric dimension $\beta(BG_2(\overline{G}))$ does not exist. □

Corollary 4.2.2. *Let K_n denote the complete graph with $n > 1$ vertices. Then, $\beta(BG_2(\overline{K_n}))$ does not exist.*

Proof. Let m be defined as $n = m+1$. In this case, every vertex in the complete graph K_n has degree m. Therefore, $K_{1,m}$ is a sub graph of K_n. According to the Theorem 4.2.1, the metric dimension $\beta(BG_2(\overline{K_n}))$ does not exist. □

Corollary 4.2.3. *Let $K_{1,n}$ denote the star graph. Then, $\beta(BG_2(\overline{K_{1,n}})$ does not exist.*

Proof. Let $K_{1,n}$ be the complete bipartite graph. $K_{1,n}$ directly fulfills the hypothesis of the Theorem 4.2.1, it follows that $\beta(BG_2(\overline{K_{1,n}}))$ does not exist.

□

Theorem 4.2.4. *Let P_n be the path graph with n vertices. Then, $\beta(BG_2(\overline{P_n})) \leq n-1$.*

Proof. Let $V = \{v_0, v_1, \ldots, v_{n-1}\}$ be the vertex set of P_n. Define the set $S = \{v'_0, v'_1, \ldots, v'_{n-2}\}$, which includes all point vertices of $BG_2(\overline{P_n})$ except v'_{n-1}. The code for v'_{n-1} with respect to S, in $BG_2(\overline{P_n})$, is given by:

$$C_S(v'_{n-1}) = (1, 1, \ldots, 1, 2).$$

Next, consider the line vertices. Let $e = v_iv_j$ be an edge in $\overline{P_n}$. We examine two cases:

1. **Case 1**: If neither i nor j equals $n-1$,

Here the code for e' with respect to S, in $BG_2(\overline{P_n})$ is:

$$C_S(e') = (2, 2, \ldots, 1, 2, \ldots, 1, 2, \ldots, 2),$$

where the 1's appear in the i-th and j-th positions. Since these positions vary with each edge, the code for each line vertex is unique.

2. **Case 2**: If $j = n-1$,

In this case, the code for e' in $BG_2(\overline{P_n})$ is:

$$C_S(e') = (2, 2, \ldots, 1, 2, \ldots, 2),$$

where the 1 appears in the i-th position, making the code unique for each such edge.

Since the codes of all vertices and edges are distinct, it follows that S is a resolving set. Therefore,

$$\beta(BG_2(\overline{P_n})) \leq n - 1.$$

□

Theorem 4.2.5. *Let C_n be the cycle graph. Then, $\beta(BG_2(\overline{C_n})) \leq n - 1$.*

Proof. Let the vertex set of C_n be $V = \{v_0, v_1, \ldots, v_{n-1}\}$, and define $S = \{v'_1, v'_2, \ldots, v'_{n-1}\}$, the set of all point vertices of $BG_2(\overline{C_n})$ except for v'_0.
First, for the vertex v'_0, the metric code relative to S is:

$$C_S(v'_0) = (2, 1, 1, \ldots, 1),$$

where the 2 appears in the final position, ensuring that v'_0 is distinguished from other vertices in S.

Next, for any line vertex e', which corresponds to an edge v_iv_j, $i < j$ in $\overline{C_n}$, we consider the following cases:

- **Case 1:** $i \neq 0$

  In this case, the metric code for the line vertex e' is:

  $$C_S(e') = (2, 2, \ldots, 1, 2, \ldots, 1, 2, \ldots),$$

  where the 1's occur in the i-th and j-th positions. Since the positions of the 1's vary depending on i and j, each edge has a unique code.

- **Case 2:** $i = 0$

Here, the metric code becomes:

$$C_S(e') = (2, 2, \dots, 1, 2, \dots),$$

where the 1 appears in the j-th position. Again, the code differs depending on j, ensuring that each line vertex has a unique code.

Therefore, the set S serves as a valid resolving set, and the metric dimension of $BG_2(\overline{C_n})$ satisfies:

$$\beta(BG_2(\overline{C_n})) \leq n - 1.$$

□

Chapter 5

Algorithmic Perspectives on Metric Dimension in $BG_1(G)$ and $BG_2(G)$ Graphs

Introduction

Efficient algorithms and the computer programs based on them, for computing metric dimension, are crucial across various fields. They help in tasks like network design, computational biology, and communication networks. Understanding a graph's metric dimension is needed for solving many real-world problems, such as optimizing resource allocation, improving fault tolerance, and understanding biological networks. This chapter brings together theoretical advancements in the previous chapters and their practical implementations, leading to new algorithms for computing metric dimension and related measures for both $BG_1(G)$ and $BG_2(G)$ graphs. Additionally, the Appendix includes carefully crafted code implementations for the algorithms developed in this chapter. The computer programs are capable of bringing out the structural properties within these complex networks.

5.1 Algorithmic Challenges in $\beta(BG_1(G))$

The algorithm for $BG_1(G)$, aimed at finding the Metric dimension of the Boolean graph of a given graph G, is presented here. It is essential to note that the considered graph must be a simple connected one.

The following are the input parameters required by the algorithm:

1. The number of vertices of G (which must be greater than 4).
2. The upper triangular part, a_{ij}, $i < j$, of the adjacency matrix.

The output includes the following:

1. Adjacency matrix of $BG_1(G)$.
2. The distance matrix of $BG_1(G)$.
3. Metric dimension of $BG_1(G)$.
4. A Metric base of $BG_1(G)$.

This algorithm provides a comprehensive set of outputs, which gives ideas about the structural properties of Boolean graph $BG_1(G)$. It helps for a deeper understanding and analysis of this graph structure, which can be valuable in various domains.

5.1.1 The Computational Framework

Step 1 : Start.

Step 2: Declare two dimensional arrays; $g1a$, $g1b$ and $g1d$.

Declare the integer variables; $g1e = 0$, n, r, i, j, k, $g1flag = 0$.

Step 3: Enter the number of vertices of the simple connected graph G with more than 4 vertices; Read $n > 4$.

Step 4: Enter the upper triangular part of the adjacency matrix of G excluding the diagonal elements.

for($i = 0, i < n, i++$)

{

for($j = i + 1, j < n, j++$)

{

Read $g1a_{ij}$.

(The upper triangular part of the adjacency matrix of G)

Put $g1a_{ii} = 0$.

If $g1a_{ij} = 1$,

{

$g1e = g1e + 1$, $g1d_{ij} = 1$, $g1b_{ij} = 1$.

$g1b_{i,n+g1e-1} = 0$, $g1b_{j,n+g1e-1} = 0$,

$g1d_{i,n+g1e-1} = 2$, $g1d_{j,n+g1e-1} = 2$.

for $k = 0$ to $k = n - 1$,

if k is different from i and j,

then set $g1b_{k,n+g1e-1} = 1$, $g1d_{k,n+g1e-1} = 1$.

}

If $g1a_{ij} = 0$, then set $g1d_{ij} = 2$, $g1b_{ij} = 0$.

}

}

Step 5: For i and j from n to $n + g1e - 1$,

If $i \neq j$ then Set $g1b_{ij} = 0$ and $g1d_{ij} = 2$.

Step 6: For i from 0 to $n + g1e - 1$ and j from i to $n + g1e - 1$,

set $g1b_{ii} = 0$, $g1d_{ii} = 0$. $g1a_{ji} = g1a_{ij}$, $g1b_{ji} = g1b_{ij}$, $g1d_{ji} = g1d_{ij}$.

Step 7: for(int $i = 0, i < n, i++$)

{

for($intj = i + 1, j < n; j++$)

{

$g1cs = 0$, $g1rs = 0$.

if($a_{i,j} = 1$)

{

$g1s = g1s + 1$.

for(int $k = 0, k < n, k++$)

{

$g1rs = g1rs + g1a_{i,k}$, $g1cs = g1cs + g1a_{k,j}$.

}

if($g1rs = 1$)

{

$g1d_{i,n-1+g1s} = 3$, $g1d_{n-1+g1s,i} = 3$.

}

if($g1cs = 1$)

{

$d_{n-1+g1s,j} = 3$, $d_{j,n-1+g1s} = 3$.

}

}

}

}

Step 8: For $r = 2$ to $r = n$,

{

for $k = 1$ to ${}^{n+g1e}C_r$,

{

Consider each ${}^{n+g1e}C_r$ combinations of vertices of $BG_1(G)$.

Let the k^{th} combination of r vertices be $(v_{\alpha_1}, v_{\alpha_2}, \cdots, v_{\alpha_r})$

Let $v_{\beta_1}, v_{\beta_2}, \cdots, v_{\beta_{n+g1e-r}}$ be the vertices

outside the current combination.

$g1flag = 0$.

for $i = 1$ to $n + g1e - r - 1$

{

if $g1flag = 1$,break;

for $j = i + 1$ to $n + g1e - r$

{

if(($g1d_{\beta_{i\alpha_1}}, g1d_{\beta_{i\alpha_2}}, \cdots, g1d_{\beta_{i\alpha_r}}$) =

$(g1d_{\beta_{j\alpha_1}}, g1d_{\beta_{j\alpha_2}}, \cdots, g1d_{\beta_{j\alpha_r}})$)

{$g1flag = 1$,break.}

if($i = n + g1e - r - 1$ and $j = n + g1e - r$) go to Step 9.

}

}

}

}

Step 9: Print $[g1b_{ij}]$ -Adjacency matrix of $BG_1(G)$.

Print $[g1d_{ij}]$ - Distance matrix of $BG_1(G)$.

Print r -Metric Dimension of $BG_1(G)$.

Print { $v_{\alpha_1}, v_{\alpha_2}, \cdots, v_{\alpha_r}$}-Metric base of $BG_1(G)$.

Step 10:Stop.

Note:

A carefully crafted code of this algorithm is included in Appendix B.

5.1.2 Metric Dimension Computation for P_5

In this section, an example and figure-based illustration are provided to demonstrate that the output of the algorithm is correct. Actually, it is the output generated by the input when we execute the code provided in Appendix B.

Example 5.1.1.

Consider the graph P_5. Then inputs and outputs will be as follows,

Input:

Enter the number of vertices of the simple connected graph G with more than 4 vertices.

5

Enter the upper triangular part of the adjacency matrix of G excluding the diagonal elements.

1

0

0

0

1

0

0

1

0

1

Based on this input the algorithm generates the following output,

Output:

Adjacency matrix of $BG_1(G)$

	V0	V1	V2	V3	V4	V5	V6	V7	V8
V0	0	1	0	0	0	0	1	1	1
V1	1	0	1	0	0	0	0	1	1
V2	0	1	0	1	0	1	0	0	1
V3	0	0	1	0	1	1	1	0	0
V4	0	0	0	1	0	1	1	1	0
V5	0	0	1	1	1	0	0	0	0
V6	1	0	0	1	1	0	0	0	0
V7	1	1	0	0	1	0	0	0	0
V8	1	1	1	0	0	0	0	0	0

Table 5.1: The adjacency matrix of $BG_1(P_5)$

Distance matrix of $BG_1(G)$

	V0	V1	V2	V3	V4	V5	V6	V7	V8
V0	0	1	2	2	2	3	1	1	1
V1	1	0	1	2	2	2	2	1	1
V2	2	1	0	1	2	1	2	2	1
V3	2	2	1	0	1	1	1	2	2
V4	2	2	2	1	0	1	1	1	3
V5	3	2	1	1	1	0	2	2	2
V6	1	2	2	1	1	2	0	2	2
V7	1	1	2	2	1	2	2	0	2
V8	1	1	1	2	3	2	2	2	0

Table 5.2: The distance matrix of $BG_1(P_5)$

Metric dimension of $BG_1(G)$ =3.

A Metric base for $BG_1(G)$ is { v0, v1, v2 }.

The following figures, 5.1 and 5.2 illustrate and validate this output.

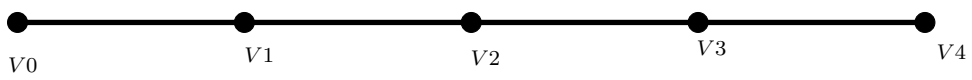


Figure 5.1: The graph P_5

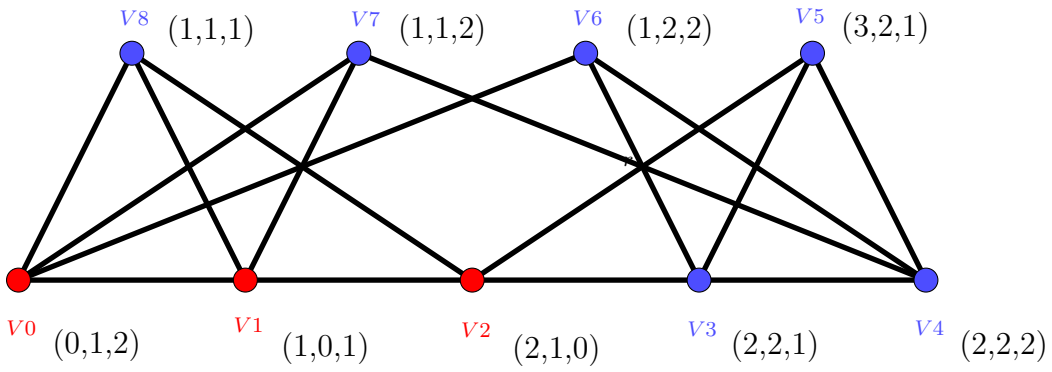


Figure 5.2: The graph $BG_1(P_5)$ with metric codes

5.2 An Algorithmic Solution for $\beta(BG_2(G))$

In this section, an algorithm is introduced to simplify the complexities of the metric dimension problems in the Boolean graph ($BG_2(G)$).

Simply provide the number of vertices of G and the upper triangular part of its adjacency matrix, a_{ij}, where $i < j$, and witness the following output:

1. Adjacency matrix of $BG_2(G)$.

2. The distance matrix of $BG_2(G)$.

3. The metric dimension of $BG_2(G)$.

4. A metric base of $BG_2(G)$, providing a set of vertices that defines the metric dimension of $BG_2(G)$.

This algorithm provides many information about $BG_2(G)$ graphs and their applications.

5.2.1 The Procedural Framework

Step 1 : Start.

Step 2: Declare two dimensional arrays; $g2a$, $g2b$ and $g2d$.

Declare the integer variables; $g2e = 0$, n, r, i, j, k, $g2flag = 0$.

Step 3: Enter the number of vertices of the simple connected

graph G with more than 2 vertices, Read $n > 2$.

Step 4: Enter the upper triangular part of the adjacency matrix of G

excluding the diagonal elements.

for($i = 0, i < n, i++$)

{

for($j = i + 1, j < n, j++$)

{

If $g2a_{ij} = 1$,

{

$g2e = g2e + 1$, $g2d_{ij} = 1$, $g2b_{ij} = 1$,

$g2b_{i,n+g2e-1} = 1$, $g2b_{j,n+g2e-1} = 1$,

$g2d_{i,n+g2e-1} = 1$, $g2d_{j,n+g2e-1} = 1$.

for $k = 0$ to $k = n - 1$,

if k is different from i and j,

then set $g2b_{k,n+g2e-1} = 0$, $g2d_{k,n+g2e-1} = 2$.

if $g2e > 1$,

for $g = n$ to $g = n + g2e - 1$,

{

if $g2b_{ig} = 0$ and $g2b_{jg} = 0$,

then set $g2b_{n+g2e-1,g} = 1$ and $g2d_{n+g2e-1,g} = 1$.

else,

$g2b_{n+g2e-1,g} = 0$ and $g2d_{n+g2e-1,g} = 2$.

}

}

If $g2a_{ij} = 0$, then set $g2b_{ij} = 0$.

Step 5: for i from 1 to $n-1$ and j from $i+1$ to $n-1$,

If $g2a_{ij} = 0$,

For $k = 0$ to $n-1$,

{

if $g2b_{ik} = 1$ and $g2b_{jk} = 1$,

then set $g2d_{ij} = 2$,break;

if $k = n-1$ then set $g2d_{ij} = 3$.

}

Step 6: For i from 0 to $n + g2e - 1$ and j from i to $n + g2e - 1$,

set $g2a_{ii} = 0$,$g2b_{ii} = 0$,$g2d_{ii} = 0$.

$g2a_{ji} = g2a_{ij}$,$g2b_{ji} = g2b_{ij}$,$g2d_{ji} = g2d_{ij}$.

Step 7: For $r = 2$ to $r = n$,

{

for $k = 1$ to ${}^{n+g2e}C_r$

{

Consider each ${}^{n+g2e}C_r$ combinations of vertices of $BG_2(G)$.

Let the k^{th} combination of r vertices be $(v_{\alpha_1}, v_{\alpha_2}, \cdots, v_{\alpha_r})$.

Let $v_{\beta_1}, v_{\beta_2}, \cdots, v_{\beta_{n+g2e-r}}$ be the vertices

outside the current combination.

$g2flag = 0$.

for $i = 1$ to $n + g2e - r$,

{

if $g2flag = 1$,break;

for $j = i + 1$ to $n + g2e - r$.

{

if($g2d_{\beta_{i\alpha_1}}, g2d_{\beta_{i\alpha_2}}, \cdots, g2d_{\beta_{i\alpha_r}}$) =

($g2d_{\beta_{j\alpha_1}}, g2d_{\beta_{j\alpha_2}}, \cdots, g2d_{\beta_{j\alpha_r}}$).

{$g2flag = 1$,break.}

if($j = n + g2e - r$) go to Step 8.

}

}

}

}

Step 8: Print $[g2b_{ij}]$ -Adjacency matrix of $BG_2(G)$.

Print $[g2d_{ij}]$ - Distance matrix of $BG_2(G)$.

Print r -Metric Dimension of $BG_2(G)$.

Print { $v_{\alpha_1}, v_{\alpha_2}, \cdots, v_{\alpha_r}$}-Metric base of $BG_2(G)$.

Step 9: Stop.

Note:

Appendix C contains the complete code for this algorithm.

5.2.2 Computational Analysis of $K_{1,5}$

An output generated by the code included in Appendix C, corresponding to the graph $K_{1,5}$ is presented here.

Example 5.2.1. *Consider the graph* $K_{1,5}$.

The input and output of the above algorithm will be as follows;

Input:

Enter the number of vertices of the simple connected Graph G with more than 2

vertices.

6

Enter the upper triangular part of the adjacency matrix of G excluding the diagonal elements.

1

1

1

1

1

0

0

0

0

0

0

0

0

0

0

Output:

Metric dimension=5.

A metric base for $BG_2(G)$ is,

{ v1, v2, v3, v4, v5 }.

Adjacency matrix of $BG_2(G)$

	V0	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
V0	0	1	1	1	1	1	1	1	1	1	1
V1	1	0	0	0	0	0	1	0	0	0	0
V2	1	0	0	0	0	0	0	1	0	0	0
V3	1	0	0	0	0	0	0	0	1	0	0
V4	1	0	0	0	0	0	0	0	0	1	0
V5	1	0	0	0	0	0	0	0	0	0	1
V6	1	1	0	0	0	0	0	0	0	0	0
V7	1	0	1	0	0	0	0	0	0	0	0
V8	1	0	0	1	0	0	0	0	0	0	0
V9	1	0	0	0	1	0	0	0	0	0	0
V10	1	0	0	0	0	1	0	0	0	0	0

Table 5.3: The adjacency matrix of $BG_2(K_{1,5})$

Distance matrix of $BG_2(G)$

	V0	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
V0	0	1	1	1	1	1	1	1	1	1	1
V1	1	0	2	2	2	2	1	2	2	2	2
V2	1	2	0	2	2	2	2	1	2	2	2
V3	1	2	2	0	2	2	2	2	1	2	2
V4	1	2	2	2	0	2	2	2	2	1	2
V5	1	2	2	2	2	0	2	2	2	2	1
V6	1	1	2	2	2	2	0	2	2	2	2
V7	1	2	1	2	2	2	2	0	2	2	2
V8	1	2	2	1	2	2	2	2	0	2	2
V9	1	2	2	2	1	2	2	2	2	0	2
V10	1	2	2	2	2	1	2	2	2	2	0

Table 5.4: The distance matrix of $BG_2(K_{1,5})$

The two figures, 5.3 and 5.4 validate the tables and other outputs of the algorithm. The graph $K_{1,5}$ and $BG_2(K_{1,5})$ along with metric codes of all vertices are given.

In the figure, Figure 5.4 of $BG_2(K_{1,5})$, the red colored vertices represent the metric base suggested by the algorithm and the metric codes of other vertices are determined based on this vertices.

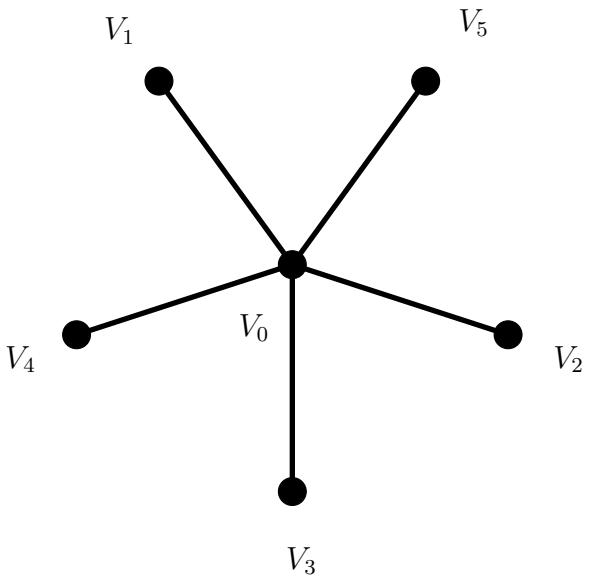


Figure 5.3: The graph $K_{1,5}$

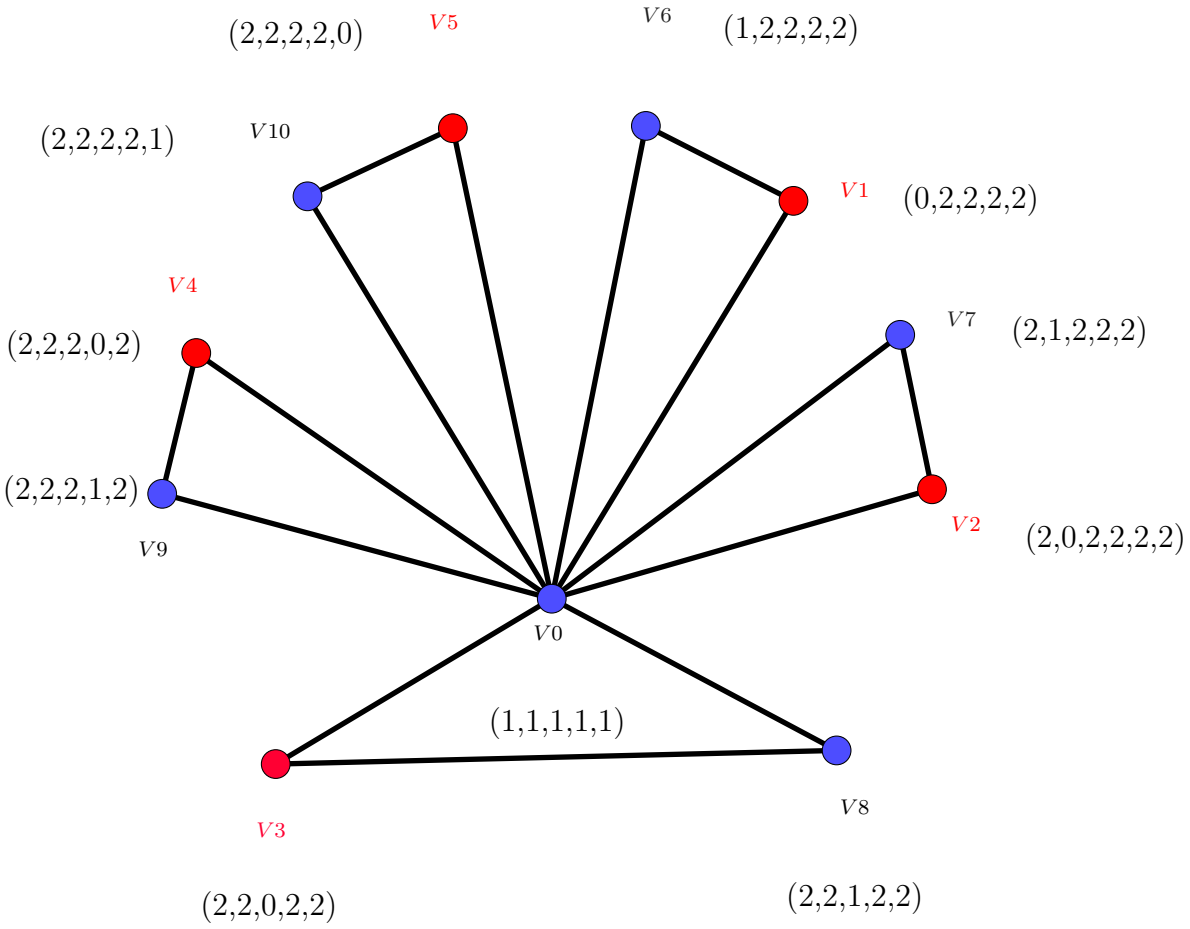


Figure 5.4: The graph $BG_2(K_{1,5})$ with metric codes

5.3 Correctness and Efficiency of Algorithms for $\beta(BG_1(G))$ and $\beta(BG_2(G))$

5.3.1 Correctness of Algorithms

The correctness of an algorithm is established by demonstrating that it consistently produces the correct output for all valid inputs. Technically, there are two types of correctness: (1) **Partial correctness**, which means that if the algorithm terminates, it will give the correct output, and (2) **Total correctness**, which ensures that the algorithm not only produces the correct output but also always terminates. In this section, the algorithms presented are proven to be totally correct.

Step-by-Step Analysis

In this section, the algorithm for the $BG_1(G)$ graph is analyzed.

- **Step 3:** In this step, the number of vertices, n, is received, with the condition $n > 4$. This restriction ensures that the graph G is sufficiently large so that Theorem 2.1.2 for $BG_1(G)$ is applicable.
- **Steps 4-6:** The adjacency matrix of the input graph G is processed. For each entry $g1a_{ij} = 1$, the presence of an edge is detected, and the adjacency of the corresponding line vertex in $BG_1(G)$ is set accordingly. The adjacency in $BG_1(G)$ is set based on the definition 1.3.1. The algorithm also calculates the distance between vertices in $BG_1(G)$ based on Theorem 2.1.2. The symmetry of both the adjacency matrix and distance matrix is used to set these matrices efficiently.
- **Step 7:** The algorithm filters pendant edges by identifying vertices with only one connection to find vertices of $BG_1(G)$ lying at a distance of 3 units,

as per Theorem 2.1.2.

- **Step 8:** The metric dimension r is iteratively calculated starting from $r = 2$ to $r = n$. According to Theorem 3.1.1, since $n > 4$, $BG_1(G)$ cannot be a path, and thus the metric dimension is greater than 1. That is why it has started from $r = 2$. Corollary 3.1.3 guarantees that the metric dimension does not exceed n. For each r-tuple of vertices in $BG_1(G)$, the algorithm checks if the metric codes of vertices outside the r-tuple are identical. If identical codes are found, the algorithm skips the current r-tuple and continues until all r-tuples are exhausted or a valid resolving set is found. Since the search moves from r-tuple to $r + 1$- tuple only if there is no resolving set of order r, thus the first resolving set found by the algorithm will be a resolving set with minimum cardinality, that is, the resolving set will be a metric base. Corollary 3.1.3 guarantees the search for a metric base will not go beyond the set of n-tuples. This ensures the termination of this algorithm after a finite number of steps. this termination and structure of the algorithm guarantees the correct output.

That is the algorithm is totally correct.

A similar set of arguments used to prove the total correctness of the algorithm for $BG_1(G)$ can be applied to the algorithm for $BG_2(G)$. Therefore, we conclude that the algorithm for $BG_2(G)$ is also totally correct.

5.3.2 Time Complexity Analysis

The time complexity of the algorithm for $BG_1(G)$ is determined by the time taken to input the details of G and the number of iterations required to check all possible r-tuples of vertices and verify the uniqueness of the metric codes outside the r-tuple.

Step-by-Step Complexity

- **Step 3:** Reading the number of vertices n takes constant time $O(1)$.

- **Steps 4:** The loops with running variables i and j makes $O(\frac{n(n-1)}{2})$ time. and the inner loop with the running variable has to run for n iterations, so is of order $O(n)$. So the total time complexity of this step is $O(\frac{n(n-1)}{2})$ $O(n)$ $= O(n^3)$.

- **Steps 5:** The two loops make a time complexity $O((g1e-1)^2 - n)$. In worst case, that is when $g1e$ assumes the value $\frac{n(n-1)}{2}$, the time complexity is $O(n^4)$.

- **Steps 6:** The time complexity of the two nested loops is $1+2+...+n+g1e$ $O(\frac{(n+g1e)(n+g1e+1)}{2}) = O(n^4)$ time, as $g1e$ can assume a maximum value $\frac{n(n-1)}{2}$.

- **Step 7:** The first two loops produce a time complexity $O(\frac{n(n-1)}{2})$ and the inner loop with running variable k produces $O(n)$ time complexity. So, in this step the time complexity is $O(\frac{n(n-1)}{2})O(n) = O(n^3)$.

- **Step 8:** The algorithm iterates over combinations of r vertices. The first two loops makes a time complexity,

$$O\left(\sum_{r=2}^{n}\binom{n+g1e}{r}\right) = O(n^{2n})$$

Since,

$$\binom{n}{r} = \frac{n(n-1)\cdots(n-r+1)}{r!}$$

and $g1e$ can assume $\frac{n(n-1)}{2}$ in its worst case.

The next two nested inner loops make the time complexity,

$$O((n + g1e - r - 1) + (n + g1e - r - 2) + \cdots + 2 + 1)$$
$$= O\left(\frac{(n + g1e - r - 1)(n + g1e - r - 1)}{2}\right) = O(n^4),$$

as $g1e$ can assume $\frac{n(n-1)}{2}$ in its worst case. So, the total time complexity is

$$O(n^{2n}) \cdot O(n^4) = O(n^{2n+4}).$$

- **Step 9:** Outputting the adjacency matrix, distance matrix, and metric base takes $O(n^2)$ time, as these are matrix-based outputs and involve iterating over n^2 entries.

Thus the total time complexity of the algorithm is $O(n^{2n+4})$

Hence, it can be concluded that both algorithms are totally correct and have exponential time complexity.

The same approach can be used to determine the time complexity of $BG_2(G)$.

Chapter 6

Algorithmic Complexity of Metric Dimension in $BG_1(\overline{G})$ and $BG_2(\overline{G})$ Graphs

Introduction

Graph theory is a foundational tool for modeling connections and relationships within complex systems. It gives access to various real-world phenomena. In graph theory, the study of the concept of metric dimension provide essential ideas about the structure and navigability of networks. Complement graphs highlight connections between vertices that were not adjacent in the original graph. Algorithms play an important role in graph theory, it gives efficient methods for solving complex problems and helping to draw meaningful information about the networks. By developing and implementing effective algorithms, we can study the theoretical concepts and provide practical solutions for real-world problems. In summary, this chapter focuses on the algorithmic analysis of metric dimension and related measures in $BG_1(\overline{G})$ and $BG_2(\overline{G})$ graphs. By inventing algorithms, it aims to contribute to both theoretical graph theory and practical network analysis.

6.1 Algorithmic Analysis for the Metric Dimension of $BG_1(\overline{G})$

In the algorithm for determining the Metric dimension of the Boolean graph $BG_1(\overline{G})$ associated with a given graph G, it's very important to ensure that the graph is a simple one. The algorithm requires two input parameters: the number of vertices in G (which must exceed 4) and the upper triangular part of the adjacency matrix, denoted as a_{ij}, where $i < j$. The output of the algorithm consists of the adjacency matrix of $BG_1(\overline{G})$, the distance matrix of $BG_1(\overline{G})$, the Metric dimension of $BG_1(\overline{G})$, and a Metric base of $BG_1(\overline{G})$. These outputs give many information about the structural properties of the $BG_1(\overline{G})$, which helps in the study and analysis of this graph structure.

6.1.1 The Sequential Procedure for $\beta(BG_1(\overline{G}))$

The algorithm for calculating the metric dimension and related things of $BG_1(\overline{G})$ is outlined below:

Step 1: Start.

Step 2: Declare two dimensional arrays; $g1ca$, $g1cb$ and $g1cd$.

Declare one dimensional arrays $g1cStack$ and $g1cVstd$.

Declare the integer variables; $g1ce = 0$, n, r, i, j, k, $g1cflag = 0$.

Step 3: Read $n > 4$,the number of vertices in G.

Step 4: For i from 0 to $n-1$ and j from $i+1$ to $n-1$

{ Read $g1ca_{ij}$,

Put $g1ca_{ii} = 0$.

$g1ca_{ij} = 1 - g1ca_{ij}$.

If $g1ca_{ij} = 1$,

{

$g1ce = g1ce + 1$, $g1cd_{ij} = 1.$, $g1cb_{ij} = 1$

$g1cb_{i,n+g1ce-1} = 0$, $g1cb_{j,n+g1ce-1} = 0$,

$g1cd_{i,n+g1ce-1} = 2$, $g1cd_{j,n+g1ce-1} = 2$.

for $k = 0$ to $k = n - 1$,

if k is different from i and j,

then set $g1cb_{k,n+g1ce-1} = 1$, $g1cd_{k,n+g1ce-1} = 1$.

}

If $g1ca_{ij} = 0$, then set $g1cd_{ij} = 2$ and $g1cb_{ij} = 0$.

}

Step 5: For i and j from n to $n + g1ce - 1$,

If $i \neq j$ then Set $g1cb_{ij} = 0$ and $g1cd_{ij} = 2$.

Step 6: For i from 0 to $n + g1ce - 1$ and j from i to $n + g1ce - 1$,

Set $g1cb_{ii} = 0$, $g1cd_{ii} = 0$. $g1ca_{ji} = g1ca_{ij}$, $g1cb_{ji} = g1cb_{ij}$,

$g1cd_{ji} = g1cd_{ij}$.

Step 7: Choose i, $0 \leq i \leq n + g1ce - 1$, set $k = 0$, $g1cStack(k) = i$.

Push i into g1cVstd.

Step 8: If the whole g1cStack is empty, then go to **Step 11**.

Step 9: If (there is j such that $j \notin g1cVstd$ and $g1cb_{g1cStack(k)j} = 1$)

then,

{

k=k+1, push j to $g1cVstd$, $g1cStack(k) = j$,

go to **Step 9**.

}

Step 10: $g1cStack(k) = empty$ (erase the entry),

$k = k - 1$ and go to **Step 8**.

Step 11: If(Number of indices in g1cVstd=n+g1ce-1)

then go to **Step 13**.

else,

print " The graph $BG_1(\overline{G})$ is disconnected and $\beta(BG_1(\overline{G}))$

does not exist" and go to **Step 15**.

Step 12: for(int $i = 0, i < g1cn, i++$)

{

for($intj = i + 1, j < g1cn; j++$)

{

$g1ccs = 0$, $g1crs = 0$

if($a_{i,j} = 1$)

{

$g1cs = g1cs + 1$

for(int $k = 0, k < g1cn, k++$)

{

$g1crs = g1crs + g1ca_{i,k}$, $g1ccs = g1ccs + g1ca_{k,j}$

}

if($g1crs = 1$)

{

$g1cd_{i,g1cn-1+g1cs} = 3$, $g1cd_{g1cn-1+g1cs,i} = 3$.

}

if($g1ccs = 1$)

{

$d_{g1cn-1+g1cs,j} = 3$, $d_{j,g1cn-1+g1cs} = 3$.

}

}

}

}

Step 13: For $r = 2$ to $r = n$,

{

for $k = 1$ to ${}^{n+g1ce}C_r$

{

Consider each ${}^{n+g1ce}C_r$ combinations of vertices of$BG_1(\overline{G})$.

Let the k^{th} combination of r vertices be $(v_{\alpha_1}, v_{\alpha_2}, \cdots, v_{\alpha_r})$

Let $v_{\beta_1}, v_{\beta_2}, \cdots, v_{\beta_{n+g1ce-r}}$ be the vertices

outside the current combination.

$g1cflag = 0$.

for $i = 1$ to $n + g1ce - r - 1$

{

if $g1cflag = 1$,break;

for $j = i + 1$ to $n + g1ce - r$

{

if($g1cd_{\beta_{i\alpha_1}}, g1cd_{\beta_{i\alpha_2}}, \cdots, g1cd_{\beta_{i\alpha_r}}$) =

$(g1cd_{\beta_{j\alpha_1}}, g1cd_{\beta_{j\alpha_2}}, \cdots, g1cd_{\beta_{j\alpha_r}})$

$\{g1cflag = 1$,break$\}$

if($i = n + g1ce - r - 1$ and $j = n + g1ce - r$)

go to **Step 14**.

}

}

}

}

Step 14: Print $[g1cb_{ij}]$ -Adjacency matrix of $BG_1(\overline{G})$.

Print $[g1cd_{ij}]$ - Distance matrix of $BG_1(\overline{G})$.

Print r -Metric Dimension of $BG_1(\overline{G})$.

Print { $v_{\alpha_1}, v_{\alpha_2}, \cdots, v_{\alpha_r}$}-Metric base of $BG_1(\overline{G})$.

Step 15: Stop.

Note:

1. Steps 7 to 10 contain the code for the well-known DFS algorithm, used to check whether a graph is connected [5].

2. The C++ code of this algorithm is included in Appendix D.

6.1.2 Deciphering of the Algorithm for $\beta(BG_1(\overline{G}))$ with P_5

Here, an illustration of the algorithm's functionality for $\beta(BG_1(\overline{G}))$ is provided by applying the code from Appendix D to the graph P_5, along with corresponding pictorial representations.

The figures 6.1, 6.2, and the following outputs corresponding to the given input are illustrations for Theorem 4.1.3.

The red colored vertices of $\beta(BG_1(\overline{P_5}))$ in the figure 6.2 indicates the Metric base.

The algorithm outputs the message, "The $\beta(BG_1(\overline{G}))$ is disconnected", whenever the graph $\beta(BG_1(\overline{G}))$ is found to be disconnected.

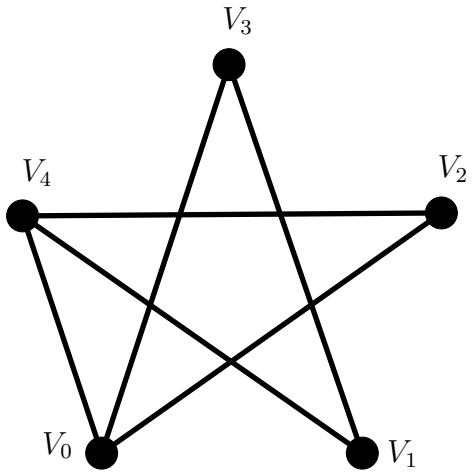


Figure 6.1: The graph $\overline{(P_5)}$

Input

Enter the number of vertices of the simple connected graph G with more than 4 vertices:

5.

Enter the upper triangular part of the adjacency matrix of G excluding the diagonal elements:

1

0

0

0

1

0

0

1

0

1

Output

Adjacency matrix of $BG_1(\overline{G})$

	V0	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
V0	0	0	1	1	1	0	0	0	1	1	1
V1	0	0	0	1	1	1	1	1	0	0	1
V2	1	0	0	0	1	0	1	1	1	1	0
V3	1	1	0	0	0	1	0	1	0	1	1
V4	1	1	1	0	0	1	1	0	1	0	0
V5	0	1	0	1	1	0	0	0	0	0	0
V6	0	1	1	0	1	0	0	0	0	0	0
V7	0	1	1	1	0	0	0	0	0	0	0
V8	1	0	1	0	1	0	0	0	0	0	0
V9	1	0	1	1	0	0	0	0	0	0	0
V10	1	1	0	1	0	0	0	0	0	0	0

Table 6.1: The adjacency matrix $BG_1(\overline{P_5})$

Distance matrix of $BG_1(\overline{G})$

	V0	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
V0	0	2	1	1	1	2	2	2	1	1	1
V1	2	0	2	1	1	1	1	1	2	2	1
V2	1	2	0	2	1	2	1	1	1	1	2
V3	1	1	2	0	2	1	2	1	2	1	1
V4	1	1	1	2	0	1	1	2	1	2	2
V5	2	1	2	1	1	0	2	2	2	2	2
V6	2	1	1	2	1	2	0	2	2	2	2
V7	2	1	1	1	2	2	2	0	2	2	2
V8	1	2	1	2	1	2	2	2	0	2	2
V9	1	2	1	1	2	2	2	2	2	0	2
V10	1	1	2	1	2	2	2	2	2	2	0

Table 6.2: The distance matrix $BG_1(\overline{P_5})$

Metric dimension of $BG_1(\overline{G}) = 4$

A Metric base for $BG_1(\overline{G})$ is { V0, V1, V2, V3 }

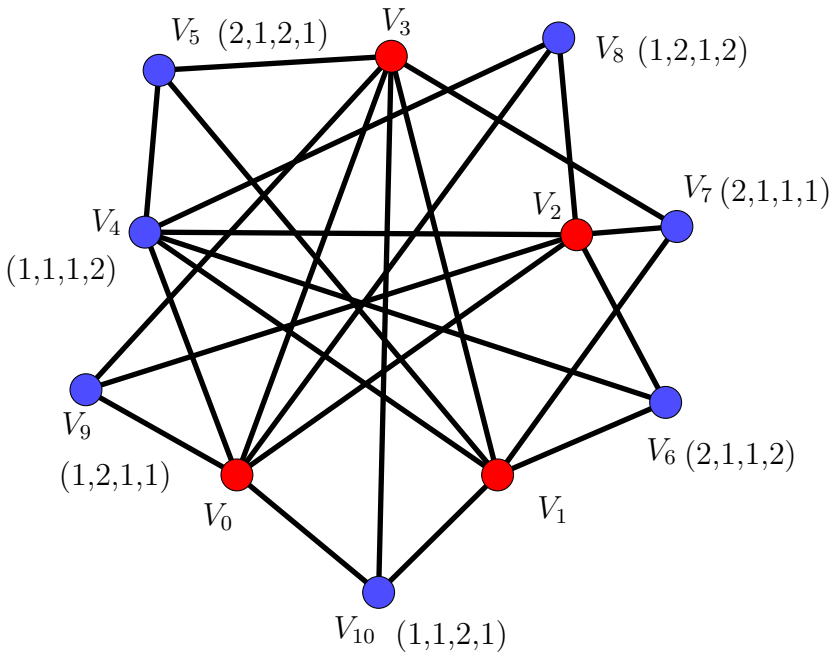


Figure 6.2: The graph $BG_1(\overline{P_5})$

6.2 Metric Dimension of $BG_2(\overline{G})$: The Algorithmic Approach

In the process of determining the Metric dimension of the Boolean graph $BG_2(\overline{G})$ associated with a given graph G, it's essential to ensure the graph is simple. This algorithm needs two input parameters: the vertex count of G (which must exceed 3) and the upper triangular portion of its adjacency matrix, represented as $g2ca_{ij}$, where $i < j$. The resulting output includes the adjacency matrix of $BG_2(\overline{G})$, the distance matrix of $BG_1(\overline{G})$, the Metric dimension of $BG_2(\overline{G})$, and a Metric base of $BG_2(\overline{G})$. These outputs provide a better understanding of the structural attributes of $BG_2(\overline{G})$.

6.2.1 The Systematic Procedure for $\beta(BG_2(\overline{G}))$

The following is the algorithm for $\beta(BG_2(\overline{G}))$. If the graph $BG_2(\overline{G})$ is disconnected, the algorithm outputs a message stating that $BG_2(\overline{G})$ is disconnected.

Step 1 : Start.

Step 2: Declare two dimensional arrays; $g2ca$, $g2cb$ and $g2cd$.

Declare the integer variables; $g2ce = 0$, n, r, i, j, k, $g2cflag = 0$.

Step 3: Read $n > 3$,the number of vertices in G.

Step 4: For $i = 0$ to $n - 1$ and $j = i + 1$ to $n - 1$,

{

Read $g2ca_{ij}$,

Put $g2ca_{ii} = 0$.

$g2ca_{ij} = 1 - g2ca_{ij}$.

If $g2ca_{ij} = 1$,

{

$g2ce = g2ce + 1$, $g2cd_{ij} = 1$, $g2cb_{ij} = 1$,

$g2cb_{i,n+g2ce-1} = 1$, $g2cb_{j,n+g2ce-1} = 1$,

$g2cd_{i,n+g2ce-1} = 1$, $g2cd_{j,n+g2ce-1} = 1$.

for $k = 0$ to $k = n - 1$,

if k is different from i and j,

then set $g2cb_{k,n+g2ce-1} = 0$, $g2cd_{k,n+g2ce-1} = 2$.

for $g = n$ to $g = n + g2ce - 1$,

if $g2cb_{ig} = 0$ and $g2cb_{jg} = 0$,

then set $g2cb_{n+g2ce-1,g} = 1$ and $g2cd_{n+g2ce-1,g} = 1$.

else,

$g2cb_{n+g2ce-1,g} = 0$ and $g2cd_{n+g2ce-1,g} = 2$.

}

If $g2ca_{ij} = 0$, then set $g2cb_{ij} = 0$.

}

Step 5: for i from 1 to $n-1$ and j from $i+1$ to $n-1$,

If $g2ca_{ij} = 0$

For $k = 0$ to $n-1$

{

if $g2cb_{ik} = 1$ and $g2cb_{jk} = 1$,

then set $g2cd_{ij} = 2$,break;

if $k = n-1$ then set $g2cd_{ij} = 3$.

}

Step 6: For i from 0 to $n + g2ce - 1$ and j from i to $n + g2ce - 1$

set $g2ca_{ii} = 0$,$g2cb_{ii} = 0$,$g2cd_{ii} = 0$.

$g2ca_{ji} = g2ca_{ij}$,$g2cb_{ji} = g2cb_{ij}$,$g2cd_{ji} = g2cd_{ij}$.

Step 7: Choose i, $0 \leq i \leq n + g2ce - 1$, set $k = 0$, $g2cStack(k) = i$.

Push i into g2cVstd.

Step 8:If g2cStack is empty goto **Step 11**.

Step 9: If (there is j such that $j \notin g2cVstd$ and $g2cb_{g2cStack(k)j} = 1$)

then,

{

k=k+1, push j to $g2cVstd$, $g2cStack(k) = j$,

goto **Step 9**.

}

Step 10:$g2cStack(k) = empty$, $k = k - 1$ and goto **Step 8**.

Step 11:If(Number of indices in g2cVstd=n+g2ce-1)

then goto **Step 12**.

else,

print " The graph $BG_2(\overline{G})$ is disconnected and $\beta(BG_2(\overline{G}))$

does not exist" and goto **Step 14**.

Step 12: For $r = 2$ to $r = n$,

{

for $k = 1$ to ${}^{n+g2ce}C_r$

{

Consider each ${}^{n+g2ce}C_r$ combinations of vertices of $BG_2(\overline{G})$.

Let the k^{th} combination of r vertices be $(v_{\alpha_1}, v_{\alpha_2}, \cdots, v_{\alpha_r})$

Let $v_{\beta_1}, v_{\beta_2}, \cdots, v_{\beta_{n+g2ce-r}}$ be the vertices

outside the current combination.

$g2cflag = 0$

for $i = 1$ to $n + g2ce - r$

{

if $g2cflag = 1$,break;

for $j = i + 1$ to $n + g2ce - r$

{

if($g2cd_{\beta_{i\alpha_1}}, g2cd_{\beta_{i\alpha_2}}, \cdots, g2cd_{\beta_{i\alpha_r}}$) =

$(g2cd_{\beta_{j\alpha_1}}, g2cd_{\beta_{j\alpha_2}}, \cdots, g2cd_{\beta_{j\alpha_r}})$

$\{g2cflag = 1$,break$\}$

if($j = n + g2ce - r$) go to **Step 13**

}

}

}

}

Step 13: Print $[g2cb_{ij}]$ -Adjacency matrix of $BG_2(\overline{G})$.

Print $[g2cd_{ij}]$ - Distance matrix of $BG_2(\overline{G})$.

Print r -Metric Dimension of $BG_2(\overline{G})$.

Print $\{ v_{\alpha_1}, v_{\alpha_2}, \cdots, v_{\alpha_r}\}$-Metric base of$BG_2(\overline{G})$.

Step 14:Stop.

The well-known Depth-First Search (DFS)[5] method is employed from steps 7 to 11 to check the connectivity of the graph. The algorithm has been efficiently coded in C++ and is included in Appendix E.

6.2.2 Input-Output Analysis with P_4

Let's use the graph P_4 as an example to demonstrate the functionality of the algorithm. The input, output, and diagrammatic representations of $\overline{P_4}$ and $BG_2(\overline{P_4})$ are given for illustration.

Input

Enter the number of vertices of the simple connected graph G with more than 3 vertices:

4.

Enter the upper triangular part of the adjacency matrix of G excluding the diagonal elements:

1

0

0

1

0

1

Adjacency matrix of $BG_2(\overline{G})$

	V0	V1	V2	V3	V4	V5	V6
V0	0	0	1	1	1	1	0
V1	0	0	0	1	0	0	1
V2	1	0	0	0	1	0	0
V3	1	1	0	0	0	1	1
V4	1	0	1	0	0	0	1
V5	1	0	0	1	0	0	0
V6	0	1	0	1	1	0	0

Table 6.3: The adjacency matrix $BG_2(\overline{P_4})$

Distance matrix of $BG_2(\overline{G})$

	V0	V1	V2	V3	V4	V5	V6
V0	0	2	1	1	1	1	2
V1	2	0	3	1	2	2	1
V2	1	3	0	2	1	2	2
V3	1	1	2	0	2	1	1
V4	1	2	1	2	0	2	1
V5	1	2	2	1	2	0	2
V6	2	1	2	1	1	2	0

Table 6.4: The distance matrix $BG_2(\overline{P_4})$

Metric dimension of $BG_2(\overline{G})$= 2.

A Metric base of of $BG_2(\overline{G})$ is { V1, V4 }.

Figures 6.4 and 6.5 exhibit the accuracy of the output.

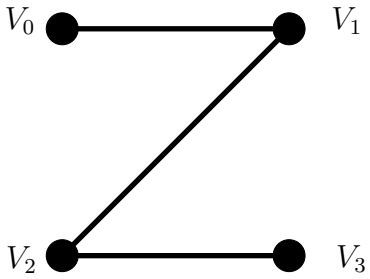


Figure 6.3: The graph P_4

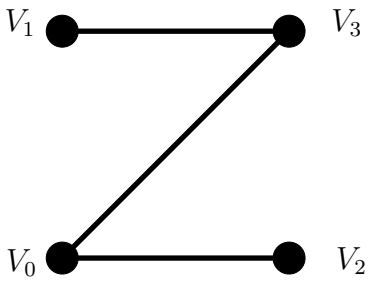


Figure 6.4: The graph $\overline{(P_4)}$

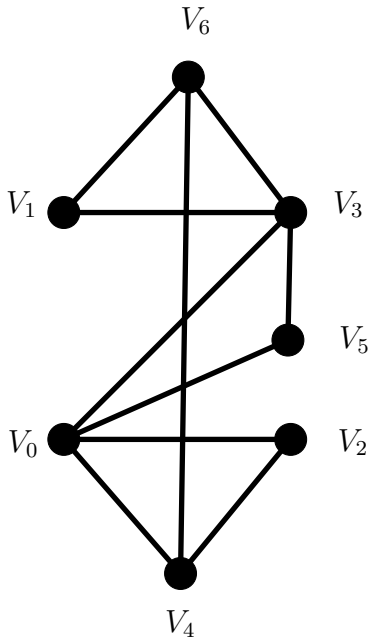


Figure 6.5: The graph $BG_2(\overline{P_4})$

Chapter 7

Real-World Applications for Metric Dimension of $BG_1(G)$ and $BG_2(G)$ Graphs

Introduction

In mathematics, the development of new theories is a major contribution, and their relevance is further enhanced when they can be applied to solve real-world problems. While theoretical contributions are valuable in their own right, finding practical applications helps extend their usage, demonstrating how abstract ideas can lead to innovative solutions in various fields. In this chapter, mainly two real-world problems are discussed, where the $BG_1(G)$ and $BG_2(G)$ graphs, along with their metric dimensions, play major roles in finding solutions. These applications demonstrate the importance and relevance of the theories developed in the previous chapters.

7.1 Optimizing Robot Navigation with $\beta(BG_2(G))$

Imagine a robot navigating a plane surface as shown in figure 7.1, tasked with moving along a specially designed path to complete a specific mission. As the robot

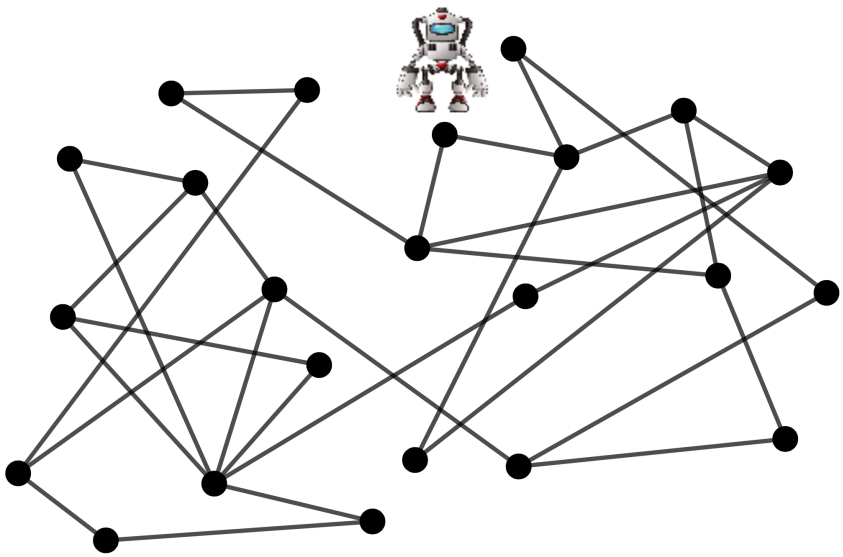


Figure 7.1: Track for robot movement

moves, it relies on signals from various *signal receiving centers*, represented by the vertices, placed strategically across the track. These centers help to identify the robot's position by calculating its distance from them to *signal processing centers*. It is some of these signal receiving centers work as signal processing centers also, this signal processing centers process the robot's location data and is monitoring the robot's movement.

Now, here's the challenge: the cost of setting up a signal processing center is comparatively high, so it is a need to minimize their number while still maintaining accurate robot tracking. Let's take a look at how this can be solved.

The Scenario: A Simple Track with Signal Centers

Consider a simple track where five signal receiving centers, as shown in figure 7.2, are placed along the track. The task at hand is to determine the minimum number of signal processing centers required to pinpoint the robot's position accurately and where to place them.

Step 1: Can One Signal Processing Center Work?

We start by considering the case of using just a single signal processing center in figure 7.2, say at point A. If A is the only signal processing center, the distances from A to points B and D will be identical (one unit away), and hence the robot's position cannot be accurately determined, as both positions would have the same distance "code." The same logic works for the other signal receiving centers, which means that using just one signal processing center fails to identify the robot's position correctly.

Step 2: Trying Two Signal Processing Centers

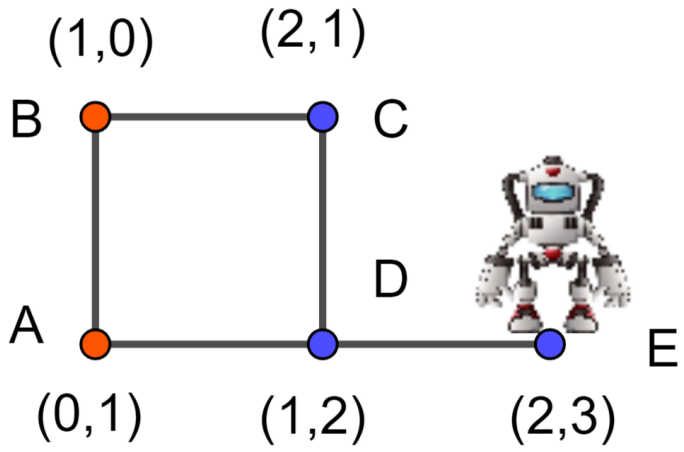


Figure 7.2: Track for robot movement

Next, try placing two signal processing centers, for example at points A and C. If we do this, the distances from points B and D to A and C will still result in identical distance codes, (1,1), making it impossible to distinguish between the two locations. However, if we place signal processing centers at points A and B instead, the code for each signal receiving center becomes unique, refer to figure 7.2. This means the robot's position can now be identified correctly.

This leads to two important questions on a general network:

1. What is the minimum number of signal processing centers needed to accurately track the robot?.
2. Where should these signal processing centers be placed?.

The answer to the first question is in fact the *metric dimension* of the network, while the answer to the second question is a *metric base* of the network.

Expanding the Network for Efficiency: The $BG_2(G)$ Expansion

In real-world scenarios, situations may arise where the signal receiving centers fail to accurately detect the robot due to various factors like long distances between signal receiving centers or poor weather conditions. To overcome this, we can strengthen the network by adding more signal receiving centers.

One of the most effective ways to expand the network is through what we call the $BG_2(G)$ expansion. Here's how it works: Introduce a new signal receiving center near the middle of the track (edge) between the existing centers, and connect it to its parent centers. If any pair of newly formed signal receiving centers do not share a common parent, connect them to ensure the network becomes more efficient.

This $BG_2(G)$ expansion totally alters the network by increasing the number of signal receiving centers and modifying the structure. With this change, we again face the two key questions:

1. What is the new minimum number of signal processing centers needed for accurate tracking in the expanded network?

2. Where should these signal processing centers be placed in the new network?

Solving the Problem with Algorithms

The answers to these fundamental questions can be easily found using the *second algorithm from Chapter 5*. By inputting the number of signal receiving centers

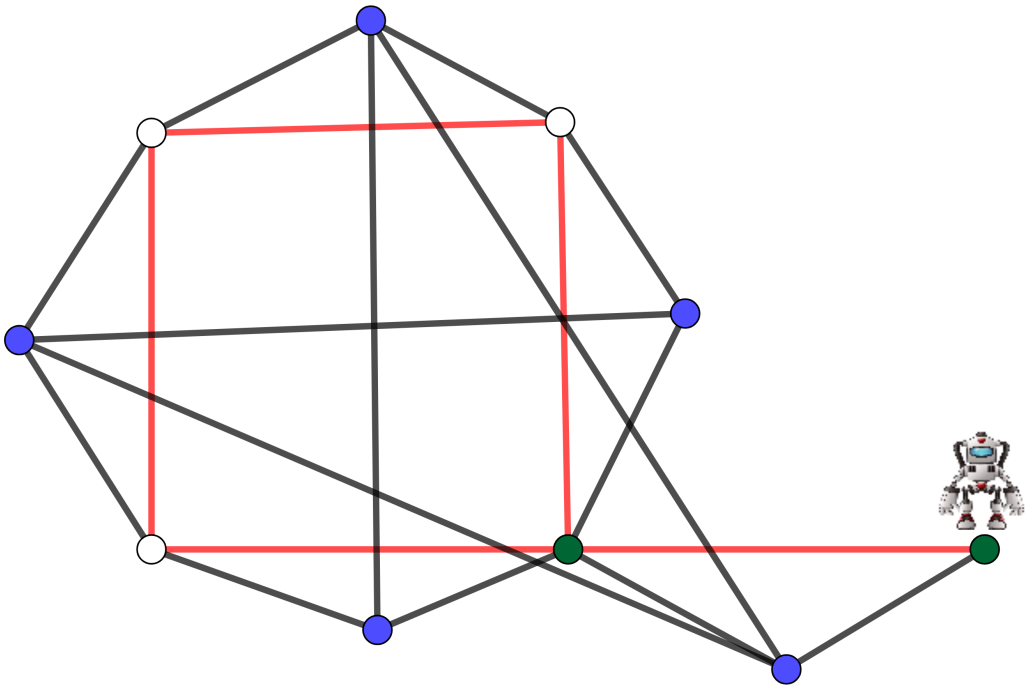


Figure 7.3: Robot On Optimized Network

and the connections between them in the original track, the algorithm expands the network into its BG_2 form. It then calculates the **metric dimension** and the **metric base** of the new network.

In case of the simple track represented by the figure 7.2, output of the algorithm has the following recommendations about the corresponding expanded network shown in figure 7.3:

- It needs 3 number of signal processing centers required for the new, optimized network.
- The optimal locations for placing these centers are respectively in the positions of A, B, and C in the old network(they are shown in white color in figure 7.3), to ensure accurate tracking of the robot.

By applying this approach, we can design an efficient, cost-effective network that ensures reliable robot navigation for any number of vertices.

7.2 Application of $BG_1(G)$, Expansion in Collaboration Network

Consider a scenario where a group of companies collaborates to manufacture an innovative product, a high-tech drone. Each part of the drone including its navigation system, propulsion, camera module, and communication system is produced by a subset of these companies working together. For example, Company A may collaborate with Company B on the navigation system, while Company B and Company C work together on the camera module. This collaboration can be represented as a network where companies are vertices and their collaborations are edges.

At the final integration stage, the complexity increases as each company now needs to ensure that its component seamlessly integrates with the others. For this purpose, every pair of companies working together on any part of the product must establish a direct communication channel to coordinate with other

companies. This requirement gives rise to an expanded network, where these direct communication wings between collaborating companies form new connections to all other companies.

This structure is an expansion of the original collaboration network G, which ensures that the communication among different wings is efficient and comprehensive. Each new wing, formed by a pair of collaborating companies, must now interact with the rest of the companies to ensure the successful integration of their components into the final product. This expanded structure is $BG_1(G)$ of the original collaboration network G.

The $BG_1(G)$ expansion introduces a more complex network where efficient communication is key to successful product integration. However, the challenge lies in determining the optimal number of communication centers, or nodal centers, needed to minimize cost while ensuring every wing can communicate with the other companies efficiently.

Here is where the first algorithm from Chapter 5 becomes crucial. The algorithm calculates the metric dimension and the metric base of the $BG_1(G)$. The metric dimension provides the minimum number of nodal centers required to uniquely identify and facilitate communication between the different wings in the network. By placing these nodal centers at the positions determined by the metric base, the companies can ensure that their communication system is not only cost-effective but also the final integration stage is smooth.

This process improves the overall efficiency of the collaboration. It demonstrates the practical application of $BG_1(G)$ expansion and metric dimension theory in solving real-world challenges in collaborative manufacturing.

7.3 Applications in Other Domains

In addition to the applications discussed in this chapter, the theories developed for $BG_1(G)$ and $BG_2(G)$ graphs and their metric dimensions have a wide range of applications in other fields. For instance, in transportation networks, optimizing routes and minimizing costs can benefit from the metric dimension theory to efficiently place monitoring or control stations. Similarly, Furthermore, in social networks, the application of these theories can help analyze communication patterns and optimize information flow across various connected groups. These diverse applications highlight the broad utility and relevance of the theoretical contributions made in this study.

Chapter 8

Conclusion

The study focuses on the structural properties and metric dimensions of the expanded graphs $BG_1(G)$ and $BG_2(G)$, it gives both theoretical contributions and practical applications. The study began by establishing foundational concepts and searching for the basic properties of these graphs, which needed to understand their behavior and relevance.

Substantial attention in this work is dedicated to developing theories related to the metric dimensions of $BG_1(G)$ and $BG_2(G)$, leading to several findings. The theorems about the metric dimension of the Boolean graphs, especially the one, showing that it will not exceed the number of vertices of G, along with the study of the distances between vertices, as highlighted by various theorems, played the main role in shaping the algorithms for determining the metric dimension. Theorems were introduced that established the exact metric dimensions when the graph G is a path, complete graph, star graph, or cycle. These precise results not only refined the theoretical study of metric dimension but also gave a strong base for developing algorithms. The thesis also contributes to the field of metric dimension by proving new results related to $BG_1(\overline{G})$ and $BG_2(\overline{G})$. These findings improve the understanding of the properties and metric dimensions of these expanded graphs.

The algorithms proposed in this work contribute to the broader field of graph

theory, which gives tools that can address real-world problems. Their application extends beyond academic interest, particularly in areas such as robotics and collaboration networks, where optimizing navigation and communication is critical.

However, certain limitations were encountered, particularly concerning the computational complexity of the algorithms for large graphs. Future research could focus on optimizing these algorithms by searching for polynomial-time algorithms or expanding their applicability to different types of graphs.

In summary, this thesis makes a considerable addition to the field by presenting the theoretical ideas about $BG_1(G)$ and $BG_2(G)$, while also providing practical tools and algorithms for determining their metric dimensions. These findings open numerous possibilities for future research and application.

Chapter 9

Recommendations

This thesis has added many ideas to the study of the metric dimension in Boolean graphs $BG_1(G)$ and $BG_2(G)$, providing theoretical results and practical algorithms. However, there are several directions in which future research can build upon these findings to further enhance the understanding of the metric dimension in graph theory.

- **Exploring Metric Dimensions of Other Boolean Graphs:** While this study focused primarily on $BG_1(G)$ and $BG_2(G)$, there is a scope to study the metric dimension of other Boolean graphs derived from a graph G by redefining the adjacency in different ways. Examining whether similar bounds or exact values for the metric dimension hold in other Boolean graphs would be a valuable direction for future work.

- **Optimizing the Existing Algorithms:** The algorithms presented in this thesis for determining the metric dimension of $BG_1(G)$ and $BG_2(G)$ are efficient, but there is room for improvement. One potential area of research is optimizing these algorithms to reduce their time complexity, especially for large graphs. Such innovations could improve the practical applicability of the algorithms, making them more suitable for real-time or large-scale applications.

- **Expanding Applications to New Domains:** The theoretical advancements in this work, particularly in determining the metric dimension of Boolean graphs, have applications in robotics, collaboration networks, and communication systems. However, there is scope for these findings to be applied to other domains as well. Areas such as network security, transportation logistics, and sensor networks could benefit from the efficient navigation and location tracking that metric dimension algorithms offer. Future studies should explore these domains to identify the most suitable applications for the theoretical advancements made in this thesis.

In conclusion, future research could focus on extending the theoretical contributions to other types of Boolean graphs, optimizing the existing algorithms, and exploring new and diverse application areas to further enhance the relevance and impact of the results presented in this thesis.

Appendix

A List of Research Papers

1. Sameerali C.P. and Sameena K. "On Metric Dimension of Boolean Graph $BG_1(G)$." *South East Asian Journal of Mathematics and Mathematical Sciences*, Vol. 18, No. 3 (2022), pp. 329-338. https://doi.org/10.56827/SEAJMMS.2022.1803.27 . ISSN (Online): 2582-0850, ISSN (Print): 0972-7752.

2. Sameerali C.P. and Sameena K. "Metric Dimension of$BG_2(G)$ in an Algorithmic Aspect." *Indian Journal of Natural Sciences*, Vol. 14, Issue 80, Oct 2023, International Bimonthly (Print) – Open Access. ISSN: 0976–0997.

3. C.P. Sameerali and K. Sameena. "On Properties of Boolean Graph $BG_2(G)$." *Advances in Mathematics: Scientific Journal*, 9 (2020), no. 4, 1845–1850. ISSN: 1857-8365 (printed); 1857-8438 (electronic). https://doi.org/10.37418/amsj.9.4.41 Spec. Issue on NCFCTA-2020.

4. Sameerali C.P. and Sameena K. "Boolean Graphs of Complement: A Metric Dimension Approach." *Asian European Journal of Mathematics (AEJM)*, Special Issue on Algebra and Discrete Mathematics (ICADM 2024),(Communicated).

5. Sameerali C.P. and Sameena K. "On the Composition and Powers of Boolean Graphs",(Communicated).

6. Sameerali C.P. and Sameena K. "Metric Dimension for Boolean Graphs of Fundamental Graph Familes",(Communicated).

7. Sameerali C.P. and Sameena K. "Metric Dimension Algorithm for Boolean Graph: Bridging Theory and Real-World Applications",(Communicated).

B C++ Realization of the Algorithm for $\beta(BG_1(G))$

```
#include <iostream>
using namespace std;
// merge the following 3 lines
int g1generateCombination(int g1f[],
int g1start,int g1end,int g1r,int g1combArr[],
int g1combArrInd, int g1e,int g1d[50][50])
{
    int i, j, k, status = 0, g1flag;
    if (g1combArrInd == g1r)
    {
        g1flag = 0;
        for (i = 0; i < g1end + 1 && g1flag != 1; i++)
        {
            for (j = i + 1; j < g1end + 1 && g1flag != 1; j++)
            {
                for (k = 0; k < g1r; k++)
                {
                        // merge the following 2 lines
```

```
                if (g1d[i][g1combArr[k]]!=
                    g1d[j][g1combArr[k]])
                    break;
                if (k == g1r - 1)
                {
                    g1flag = 1;
                }
            }
        }
        if (i == g1end)
        {
            // merge the following 3 lines
            cout << "\n\n metric dimension=" <<
             g1r <<"\n\n A Metric base is\n {";
            for (i = 0; i < g1r; i++)
                cout << " v" << g1combArr[i] << ",";
            cout << "\b }\n\n";
            return 1;
        }
    }
    return 0;
}
// merge the following 2 lines
for (i = g1start; i <= g1end && g1end - i + 1 >=
g1r - g1combArrInd && status == 0; i++)
{
    g1combArr[g1combArrInd] = g1f[i];
        // merge the following 2 lines
    status = g1generateCombination(g1f, i + 1,g1end,
```

```
            g1r, g1combArr,g1combArrInd + 1, g1e, g1d);
    }
    return status;
}
 // merge the following 2 lines
int g1nextSetCombns(int g1f[], int g1n,
        int g1r, int g1e,int g1d[50][50])
{ // merge the following 2 lines
    int tempArr[g1r];
    return g1generateCombination(g1f, 0,
    g1n + g1e - 1, g1r,tempArr, 0, g1e, g1d);
}

int main()
{
    int g1b[50][50], g1d[50][50];
    int g1e = 0, g1n, g1r, i, g1flag = 0,g1cs,g1rs,g1s=0;
     // merge the following 2 lines
    printf("Enter the number of vertices  of the simple
        connected Graph G with more than 4 vertices");
    cin >> g1n;
    int g1a[g1n][g1n];
     // merge the following 3 lines
    cout << "Enter the upper triangular part of the
     adjacency matrix of G excluding the diagonal
     elements " << endl;
    for (int i = 0; i < g1n; i++)
    {
        g1a[i][i] = 0;
```

```
g1b[i][i] = 0;
g1d[i][i] = 0;
for (int j = i + 1; j < g1n; j++)
{
    cin >> g1a[i][j];
    g1a[j][i] = g1a[i][j];
    g1b[i][j] = g1a[i][j];
    g1b[j][i] = g1b[i][j];
    if (g1a[i][j] == 1)
    {
        g1e = g1e + 1;
        g1d[i][j] = 1;
        g1d[j][i] = 1;
        g1b[j][g1n + g1e - 1] = 0;
        g1b[i][g1n + g1e - 1] = 0;
        g1b[g1n + g1e - 1][j] = 0;
        g1b[g1n + g1e - 1][i] = 0;
        g1d[j][g1n + g1e - 1] = 2;
        g1d[i][g1n + g1e - 1] = 2;
        g1d[g1n + g1e - 1][j] = 2;
        g1d[g1n + g1e - 1][i] = 2;
        for (int k = 0; k < g1n; k++)
        {
            if (k != i && k != j)
            {
                g1b[k][g1n + g1e - 1] = 1;
                g1b[g1n + g1e - 1][k] = 1;
                g1d[k][g1n + g1e - 1] = 1;
                g1d[g1n + g1e - 1][k] = 1;
```

```
                    }
                }
            }
            if (g1a[i][j] == 0)
            {
                g1d[i][j] = 2;
                g1d[j][i] = 2;
            }
        }
    }
    for (int i = g1n; i < g1n + g1e; i++)
    {
        for (int j = g1n; j < g1n + g1e; j++)
        {
            g1b[i][j] = 0;
            if (i == j)
            {
                g1d[i][j] = 0;
            }
            if (i != j)
            {
                g1d[i][j] = 2;
                g1d[j][i] = 2;
            }
        }
    }
for(int i=0; i<g1n; i++)
   {
```

```
        for(int  j=i+1;j<g1n;j++)
        {
             g1cs=0;
             g1rs=0;

          if(g1a[i][j]==1)
             {
                g1s=g1s+1;
                for(int  k=0;k<g1n;k++)
                {
                     g1rs=g1rs+g1a[i][k];
                     g1cs=g1cs+g1a[k][j];
                }
                  if(g1rs==1)
                    {
                    g1d[i][g1n-1+g1s]=3;
                    g1d[g1n-1+g1s][i]=3;
                    }
                    if(g1cs==1)
                    {
                    g1d[g1n-1+g1s][j]=3;
                    g1d[j][g1n-1+g1s]=3;
                    }

             }
        }


}
```

```
cout << "Adjacency matrix of BG1(G)" << endl<< endl;
for (int i = 0; i < g1n + g1e; i++)
    cout << "\t"<< "V" << i;
cout << endl;
for (int i = 0; i < g1n + g1e; i++)
{
    cout << "V" << i;
    for (int j = 0; j < g1n + g1e; j++)
    {
        cout << "\t" << g1b[i][j];
    }
    cout << endl;
}
 // merge the following 2 lines
cout << endl<< "Distace matrix of BG1(G)" <<
 endl<< endl;
for (int i = 0; i < g1n + g1e; i++)
    cout << "\t" << "V" << i;
cout << endl;
for (int i = 0; i < g1n + g1e; i++)
{
    cout << "V" << i;
    for (int j = 0; j < g1n + g1e; j++)
    {
        cout << "\t" << g1d[i][j];
    }
    cout << endl;
}
int g1f[g1n + g1e];
```

```
    for (i = 0; i < g1n + g1e; i++)
        g1f[i] = i;
    for (g1r = 2; g1r <= g1n && g1flag == 0; g1r++)
    {
        g1flag = g1nextSetCombns(g1f, g1n, g1r, g1e, g1d);
    }
    return 0;
}
```

C C++ Implementation of the Algorithm for $\beta(BG_2(G))$

```
#include <iostream>
using namespace std;
//Merge the following 3 lines
int g2generateCombination(int g2f[], int g2start,
int g2end, int g2r, int g2combArr[],
int g2combArrInd, int g2e, int g2d[50][50])
{
    int i, j, m, k, status = 0, g2flag;
    if (g2combArrInd == g2r)
    {
        g2flag = 0;
    for (i = 0; i < g2end + 1 && g2flag != 1; i++)
        {
            //Merge the following 2 lines
            for (j = i + 1; j < g2end + 1 &&
            g2flag != 1; j++)
            {
```

```
                for (k = 0; k < g2r; k++)
                {
                    if (g2d[i][g2combArr[k]] !=
                        g2d[j][g2combArr[k]])
                        break;
                    if (k == g2r - 1)
                    {
                        g2flag = 1;
                    }
                }
            }
            if (i == g2end)
            {
                cout << "\n\n metric dimension=" << g2r
                << "\n\n A resolving set is\n {";
                for (i = 0; i < g2r; i++)
                    cout << " v" << g2combArr[i] << ",";
                cout << "\b }\n\n";
                return 1;
            }
        }
        return 0;
    }
    for (i = g2start; i <= g2end && g2end - i + 1 >=
     g2r - g2combArrInd && status == 0; i++)
    {
        g2combArr[g2combArrInd] = g2f[i];
        status = g2generateCombination(g2f, i + 1,
    g2end, g2r, g2combArr, g2combArrInd + 1, g2e, g2d);
```

```
    }
    return status;
}

int g2nextSetCombns(int f[], int g2n, int g2r,
                    int g2e, int g2d[50][50])
{
    int g2tempArr[g2r];
    return g2generateCombination(f, 0, g2n + g2e - 1,
                         g2r, g2tempArr, 0, g2e, g2d);
}

int main()
{
    int g2b[50][50], g2d[50][50];
    int g2e = 0, g2n, g2r, i, g2flag = 0, g2u, g2w;
    //Merge the following 2 lines
    cout << "Enter the number of vertices of the simple
     connected Graph G with more than 2 vertices: ";
    cin >> g2n;
    int g2a[g2n][g2n];
    //Merge the following 3 lines
    cout << "Enter the upper triangular part of the
    adjacency matrix of G excluding the diagonal
    elements: " << endl;
    for (int i = 0; i < g2n; i++)
    {
        g2a[i][i] = 0;
        g2b[i][i] = 0;
```

```
g2d[i][i] = 0;
for (int j = i + 1; j < g2n; j++)
{
    cin >> g2a[i][j];
    g2a[j][i] = g2a[i][j];
    g2b[i][j] = g2a[i][j];
    g2b[j][i] = g2b[i][j];
    if (g2a[i][j] == 1)
    {
        g2e++;
        g2d[i][j] = 1;
        g2d[j][i] = 1;
        g2b[j][g2n + g2e - 1] = 1;
        g2b[i][g2n + g2e - 1] = 1;
        g2b[g2n + g2e - 1][j] = 1;
        g2b[g2n + g2e - 1][i] = 1;
        g2d[j][g2n + g2e - 1] = 1;
        g2d[i][g2n + g2e - 1] = 1;
        g2d[g2n + g2e - 1][j] = 1;
        g2d[g2n + g2e - 1][i] = 1;
        g2b[g2n + g2e - 1][g2n + g2e - 1] = 0;
        for (int k = 0; k < g2n; k++)
        {
            if (k != i && k != j)
            {
                g2b[k][g2n + g2e - 1] = 0;
                g2b[g2n + g2e - 1][k] = 0;
                g2d[k][g2n + g2e - 1] = 2;
                g2d[g2n + g2e - 1][k] = 2;
```

```
            }
        }
        g2u = i;
        g2w = j;
        if (g2e > 1)
        {
            //Merge the following 2 lines
            for (int g = g2n; g <
            g2n + g2e - 1; g++)
            {
                //Merge the following 2 lines
                if (g2b[i][g] == 0
                    && g2b[j][g] == 0)
                {
                    g2b[g2n + g2e - 1][g] = 1;
                    g2b[g][g2n + g2e - 1] = 1;
                    g2d[g2n + g2e - 1][g] = 1;
                    g2d[g][g2n + g2e - 1] = 1;
                }
                else
                {
                    g2b[g2n + g2e - 1][g] = 0;
                    g2b[g][g2n + g2e - 1] = 0;
                    g2d[g2n + g2e - 1][g] = 2;
                    g2d[g][g2n + g2e - 1] = 2;
                }
            }
        }
    }
```

```
            if (g2a[i][j] == 0)
            {
                g2d[i][j] = 0;
                g2d[j][i] = 0;
            }
        }
    }
    for (int i = 0; i < g2n; i++)
    {
        for (int j = i + 1; j < g2n; j++)
        {
            if (g2a[i][j] == 0)
            {
                for (int k = 0; k < g2n; k++)
                {
        if (g2b[i][k] == 1 && g2b[j][k] == 1)
                    {
                        g2d[i][j] = 2;
                        g2d[j][i] = 2;
                        break;
                    }
                    if (k == g2n - 1)
                    {
                        g2d[i][j] = 3;
                        g2d[j][i] = 3;
                    }
                }
            }
        }
```

```
    }
cout << "Adjacency matrix of BG2(G)" << endl<< endl;
    for (int i = 0; i < g2n + g2e; i++)
        cout << "\t"<< "V" << i;
        cout << endl;
    for (int i = 0; i < g2n + g2e; i++)
    {
        cout << "V" << i;
        for (int j = 0; j < g2n + g2e; j++)
        {
            cout << "\t" << g2b[i][j];
        }
        cout << endl;
    }
    //Merge the following 2 lines
    cout << endl
    << "Distance matrix of BG2(G)" << endl << endl;
    for (int i = 0; i < g2n + g2e; i++)
        cout << "\t" << "V" << i;
    cout << endl;
    for (int i = 0; i < g2n + g2e; i++)
    {
        cout << "V" << i;
        for (int j = 0; j < g2n + g2e; j++)
        {
            cout << "\t" << g2d[i][j];
        }
        cout << endl;
    }
```

```
    int f[g2n + g2e];
    for (int i = 0; i < g2n + g2e; i++)
        f[i] = i;

    for (g2r = 2; g2r <= g2n && g2flag == 0; g2r++)
    {
        g2flag = g2nextSetCombns(f, g2n, g2r, g2e, g2d);
    }
    return 0;
}
```

D Coding of the Algorithm for $\beta(BG_1(\overline{G}))$

Based on the algorithm presented in last Section 6.1.1, a C++ code is generated and it is given below.

```
#include <iostream>
using namespace std;
//merge the following 2 lines
void g1cdfs(int g1cvertex, const int g1cgraph[][50],
        int g1cnumvertices, bool g1cvisited[])
{
    g1cvisited[g1cvertex] = true;

    for (int i = 0; i < g1cnumvertices; ++i)
    {
        if (g1cgraph[g1cvertex][i] && !g1cvisited[i])
        {
    g1cdfs(i, g1cgraph, g1cnumvertices, g1cvisited);
```

```
        }
    }
}
//merge the following 2 lines
bool g1cconnect(int g1cnumvertices,
                        const int g1cgraph[][50])
{
    bool g1cvisited[50] = {false};
    g1cdfs(0, g1cgraph, g1cnumvertices, g1cvisited);

    for (int i = 0; i < g1cnumvertices; ++i)
    {
        if (!g1cvisited[i])
        {
            return false;  // Graph is not connected
        }
    }

    return true;  // Graph is connected
}
//merge the following 3 lines
int g1cgenerateCombination(int f[], int start, int end,
int r, int g1ccombArr[], int g1ccombArrInd, int g1ce,
 int g1cd[50][50])
{
  int i, j, m, k, status = 0, g1cflag;
  if (g1ccombArrInd == r)
    {
    g1cflag = 0;
```

```
 for (i = 0; i < end + 1 && g1cflag != 1; i++)
 {
     for (j = i + 1; j < end + 1 && g1cflag != 1; j++)
      {
         for (k = 0; k < r; k++)
          {
              //merge the following 2 lines
             if (g1cd[i][g1ccombArr[k]] !=
                 g1cd[j][g1ccombArr[k]])
                 break;
             if (k == r - 1)
             {
                 g1cflag = 1;
             }
         }
     }
     if (i == end)
      {//merge the following 2 lines
         cout << "\n\n metric dimension=" << r
          << "\n\n A Metric base is\n {";
         for (i = 0; i < r; i++)
             cout << " v" << g1ccombArr[i] << ",";
         cout << "\b }\n\n";
         return 1;
     }
 }
 return 0;
}
//merge the following 2 lines
```

```
  for (i = start; i <= end && end - i + 1 >=
  r - g1ccombArrInd && status == 0; i++)
  {
    g1ccombArr[g1ccombArrInd] = f[i];
        //merge the following 2 lines
    status = g1cgenerateCombination(f, i + 1, end, r,
    g1ccombArr, g1ccombArrInd + 1, g1ce, g1cd);
  }
  return status;
}
//merge the following 2 lines
int g1cnextSetCombns(int f[], int n,
int r, int g1ce, int g1cd[50][50])
{
    int tempArr[r];
    //merge the following 2 lines
    return g1cgenerateCombination(f, 0,
n + g1ce - 1, r, tempArr, 0, g1ce, g1cd);
}

int main()
{
    int g1cb[50][50], g1cd[50][50];
    int g1ce = 0, n, r, i, g1cflag = 0, g1cs = 0;
    //merge the following 2 lines
    printf("Enter the number of vertices  of the
simple connected Graph G with more than 4 vertices");
    cin >> n;
    int g1ca[n][n];
```

```
//merge the following 3 lines
cout << "Enter the upper triangular part of the
adjacency matrix of G excluding the
diagonal elements " << endl;
for (int i = 0; i < n; i++)
    {
    g1ca[i][i] = 0;
    g1cb[i][i] = 0;
    g1cd[i][i] = 0;
    for (int j = i + 1; j < n; j++)
    {
        cin >> g1ca[i][j];
        g1ca[i][j] = 1 - g1ca[i][j];
        g1ca[j][i] = g1ca[i][j];
        g1cb[i][j] = g1ca[i][j];
        g1cb[j][i] = g1cb[i][j];
        if (g1ca[i][j] == 1)
        {
            g1ce = g1ce + 1;
            g1cd[i][j] = 1;
            g1cd[j][i] = 1;
            g1cb[j][n + g1ce - 1] = 0;
            g1cb[i][n + g1ce - 1] = 0;
            g1cb[n + g1ce - 1][j] = 0;
            g1cb[n + g1ce - 1][i] = 0;
            g1cd[j][n + g1ce - 1] = 2;
            g1cd[i][n + g1ce - 1] = 2;
            g1cd[n + g1ce - 1][j] = 2;
            g1cd[n + g1ce - 1][i] = 2;
```

```
                for (int k = 0; k < n; k++)
                {
                    if (k != i && k != j)
                    {
                        g1cb[k][n + g1ce - 1] = 1;
                        g1cb[n + g1ce - 1][k] = 1;
                        g1cd[k][n + g1ce - 1] = 1;
                        g1cd[n + g1ce - 1][k] = 1;
                    }
                }
            }
            if (g1ca[i][j] == 0)
            {
                g1cd[i][j] = 2;
                g1cd[j][i] = 2;
            }
        }
    }
    for (int i = n; i < n + g1ce; i++)
     {
        for (int j = n; j < n + g1ce; j++)
        {
            g1cb[i][j] = 0;
            if (i == j)
            {
                g1cd[i][j] = 0;
            }
            if (i != j)
            {
```

```
                g1cd[i][j] = 2;
                g1cd[j][i] = 2;
            }
        }
    }

    for(int i = 0; i < n; i++)
    {
    for(int j = i + 1; j < n; j++)
     {
        int g1ccs = 0;
        int g1crs = 0;

        if(g1ca[i][j] == 1)
        {
            g1cs = g1cs + 1;
            for(int k = 0; k < n; k++)
              {
                g1crs = g1crs + g1ca[i][k];
                g1ccs = g1ccs + g1ca[k][j];
              }
            if(g1crs == 1)
              {
                g1cd[i][n - 1 + g1cs] = 3;
                g1cd[n - 1 + g1cs][i] = 3;
              }
            if(g1ccs ==1)
             {
                g1cd[n - 1 + g1cs][j] = 3;
```

```
                g1cd[j][n - 1 + g1cs] = 3;
            }
        }
      }
    }

    bool isBConnected = g1cconnect(n + g1ce, g1cb);

    if (!isBConnected)
    {
  cout << "The BG1(G Gcomp) disconnected." << endl;
        return 0;
    }
//merge the following 2 lines
    cout << "Adjacency matrix of
     BG1(G Gcomp)" << endl << endl;

    for (int i = 0; i < n + g1ce; i++)
        cout << "\t" << "V" << i;
    cout << endl;
    for (int i = 0; i < n + g1ce; i++)
    {
        cout << "V" << i;
        for (int j = 0; j < n + g1ce; j++)
        {
            cout << "\t" << g1cb[i][j];
        }
        cout << endl;
    }
```

```
    //merge the following 2 lines
    cout << endl << "Distance matrix
     of BG1(G Comp)" << endl << endl;
    for (int i = 0; i < n + g1ce; i++)
        cout << "\t" << "V" << i;
    cout << endl;
    for (int i = 0; i < n + g1ce; i++)
    {
        cout << "V" << i;
        for (int j = 0; j < n + g1ce; j++)
        {
            cout << "\t" << g1cd[i][j];
        }
        cout << endl;
    }
    int f[n + g1ce];
    for (i = 0; i < n + g1ce; i++)
        f[i] = i;
    for (r = 2; r <= n && g1cflag == 0; r++)
     {
         //merge the following 2 lines
        g1cflag = g1cnextSetCombns(f,
                n, r, g1ce, g1cd);
    }
    return 0;
}
```

E Executing the Algorithm for $\beta(BG_2(\overline{G}))$ with C++ Code

A code is generated in C++ for the algorithm presented in section 6.2.1 and is given below.

```
#include <iostream>
using namespace std;
//Merge the following 2 lines
void g2cdfs(int g2cvertex, const int g2cgraph[][50],
    int g2cnumvertices, bool g2cvisited[])
{
    g2cvisited[g2cvertex] = true;

    for (int i = 0; i < g2cnumvertices; ++i)
    {
        if (g2cgraph[g2cvertex][i] && !g2cvisited[i])
        {
            g2cdfs(i, g2cgraph, g2cnumvertices, g2cvisited);
        }
    }
}

bool g2cConnect(int g2cnumvertices, const int g2cgraph[][50])
 {
    bool g2cvisited[50] = {false};
    g2cdfs(0, g2cgraph, g2cnumvertices, g2cvisited);
```

```
    for (int i = 0; i < g2cnumvertices; ++i)
    {
        if (!g2cvisited[i])
        {
            return false;  // Graph is not connected
        }
    }

    return true;  // Graph is connected
}
//Merge the following 3 lines
int g2cgenerateCombination(int f[], int start, int end,
int r, int g2ccombArr[], int g2ccombArrInd, int g2ce,
   int g2cd[50][50])
 {
    int i, j, m, k, status = 0, g2cflag;
    if (g2ccombArrInd == r)
        {
        g2cflag = 0;
        for (i = 0; i < end + 1 && g2cflag != 1; i++)
        {
            for (j = i + 1; j < end + 1 && g2cflag != 1; j++)
             {
                for (k = 0; k < r; k++)
                {
//Merge the following 2 lines
                    if (g2cd[i][g2ccombArr[k]]
                        != g2cd[j][g2ccombArr[k]])
                        break;
```

```
                    if (k == r - 1)
                    {
                        g2cflag = 1;
                    }
                }
            }
            if (i == end)
            {
                //Merge the following 2 lines
                cout << "\n\n metric dimension=" <<
                r << "\n\n A resolving set is\n {";
                for (i = 0; i < r; i++)
                    cout << " v" << g2ccombArr[i] << ",";
                cout << "\b }\n\n";
                return 1;
            }
        }
        return 0;
    }
//Merge the following 2 lines
    for (i = start; i <= end && end - i + 1 >= r -
    g2ccombArrInd && status == 0; i++)
        {
        g2ccombArr[g2ccombArrInd] = f[i];
//Merge the following 2 lines
        status = g2cgenerateCombination(f, i + 1, end, r,
                g2ccombArr, g2ccombArrInd + 1, g2ce, g2cd);
    }
    return status;
```

```
}
//Merge the following 2 lines
int g2cnextSetCombns(int f[], int n, int r,
                     int g2ce, int g2cd[50][50])
                {
    int tempArr[r];
//Merge the following 2 lines
    return g2cgenerateCombination(f, 0, n + g2ce - 1,
                          r, tempArr, 0, g2ce, g2cd);
}

int main()
{
    int g2cb[50][50], g2cd[50][50];
    int g2ce = 0, n, r, i, g2cflag = 0, g2cg, g2cu, g2cw;
//Merge the following 2 lines
    printf("Enter the number of vertices of the simple
           connected Graph G with more than 2 vertices ");
    cin >> n;
    int g2ca[n][n];
//Merge the following 2 lines
cout << "Enter the upper triangular part of the adjacency
     matrix of G excluding the diagonal elements " << endl;
    for (int i = 0; i < n; i++)
    {
        g2ca[i][i] = 0;
        g2cb[i][i] = 0;
        g2cd[i][i] = 0;
        for (int j = i + 1; j < n; j++)
```

```
{
    cin >> g2ca[i][j];
    g2ca[i][j] = 1 - g2ca[i][j];
    g2ca[j][i] = g2ca[i][j];
    g2cb[i][j] = g2ca[i][j];
    g2cb[j][i] = g2cb[i][j];
    if (g2ca[i][j] == 1)
    {
        g2ce = g2ce + 1;
        g2cd[i][j] = 1;
        g2cd[j][i] = 1;
        g2cb[j][n + g2ce - 1] = 1;
        g2cb[i][n + g2ce - 1] = 1;
        g2cb[n + g2ce - 1][j] = 1;
        g2cb[n + g2ce - 1][i] = 1;
        g2cd[j][n + g2ce - 1] = 1;
        g2cd[i][n + g2ce - 1] = 1;
        g2cd[n + g2ce - 1][j] = 1;
        g2cd[n + g2ce - 1][i] = 1;
        g2cb[n + g2ce - 1][n + g2ce - 1] = 0;
        for (int k = 0; k < n; k++)
        {
            if (k != i && k != j)
            {
                g2cb[k][n + g2ce - 1] = 0;
                g2cb[n + g2ce - 1][k] = 0;
                g2cd[k][n + g2ce - 1] = 2;
                g2cd[n + g2ce - 1][k] = 2;
            }
```

```
            }
            g2cg = i;
            g2cw = j;
            if (g2ce > 1)
            {
for (int g2cg = n; g2cg < n + g2ce - 1; g2cg++)
                {
if (g2cb[i][g2cg] == 0 && g2cb[j][g2cg] == 0)
                    {
                        g2cb[n + g2ce - 1][g2cg] = 1;
                        g2cb[g2cg][n + g2ce - 1] = 1;
                        g2cd[n + g2ce - 1][g2cg] = 1;
                        g2cd[g2cg][n + g2ce - 1] = 1;
                    } else
                    {
                        g2cb[n + g2ce - 1][g2cg] = 0;
                        g2cb[g2cg][n + g2ce - 1] = 0;
                        g2cd[n + g2ce - 1][g2cg] = 2;
                        g2cd[g2cg][n + g2ce - 1] = 2;
                    }
                }
            }
        }
        if (g2ca[i][j] == 0)
        {
            g2cb[i][j] = 0;
            g2cb[j][i] = 0;
        }
    }
```

```
}
for (int i = 0; i < n; i++)
{
    for (int j = i + 1; j < n; j++)
     {
        if (g2ca[i][j] == 0)
         {
            for (int k = 0; k < n; k++)
             {
                if (g2cb[i][k] == 1 && g2cb[j][k] == 1)
                 {
                    g2cd[i][j] = 2;
                    g2cd[j][i] = 2;
                    break;
                }
                if (k == n - 1)
                 {
                    g2cd[i][j] = 3;
                    g2cd[j][i] = 3;
                }
            }
        }
    }
}

bool isBConnected = g2cConnect(n + g2ce, g2cb);

if (!isBConnected)
{
```

```
        //Merge the following 2 lines
        cout << "The graph BG2(G complement)
        is disconnected." << endl;
        return 0;
    }
    //Merge the following 2 lines
    cout << "Adjacency matrix of BG2(G)"
    << endl << endl;
    for (int i = 0; i < n + g2ce; i++)
        cout << "\t" << "V" << i;
    cout << endl;
    for (int i = 0; i < n + g2ce; i++)
    {
        cout << "V" << i;
        for (int j = 0; j < n + g2ce; j++)
        {
            cout << "\t" << g2cb[i][j];
        }
        cout << endl;
    }
    //Merge the following 2 lines
    cout << endl << "Distance matrix of BG2(G)"
    << endl << endl;
    for (int i = 0; i < n + g2ce; i++)
        cout << "\t" << "V" << i;
    cout << endl;
    for (int i = 0; i < n + g2ce; i++)
     {
        cout << "V" << i;
```

```
        for (int j = 0; j < n + g2ce; j++)
        {
            cout << "\t" << g2cd[i][j];
        }
        cout << endl;
    }
    int f[n + g2ce];
    for (i = 0; i < n + g2ce; i++)
        f[i] = i;

    for (r = 2; r <= n && g2cflag == 0; r++)
    {
        //Merge the following 2 lines
        g2cflag = g2cnextSetCombns(f, n,
                                   r, g2ce, g2cd);
    }
    return 0;
}
```