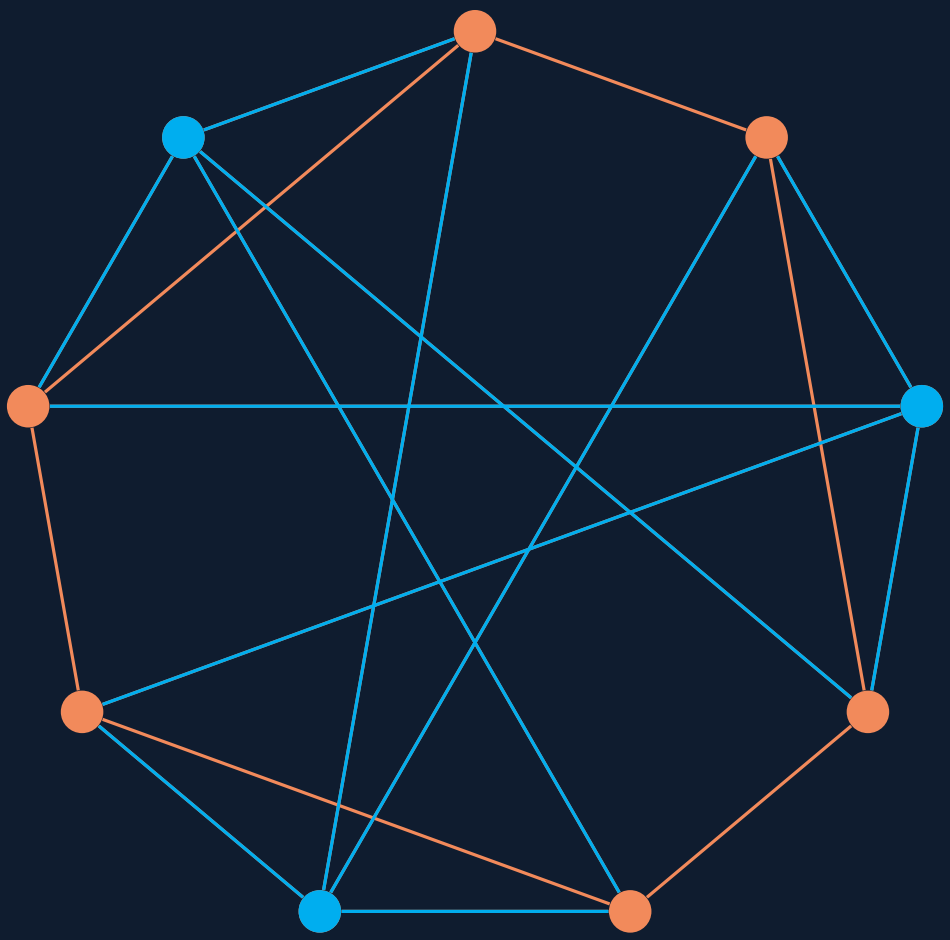
A Note on Integrity

ILP Modelling and Analysis on Graph Families

M.J.P. Reinders



Delft University of Technology

TU Delft

A Note on Integrity

ILP Modelling and Analysis on Graph Families

by

M.J.P. Reinders

to obtain the degree of Bachelor of Science
at the Delft University of Technology,
to be defended publicly on Monday July 17, 2023 at 1:30 PM.

Student number:	4899334	
Institution:	Delft University of Technology	
Thesis committee:	Dr. A. Bishnoi,	TU Delft, supervisor
	Dr. ir. R.J. Fokkink,	TU Delft
Project duration:	April, 2023 – July, 2023	

TU Delft

Preface

This thesis was written under the Department of Discrete Mathematics & Optimization of the Faculty of Electrical Engineering, Mathematics, and Computer Science at Delft University of Technology.

During the selection process of choosing my topic for this thesis, my attention was immediately drawn to graph theory topics. I developed a strong interest in this field while taking a course on graph theory. After considering various topics within this field, I eventually narrowed it down to the final topic of this thesis: The Integrity of a Graph. I chose this topic due to its connection to the min-max problem, which further fueled my growing fascination with it during the research.

I would like to thank my supervisor, Anurag Bishnoi, for guiding me throughout the entire process of creating this thesis. His invaluable guidance during my first experience in conducting original research and proving new statements has been instrumental to my progress. The weekly meetings with Anurag throughout my work on the thesis provided me with a solid foundation and helped me track my research and writing progress. I am also grateful for the continuous refinement and updating of the thesis direction based on the valuable feedback I received. Lastly, I extend my thanks to Robbert Fokkink for being a part of my graduation committee.

M.J.P. Reinders
Delft, July, 2023

Abstract

This thesis provides a fresh perspective on the (vertex) integrity of graphs, serving as a measure of network robustness. The study begins by introducing fundamental concepts and methods for evaluating the integrity of different graph families. An Integer Linear Programming (ILP) model, specifically designed for assessing integrity, is then presented. By applying this model, integrity values are calculated for various graph families, and patterns within the results are identified. These patterns contribute to establishing boundaries or determining exact values of integrity for the analyzed graph families.

The analyzed graph families encompass Glued Paths, generalized Theta graphs, (Double) Cone graphs, Fan graph, Lollipop graphs, generalized Barbell graphs, (Dutch) Windmill graphs, Paley graphs, and Kneser graphs. Additionally, two conjectures are formulated: one concerning a lower bound for the integrity of all Paley graphs, and another addressing the exact integrity values of specific Kneser graphs.

The ILP model proves to be a valuable tool for further exploration of graph family integrity, offering opportunities for additional research and expanding our understanding of network robustness.

Layman Abstract

This thesis explores how different networks, like those in transportation systems, social networks, and computer networks, can maintain its functionality and integrity in the face of disruptions, failures, or targeted attacks. It introduces a measure called (vertex) integrity to assess the strength and resilience of networks.

To evaluate network integrity, a mathematical approach called Integer Linear Programming is used. This method enables precise calculations and comparisons of network robustness. By applying this approach to various types of networks, including Theta graphs, Paley graphs, Kneser graphs, and more, the study uncovers valuable insights.

Through the analysis of integrity values, the study identifies patterns and limitations in network resilience. It sheds light on the specific characteristics and behaviors that contribute to a network's ability to recover from disruptions. These findings have practical implications for designing and optimizing networks to be more robust and reliable.

The mathematical model employed in this research establishes a strong foundation for studying network resilience. It enhances our understanding of network behavior and opens avenues for further research in exploring different aspects of network strength. This knowledge advancement allows us to develop strategies to enhance the resilience of real-world networks, ensuring their ability to withstand unforeseen challenges and maintain effective operations.

Contents

1

Introduction

In today's world, networks play a vital part in our daily lives. From social connections to transportation systems, understanding the structures and dynamics of these networks is crucial. This is where graph theory plays an important role, providing a powerful framework for studying, analyzing, and understanding the complex relations of networks. Its origin dates back to the 18th century when Swiss mathematician Leonhard Euler introduced the concept of a graph via a paper in 1741. This first paper on graph theory, about the Seven Bridges of Königsberg problem, laid the foundation of the field [14].

The city of Königsberg (now Kaliningrad, Russia) was located on both sides of the Pregel River, consisting of two large islands connected to each other and the mainland by seven bridges, as shown in Figure 1.1. The problem was whether it was possible to take a walk through the city crossing each bridge exactly once. This type of walk is now referred to as an Eulerian Trail.

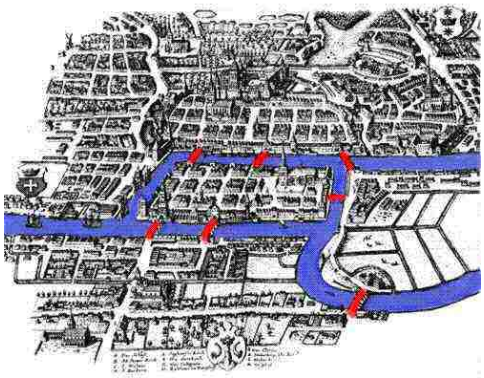


Figure 1.1: A view of Königsberg showing the seven bridges over the River Pregel [30].

Euler took an abstract approach to the problem, focusing only on which pairs of land masses are connected to each other and the number of bridges connecting them. This abstraction allowed him to focus on the essential connections rather than the physical layout of the city. Euler proved that it was impossible to find such a path, and his proof laid the foundations of graph theory.

Euler's solution to the Seven Bridges of Königsberg problem marked the birth of graph theory as a mathematical discipline. It established the idea of representing real-world problems using abstract graphs. It also laid the groundwork for the development of graph-related concepts, such as connectivity, independence number, regularity, Hamiltonicity, and many other graph properties and algorithms.

Since Euler's time, graph theory has evolved into a fundamental discipline with broad applications in various fields, including computer science, social network analysis, logistics, and optimization. The

first graph problem posed by Euler remains an important historical milestone that paved the way for the extensive study and application of graphs in modern mathematics and beyond.

1.1. Robustness of a Network

In addition to understanding the structure and dynamics of networks, assessing the robustness of a network holds significant value. The robustness refers to the ability of a network to maintain its functionality and integrity in the face of disruptions, failures, or targeted attacks. It is a vital consideration in various domains, including transportation systems, communication networks, and social networks.

The early research on robustness can be traced back to the 1970s [16, 25]. One of the first measures for the robustness of a graph is known as the vertex connectivity [15]. The vertex connectivity quantifies the minimum number of nodes that need to be removed in order to disconnect the network into two or more groups. In other words, it measures the resilience of a network against node failures.

The vertex connectivity of a graph is a fundamental measure in assessing its robustness and can provide insights into the network's ability to withstand failures or targeted attacks on individual nodes. By studying the vertex connectivity of a graph, researchers can identify critical nodes that, if compromised or removed, could significantly impact the network's functionality and integrity.

However, the vertex connectivity alone may not fully encompass the network's robustness. The vertex connectivity investigates the connectedness of the network after node deletion, but overlooks the intrinsic structure of the remaining groups after this deletion. This is where vertex integrity becomes evident.

To illustrate this point, consider two networks with identical vertex connectivity. Despite their identical vertex connectivity, their levels of integrity can vary significantly. In Chapter 2 we will explore a compelling example that clearly demonstrates this distinction. This example highlights the importance of vertex integrity as it offers a more thorough evaluation of a network's robustness, surpassing the sole consideration of vertex connectivity.

The vertex integrity, commonly referred to as integrity, was originally introduced in the late 1980s by Barefoot, Entringer, and Swart [6]. It considers two quantities: the number of deleted nodes and the size of the largest remaining group after deletion [3].

Until now, we have primarily mentioned the practical applications of vertex integrity in transportation systems. However, recently, vertex integrity has played a role in resolving one of the central open problems on minimal codes, and strong blocking sets [2].

1.2. Overview

The goal of this thesis is to investigate the integrity of specific graph families. To achieve this, we will first study the integrity of general graphs. Next, we aim to model certain properties of a graph, including the integrity, through Integer Linear Programming (ILP). These ILP models will help us discover patterns in the integrity of graph families, enabling us to establish boundaries or determine exact values of the integrity for those graph families.

Firstly, in Chapter 2, we define several important concepts and properties that will be used throughout the thesis. Then, in Chapter 3, we introduce various concepts related to the integrity of general graphs. In Chapter 4, we define and explain our ILP models for calculating the values of specific graph properties. Combining these ILP models with the concepts from Chapter 3, we prove exact values or bounds for the integrity of particular graph families in Chapter 5. Additionally, Chapter 6 delves into the open problems encountered during our research. In conclusion, we provide suggestions for further research opportunities in this thesis.

2

Preliminary

In this chapter, we will provide a brief overview of the graph theory concepts used in this thesis. For further notation, refer to the book 'Introduction to Graph Theory' by West [32].

Firstly, we define a network, its structure, and its nodes in mathematical terms, using the definition of a graph. In this thesis, we consider only undirected simple graphs.

Definition 2.1. *A **graph** G = (V, E) consists of a set of **vertices** V and a set of **edges** E, where V is non-empty and E is a subset of the set* $\{\{u,v\} : u,v \in V, u \neq v\}$*, for convenience denoted by* $\binom{V}{2}$*, of all two-element subsets of V. The set* $\{u,v\}$ *is commonly denoted by* uv*. By this definition,* G *is **undirected**, since for each edge it holds that* $e = \{u,v\} = \{v,u\}$*.* G *is also **simple**, since there is at most one edge between two vertices and none from and to the same vertex.*

Remark 2.2. *Let G = (V, E) be a graph. The **order** of G is the number of vertices, denoted by either* n *or* $|V|$*. Generally the order is denoted by* n*.*

Graphs can be visualized by a graph drawing. When visualizing, we use filled circles for the vertices and connect two circles by a line, or rather a curve, if there is an edge between the corresponding vertices.

Example 2.3. *Figure 2.1 depicts a graph drawing of the Petersen graph, a graph of order 10 with 15 edges, i.e.* $|E| = 15$

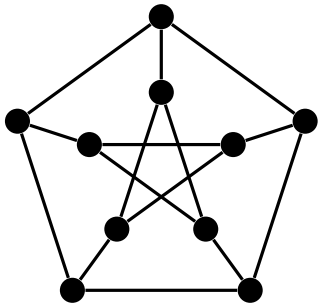


Figure 2.1: Graph drawing of the Petersen graph, a graph of order 10 with 15 edges, i.e. $|E| = 15$

2.1. Basic Graph Properties

Besides the network and its structure, it is necessary to define notations pertaining to the properties of a network in order to facilitate analysis. In our context, these notations are known as graph properties.

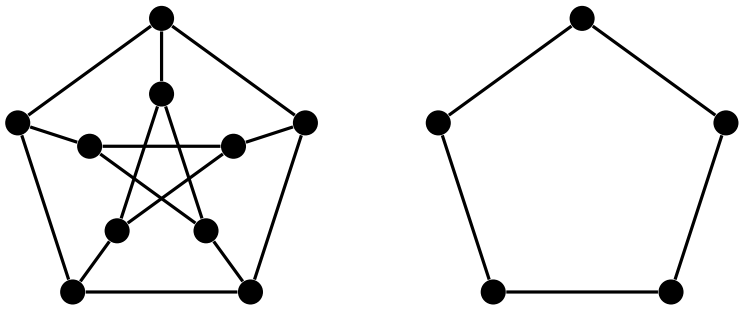
Graph properties encompass various categories, including measures of both local and global characteristics. Local graph properties are specific to individual vertices or edges within a graph, capturing their unique characteristics and features. Conversely, global graph properties encompass characteristics that apply to the entire graph as a unified entity, considering its overall structure. These properties offer a comprehensive understanding of the graph, capturing its holistic behavior.

Both local and global graph properties play a crucial role in comprehending the characteristics, behavior, and functionality of a graph. They provide valuable insights into different facets of the network's structure and dynamics, enabling analysis and interpretation from diverse perspectives. By examining these properties, we gain a deeper understanding of the network's intricacies.

Consequently, we need to define notations to refer to subnetworks of the network. The following definitions concerning (sub)graphs will facilitate this.

Definition 2.4. *For any graph G = (V, E), a **subgraph** of G is the graph H = (V', E') such that $V' \subseteq V$ and $E' \subseteq E \cap \binom{V'}{2}$. The subgraph is **induced** if $E' = E \cap \binom{V'}{2}$. A **proper** subgraph is a subgraph with either $V' \subset V$ or $E' \subset E \cap \binom{V'}{2}$.*

Example 2.5. *Figure 2.2 depicts a graph drawing of the Petersen graph, along with a proper induced subgraph, the cycle C_5 of order 5.*



(a) The Petersen graph. (b) The cycle C_5 of order 5.

Figure 2.2: The Petersen graph, along with a proper induced subgraph, the cycle C_5 of order 5.

Definition 2.6. *Let G = (V, E) be a graph. For any $S \subseteq V$, The graph $G - S$ is the proper induced subgraph H = (V', E') of G with $V' = V \setminus S$. For any $v \in V$, $G - \{v\}$ is commonly denoted by $G - v$. Another notation for $G - S$ is $G[V']$.*

Example 2.7. *Figure 2.2 depicts a graph drawing of the Petersen graph, along with the cycle C_5 of order 5. Let G be the Petersen graph and let S be the 5 inner vertices of the Petersen graph, then $G - S = C_5$.*

Now that we have the ability to mathematically describe a network and its subnetwork, it is essential to establish some fundamental terminology that will allow us to describe the structure of the graph and its (sub)parts.

Definition 2.8. *In the graph $G = (V, E)$, two vertices $u, v \in V$ are **adjacent** if $uv \in E$ and **non-adjacent** if $uv \notin E$. A set of pairwise non-adjacent vertices in a graph is called an **independent set**, also known as a **co-clique**. Conversely, a set of pairwise adjacent vertices is called a **clique**.*

Example 2.9. *Figure 2.3 depicts a graph drawing of the Petersen graph, in which the yellow vertices form an independent set and the blue vertices a clique.*

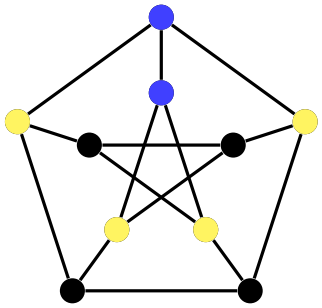


Figure 2.3: The Petersen graph, in which the yellow vertices form an independent set and the blue vertices a clique.

Definition 2.10. *Let G = (V, E) be a graph and $u, v \in V$. A **path** from u to v is the set of distinct vertices $v_0, v_1, \ldots, v_k$ such that $u = v_0$, $v = v_k$, and $v_i v_{i+1} \in E$ for all $0 \leq i \leq k-1$. A graph is called **connected** if it either has only one vertex or there exists a path between any two distinct vertices.*

Definition 2.11. *Let G = (V, E) be a graph and $u, v \in V$. The relation $u \sim v$, if u and v are connected to each other by a path, is an equivalence relation of the vertices of G. The equivalence classes of this relation are known as connected **components** of the graph.*

Definition 2.12. *Let G = (V, E) consist of components $C_1, C_2, \ldots, C_k$ for some $k \in \mathbb{Z}_{\geq 0}$. hen the **amount of components in G**, denoted by $c(G)$, is $c(G) = k$.*

We now have the ability to describe a network, its subnetworks, and its (sub)parts in mathematical terms. Next, we need to define commonly used graph properties that will assist us in our analysis.

Definition 2.13. *Let G = (V, E) consist of components $C_1, C_2, \ldots, C_k$ for some $k \in \mathbb{Z}_{\geq 0}$. Then the **largest size of a (connected) component in G**, denoted by $m(G)$, is the integer*

$$m(G) := \max_{0 \leq i \leq k} |C_i|.$$

Example 2.14. *Figure 2.4 depicts graph G, a graph with 2 components. The components have sizes 2 and 3. Hence, the largest size of a (connected) component in G is, $m(G) = 3$.*

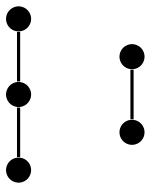


Figure 2.4: Graph drawing of graph G, a graph with two components. The components have sizes 2 and 3. Hence $m(G) = 3$.

Definition 2.15. *For any graph G, the **independence number**, denoted by $\alpha(G)$, is the largest size of an independent set in G.*

Definition 2.16. *For any graph G, the **clique number**, denoted by $\omega(G)$, is the size of the largest clique in G.*

Remark 2.17. *Determining the independence number is NP-hard [18] and the clique number is NP-complete [23], making it computationally challenging problems for general graphs.*

Example 2.18. *Consider the Petersen graph, with its graph drawing depicted in Figure 2.3. The set of yellow vertices represents the largest possible independent set in the graph. Similarly, the set of blue vertices represents the largest possible clique in the graph. Therefore, we can conclude that $\alpha(\text{Petersen graph}) = 4$ and $\omega(\text{Petersen graph}) = 2$.*

Definition 2.19. *Let $G = (V, E)$ and $u \in V$. Then the set $N_G(u) = \{v \in V : uv \in E\}$, is called the **neighbourhood** of u. The size of this set is the **degree** of vertex u, denoted by $\deg_G(u) = \#N_G(u)$. $N_G(u)$ and $\deg_G(u)$ are commonly denoted by $N_G(u)$ and $\deg_G(u)$ respectively.*

Definition 2.20. *Let G = (V, E) be a graph. The **minimum degree of G** is the integer*

$$\delta(G) := \min_{v \in V} \deg(v).$$

Definition 2.21. *Let G = (V, E) be a graph. The **maximum degree of G** is the integer*

$$\Delta(G) := \max_{v \in V} \deg(v).$$

Definition 2.22. *Let G = (V, E) be a graph. For any set $S \subseteq V$, the **maximum degree of S in G** is the integer*

$$\Delta(G, S) := \max_{s \in S} \deg_G(s).$$

Example 2.23. *Figure 2.5 depicts tree T of order 12. The orange vertex has a degree of 3, the minimum degree of the graph is $\delta(T) = 1$ and the maximum degree of the graph is $\Delta(T) = 3$. Let the magenta vertices form the set S. Then the maximum degree of S in T is $\Delta(T, S) = 1$.*

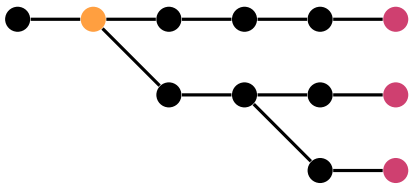


Figure 2.5: Graph drawing of tree T of order 12. The orange vertex has a degree of 3, $\delta(T) = 31$ and $\Delta(T) = 3$. Let the magenta vertices form the set S, then $\Delta(T, S) = 1$.

2.2. Spectral Graph Theory

An intriguing field within graph theory is spectral graph theory. This field utilizes algebraic methods and concepts from linear algebra to investigate various properties of graphs. Spectral graph theory focuses on the study of eigenvalues and eigenvectors of matrices associated with graphs, such as the adjacency matrix or Laplacian matrix. By examining the spectrum of these matrices, spectral graph theory provides insights into graph connectivity, graph coloring, graph clustering, and other structural properties. The use of spectral techniques enables a deeper understanding of graphs and their underlying structures, leading to applications in diverse fields such as computer science, physics, and social network analysis.

To begin, let us first establish the definition of an adjacency matrix.

Definition 2.24. *The **adjacency matrix** of a graph $G = (V, E)$ with n vertices is a square matrix A of size $n \times n$. The vertices of the graph are numbered from 1 to n. Each element $A_{i,j}$ of the adjacency matrix is set to 1 if the i-th and j-th vertices are adjacent, and 0 otherwise. Note that for undirected graphs, the adjacency matrix is a real symmetric matrix.*

Example 2.25. *Consider the empty graph $\overline{K}_n$ and the complete graph K_n. The adjacency matrix of $\overline{K}n$ is the empty matrix of size $n \times n$, where all elements Ai, j are 0. The adjacency matrix of K_n is the matrix $J_n - I_n$, where J_n is the matrix of size $n \times n$ with all elements Ai, j equal to 1, and I_n is the identity matrix of size $n \times n$.*

With the definition of the adjacency matrix, we can delve into the field of spectral graph theory by defining the eigenvalues of a graph.

Definition 2.26. *For any graph G with n vertices and adjacency matrix A, the eigenvalues of G are equivalent to the eigenvalues of the matrix A. The adjacency matrix A is a real symmetric matrix, and according to the real spectral theorem from linear algebra, it possesses real eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$. Furthermore, there exists an orthonormal basis in $\mathbb{R}^n$ consisting of eigenvectors of A.*

The eigenvalues serve as powerful indicators of the spectral nature of graphs, allowing us to explore their rich structural landscape.

Finally we use the definition of a spectrum to compact all the information into one expression.

Definition 2.27. *For any graph G, the **spectrum** of G is defined as the collection of its distinct eigenvalues with their corresponding multiplicities, denoted by $(\mu_1)^{m_1}, \ldots, (\mu_k)^{m_k}$.*

Example 2.28. *Consider the empty graph $\overline{K}_n$, the complete graph K_n, and the Petersen graph. The empty graph has spectrum $(0)^n$, the complete graph has spectrum $(n-1)^1, (-1)^{n-1}$, and the Petersen graph has spectrum $(3)^1, (1)^5, (-2)^4$.*

2.3. Basic Graph Structures

Now that we have established various concepts and properties for effective network analysis, it is essential to define specific structures that networks can have. These structures exhibit unique characteristics and play a significant role in shaping the behavior and properties of the network.

Definition 2.29. *A graph is called **d-regular**, if all vertices in the graph have degree d. If all vertices in a graph have the same degree, the graph is called **regular**.*

Example 2.30. *The cycle C_n is an example of a 2-regular graph. Similarly, the Petersen graph is a 3-regular graph. Conversely, the path P_n of order n is not regular because it has two vertices with a degree of 1, while the other vertices have a degree of 2.*

Definition 2.31. *[8] A **strongly regular graph**, denoted by $\mathrm{srg}(n, d, a, b)$, is a d-regular graph with n vertices that has the following 2 properties. Any two adjacent vertices have exactly a common neighbours and any two non-adjacent vertices have exactly b common neighbours.*

Example 2.32. *The Petersen graph is a $srg(10, 3, 0, 1)$, the cycle C_n is a $srg(n, 2, 0, 1)$ and the complete bipartite graph $K_{n,n}$ is a $srg(2n, n, 0, n)$.*

As networks become more complex, describing them can be challenging. However, by recognizing that a network consists of multiple known networks, the following definitions will assist us in describing these complex networks in familiar terms.

Definition 2.33. *[22] The union $G = G_1 \cup G_2$ of graphs G_1 and G_2, with disjoint sets V_1 and V_2 and edge sets E_1 and E_2, is the graph the graph with $V = V_1 \cup V_2$ and $E = E_1 \cup E_2$. The disjoint union of k copies of graph G is commonly denoted by kG, with the i-th copy denoted as G^i.*

Example 2.34. *Figure 2.6 depicts the graph $C_5 \cup K_4$, the graph union of the cycle C_5 of order 5 and the complete graph K_4 of order 4.*

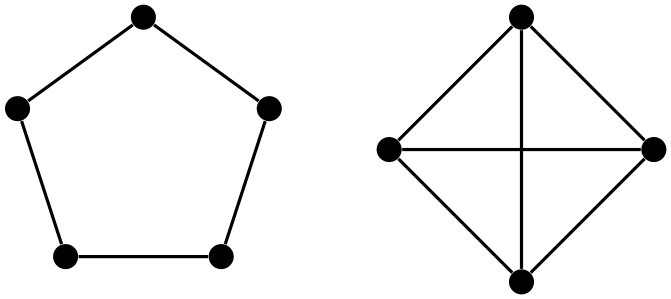


Figure 2.6: Graph drawing of $C_5 \cup K_4$, the graph union of the cycle C_5 of order 5 and the complete graph K_4 of order 4.

Definition 2.35. *[22] The join $G = G_1 + G_2$ of graphs G_1 and G_2, with disjoint sets V_1 and V_2 and edge sets E_1 and E_2, is the graph the graph union $G_1 \cup G_2$ together with all the edges joining V_1 and V_2.*

Example 2.36. *Figure 2.7 depicts the graph $P_4 + P_3$, the graph join of the path P_4 of order 4 and the path P_3 of order 3.*

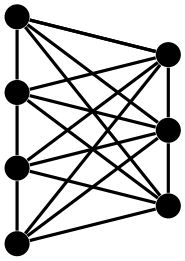


Figure 2.7: Graph drawing of $P_4 + P_3$, the graph join of the path P_4 of order 4 and the path P_3 of order 3.

2.4. Measures for the Robustness of a Network

Finally, we can mathematically define both measures, vertex connectivity and vertex integrity introduced in Chapter 1, for assessing the robustness of the network in mathematical terms.

Definition 2.37. *[15] Let G = (V, E) be a simple graph and let $\delta(G)$ denote the minimum degree of G. The **vertex connectivity** of G, denoted by $\kappa(G)$, is the minimum size of a vertex cut set, i.e., a set S ⊆ V such that G - S is disconnected or has only 1 vertex.*

Proposition 2.38. *[33] Let G = (V, E) be a simple graph and let $\delta(G)$ denote the minimum degree of G. Then, we have*

$$0 \leq \kappa(G) \leq \delta(G).$$

High vertex connectivity in a graph implies that the graph can endure the removal of vertices without losing its connectivity. Even if we delete $\kappa(G)$ vertices, the graph remains connected, ensuring that there is still a path between any two vertices.

Definition 2.39. *[6] Let G = (V, E) be a simple connected graph. For any subgraph H, let m(H) denote the largest size of a connected component in H. The **integrity** of G is the integer*

$$\iota(G) := \min_{S \subseteq V} \{|S| + m(G - S)\}.$$

High vertex integrity in a graph implies that the graph can endure the removal of vertices without losing its internal connectivity within the remaining groups. Even if some vertices are deleted, the remaining vertices form connected groups, ensuring that there is still a path between any two vertices within each

group. However, the graph as a whole may not remain connected if the removed vertices disconnect the entire graph. On the contrary, low vertex integrity suggests that removing a few critical vertices can cause the graph to break into disconnected components, with each component containing only a small subset of vertices.

The integrity values for various graph families have already been established and were initially discovered by Barefoot, Entringer, and Swart [5, 6]. In the following theorem, we present the integrity values for several graph families.

Theorem 2.40. *[3, 5, 6] The integrity of*
(a) the complete graph K_n is n;
(b) the null graph $\overline{K}_n$ is 1;
(c) the star $K_{1,n}$ is 2;
(d) the path P_n is $\lceil 2\sqrt{n+1} \rceil - 2$;
(e) the cycle C_n is $\lceil 2\sqrt{n} \rceil - 1$;
(f) the complete bipartite graph $K_{m,n}$ is $1 + \min\{m, n\}$;

To compare the two measures, we consider two examples: the star and a path.

Example 2.41. *Consider a train station network, where the vertices represent the train stations and the edges represent the railroads between the stations. Let the network have a structure that is identical to the star $K_{1,n-1}$ of order n, denoted by S_n. Figure 2.8 depicts a drawing of a star $K_{1,5}$ of order 6.*

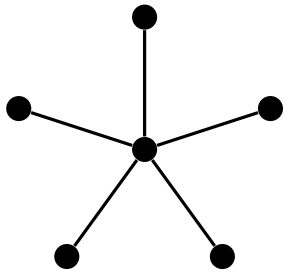


Figure 2.8: The star $K_{1,5}$ of order 6.

Then, we have $\kappa(S_n) = 1$ and $\iota(S_n) = 2$. This implies that regardless of the amount of vertices, the vertex connectivity is one and the vertex integrity is two. Both measures indicate a low level of robustness, as the removal of the center vertex disconnects the whole graph into components of size one. Applying this result to the train station network, we find that if the center train station experiences failure, the whole train station network will be shut down. Hence, the level of robustness in this train station network is very low, indicating that constructing a network with a star structure would not be the optimal choice in terms of robustness.

Example 2.42. *Consider a train station network, where the vertices represent the train stations and the edges represent the railroads between the stations. Let the network have the same structure as a path of order n, denoted by P_n. Figure 2.9 depicts a drawing of the path P_6 of order 6.*

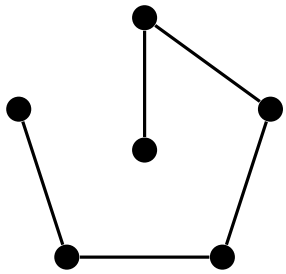


Figure 2.9: The path P_6 of order 6.

Then $\kappa(P_n) = 1$ *and* $\iota(P_n) = \lceil 2\sqrt{n+1} \rceil - 2$. *In this example there is a clear difference in the values of the measures. According to the vertex connectivity the level of robustness of the system is very low, as the removal of one vertex disconnects the graph. However, the integrity shows that even after removing a 'small' number of vertices the graph still has at least one relatively 'large' component, ensuring some level of connectivity.*

Upon careful analysis of the results obtained in Example 2.41 and 2.42, it becomes apparent that if we solely rely on vertex connectivity, both networks would be evaluated as having equal robustness. However, the integrity measure provides a more nuanced view, revealing that the network structured as a path maintains a higher level of connectivity even after node deletions compared to the network structured as a star. This observation highlights the significance of considering vertex integrity to gain a deeper understanding of a network's resilience, especially when comparing different network structures.

3

Integrity of a Graph

In this Chapter we focus on general concepts to determine bounds or exact values for the integrity. These concepts will be applied in Chapter 5.

Firstly, we aim to find a lower bound for the integrity, as it is the most challenging thing to compute in a minimization problem. To achieve this, we begin by seeking a lower bound for one of the terms in the definition, specifically the size of the largest component of the subgraph $G - S$.

Lemma 3.1. *Let G = (V, E) be a simple graph with n vertices. For any subgraph H, let $c(H)$ denote the amount of components in H. Then, for any set S ⊆ V,*

$$m(G - S) \geq \frac{n - |S|}{c(G - S)}.$$

Proof. Let $S \subseteq V$ be a set of vertices. Then $G - S$ has $c(G - S)$ components remaining. Then 1 component must have at least $\frac{n - |S|}{c(G - S)}$ vertices. To prove this, we use proof by contradiction. Thus assume all components have less than $\frac{n - |S|}{c(G - S)}$ vertices and for $1 \leq i \leq c(G - S)$ let C_i be the set of vertices that are in the i-th component. Then, we have

$$V(G - S) := \sum_{i=1}^{c(G-S)} |C_i| < \sum_{i=1}^{c(G-S)} \frac{n - |S|}{c(G - S)} = n - |S|.$$

This is a contradiction as there are $n - |S|$ vertices in $G - S$. Thus we have proven that there is a component with at least $\frac{n - |S|}{c(G - S)}$ vertices.

□

Lemma 3.1 provides a lower bound for the largest size of a component in the subgraph $G - S$, where G is our original graph, and S is a subset of vertices in graph G. When the graph breaks up into even components, meaning that the sizes of the components are relatively balanced, the lower bound given by the lemma is likely to be a tight bound. This implies that the actual largest size of a component in $G - S$ is close to the lower bound provided by the lemma.

However, in cases where the graph breaks into one big component and many small components, the lower bound may not be tight. This indicates that the actual largest size of the big component in $G - S$ could be significantly larger than the lower bound given by the lemma. In such scenarios, the lower bound does not accurately capture the size of the big component.

Substituting Lemma 3.1 into the definition of integrity, we obtain the following lower bound.

Theorem 3.2. *Let G = (V, E) be a simple connected graph with n vertices. For any subgraph H, let $c(H)$ denote the amount of components in H. Then, for any set S ⊆ V,*

$$\iota(G) \geq \min_{S \subseteq V} \left\{ |S| + \frac{n - |S|}{c(G - S)} \right\}.$$

Proof. Substituting Lemma 3.1 in Definition 2.39 gives

$$\iota(G) := \min_{S \subseteq V} \{|S| + m(G - S)\} \geq \min_{S \subseteq V} \left\{ |S| + \frac{n - |S|}{c(G - S)} \right\}.$$

□

Next, we seek to find an upper bound for the number of components in $G - S$ to obtain a more refined and improved lower bound for the integrity. By combining the lower bound for the largest component size and the upper bound for the number of components, we can derive a more accurate estimate of the integrity of the graph.

Lemma 3.3. *Let G = (V, E) be a simple connected graph with n vertices, and let $\deg_G(v)$ denote the degree of v in G. For any subgraph H, let $c(H)$ denote the number of components in H. Then, for any set $S \subseteq V$,*

$$c(G - S) \leq 1 + \sum_{s \in S} \deg_G(s) - |S|.$$

Proof. Let $S \subseteq V$ be a set of vertices. To proof this, we use strong induction on $|S|$.

Base: Take $|S| = 0$, then $S = \emptyset$. As G is connected, $G - S$ is also connected and has at most $1 \leq 1 + \sum_{s \in S} \deg_G(s) - |S|$ components.

Induction Step: Assume the hypothesis holds for sets with size $< |S|$ and that $|S| > 0$. Let $S = T \cup \{v\}$. Then $G - T$ has at most $1 + \sum_{t \in T} \deg_G(t) - |T|$ components remaining, since $|T| < |S|$. Now v is either a vertex cut or not. If v is a vertex cut, there are at most $\deg_G(v) - 1$ components more. If v is not a vertex cut, there are at most 0 components more. So in either case the are at most $\deg_G(v) - 1$ components more. Hence for any $S \subseteq V$, $G - S$ has at most $1 + \sum_{s \in S} \deg_G(s) - |S|$ components remaining.

□

Example 3.4. *This is an example for which the upper bound provided in Lemma 3.3 is strict. Let's consider the Star $K_{1,n}$ with $n + 1$ vertices, and let the center vertex form our set $S \subseteq V$. Thus $|S| = 1$. Then, by Lemma 3.3,*

$$c(K_{1,n} - S) \leq 1 + n - 1 = n,$$

and we know that this upper bound is strict as $c(K_{1,n} - S) = n$.

Lemma 3.3 provides an upper bound for the number of components in the subgraph $G - S$. When every vertex v in S creates $\deg_G(v) - 1$ components, the upper bound is tight. This means that in scenarios where each vertex in S contributes exactly $\deg_G(v) - 1$ components to the subgraph, the upper bound accurately reflects the actual number of components in $G - S$. Consequently, the lemma is more likely to be tight for vertices with small degrees.

Substituting Lemma 3.3 into the obtained lower bound for the largest size of a component and the integrity, we obtain the following lower bound for the largest size of a component in the subgraph $G - S$ and the integrity.

Lemma 3.5. *Let G = (V, E) be a simple connected graph with n vertices and let $\deg_G(v)$ denote the degree of v in G. Then, for any set S ⊆ V,*

$$m(G - S) \geq \frac{n - |S|}{1 + \sum\limits_{s \in S} \deg_G(s) - |S|}.$$

Proof. Substituting Lemma 3.3 in Lemma 3.1 gives

$$m(G-S) \geq \frac{n-|S|}{c(G-S)} \geq \frac{n-|S|}{1+\sum\limits_{s\in S}\deg_G(s)-|S|}.$$

□

Substituting Lemma 3.5 into the definition of integrity, we get the following lower bound.

Corollary 3.6. *Let G = (V, E) be a simple connected graph with n vertices, and let $\deg_G(v)$ denote the degree of v in G. Then,*

$$\iota(G) \geq \min_{S\subseteq V}\left\{|S| + \frac{n-|S|}{1+\sum\limits_{s\in S}\deg_G(s)-|S|}\right\}.$$

Proof. Substituting Lemma 3.5 in Theorem 3.2 gives

$$\iota(G) \geq \min_{S\subseteq V}\left\{|S| + \frac{n-|S|}{c(G-S)}\right\} \geq \min_{S\subseteq V}\left\{|S| + \frac{n-|S|}{1+\sum\limits_{s\in S}\deg_G(s)-|S|}\right\}.$$

□

Now, in cases where the exact degrees of each vertex in S are unknown, we seek to find a compact term that captures this uncertainty. This leads us to the following lower bound for the integrity.

Corollary 3.7. *Let G = (V, E) be a simple connected graph with n vertices, let $\Delta(G,S)$ denote the maximum degree of S in G. Then,*

$$\iota(G) \geq \min_{S\subseteq V}\left\{|S| + \frac{n-|S|}{1+(\Delta(G,S)-1)|S|}\right\}.$$

Proof. As $\sum\limits_{s\in S}\deg_G(s) \leq \Delta(G,S)|S|$, by Corollary 3.6, we obtain

$$\iota(G) \geq \min_{S\subseteq V}\left\{|S| + \frac{n-|S|}{1+\sum\limits_{s\in S}\deg_G(s)-|S|}\right\} \geq \min_{S\subseteq V}\left\{|S| + \frac{n-|S|}{1+(\Delta(G,S)-1)|S|}\right\}.$$

□

In connected graphs, the largest component size is equal to the number of vertices in the graph. By utilizing this fact, along with the observation that the integrity is bounded above by the number of vertices, we can arrive at the following equality for the integrity.

Theorem 3.8. *[3] For any simple connected graph G = (V, E) with n vertices,*

$$\iota(G) = 1 + \min_{v\in V}\iota(G-v).$$

Proof. From Definition 2.39, we have that

$$\iota(G) = \min\left\{m(G), 1+\min_{v\in V}\iota(G-v)\right\}.$$

Since G is connected, we have that $m(G) = n$ and we know from this definition that $\iota(G-v) \leq n-1$. Hence we get that $1+\iota(G-v) \leq m(G)$.

□

By utilizing vertex connectivity, we can further extend the equality obtained in Theorem 3.8 to the following equality for the integrity.

Corollary 3.9. *Let G = (V, E) be a simple graph, and let $\kappa(G)$ denote the vertex connectivity of G. Then, for any $0 \leq k \leq \kappa(G)$,*

$$\iota(G) = k + \min_{\substack{S \subseteq V, \\ |S|=k}} \iota(G - S).$$

Proof. We use strong induction on k.

Base: Take $k = 0$, then

$$\iota(G) = 0 + \min_{\substack{S \subseteq V, \\ |S|=0}} \iota(G - S) = \iota(G).$$

Induction Step: Assume the hypothesis holds for $k - 1$ and that $0 < k <= \kappa(G)$. For $k - 1$, we have

$$\iota(G) = (k-1) + \min_{\substack{S \subseteq V, \\ |S|=(k-1)}} \iota(G - S).$$

For any set $S \subseteq V$ with $|S| = k - 1$, since $|S| < \kappa(G)$, $G - T$ is connected. By Theorem 3.8, we get

$$\iota(G) = (k-1) + \min_{\substack{S \subseteq V, \\ |S|=(k-1)}} \left\{1 + \min_{v \in V} \iota((G - S) - v)\right\} = k + \min_{\substack{S \subseteq V, \\ |S|=k}} \iota(G - S).$$

□

For the purpose of case analysis, it is beneficial to split the integrity into two components: one where the exact value is known, and the other where a bound can be established. In Chapter 5, the following corollary will be utilized extensively.

Corollary 3.10. *Let $G = (V, E)$ be a simple connected graph. For any set $S \subseteq V$, let S^c denote the set $V \setminus S$. Then, for any non-empty set $T \subseteq V$,*

$$\iota(G) = 1 + \min\left\{\min_{v \in T} \iota(G - v), \min_{v \in T^c} \min_{S \subseteq T^c \setminus v} \{|S| + m(G - v - S)\}\right\}.$$

Proof. As G is connected, by Theorem 3.8, we have

$$\iota(G) = 1 + \min_{v \subseteq V} \iota(G - v).$$

Now, we can split the minimum into two cases: one where the vertex v is in the set T, and another where it is not. Thus, we have

$$\min_{v \subseteq V} \iota(G - v) = \min\left\{\min_{v \subseteq T} \iota(G - v), \min_{v \subseteq T^c} \iota(G - v)\right\}.$$

The right inner minimum above can be expressed, by Definition 2.39, as

$$\min_{v \subseteq T^c} \iota(G - v) = \min_{v \subseteq T^c} \min_{S \subseteq V \setminus v} \{|S| + m(G - v - S)\}$$

Expanding this form leads to cases: one where the set S has a vertex from the set T, and another where there is no such vertex. Therefore, we get

$$\min_{v \subseteq T^c} \iota(G - v) = \min_{v \subseteq T^c} \left\{ \begin{array}{cc} \min\limits_{\substack{S \subseteq V \setminus v \\ S \cap T = \emptyset}} \{|S| + m(G - v - S)\}, & \min\limits_{\substack{S \subseteq V \setminus v \\ S \cap T \neq \emptyset}} \{|S| + m(G - v - S)\} \end{array} \right\}$$

Focusing only on the right inner minimum above, by Definition 2.39, we obtain

$$\begin{aligned}\min_{v\subseteq T^c}\min_{\substack{S\subseteq V\setminus v\\ S\cap T\neq\emptyset}}\{|S|+m(G-v-S)\} &= \min_{v\subseteq T^c}\min_{\substack{S\subseteq V\setminus v\\ t\in S\cap T}}\{|S|+m(G-v-S)\}\\ &= \min_{v\subseteq T^c}\min_{\substack{S\subseteq V\setminus v\\ t\in S\cap T}}\{|S|+m(G-t-((S\setminus t)\cup v))\}\\ &= \min_{t\subseteq T}\min_{\substack{S\subseteq V\setminus t\\ S\cap T^c\neq\emptyset}}\{|S|+m(G-t-S)\}\\ &\geq \min_{t\subseteq T}\min_{S\subseteq V\setminus t}\{|S|+m(G-t-S)\}\\ &= \min_{v\subseteq T}\iota(G-v)\end{aligned}$$

Combining these results, we conclude that

$$\iota(G)=1+\min_{v\subseteq V}\iota(G-v).$$

□

Now, let's investigate the connection between the integrity of a graph and its subgraph. As stated in Theorem 3.8, this equality suggests the existence of a subgraph within the original graph that has a lower integrity value.

Corollary 3.11. *Let G = (V, E) be a simple connected graph with $n>1$ vertices. Then there exists a subgraph H with $n-1$ vertices, such that*

$$\iota(H)<\iota(G).$$

Proof. As G is connected, by Corollary 3.9, we obtain

$$\iota(G)=1+\min_{\substack{S\subseteq V,\\ |S|=1}}\iota(G-S)>\min_{\substack{S\subseteq V,\\ |S|=1}}\iota(G-S)$$

This implies that there exists a vertex $v\in V$, such that

$$\iota(G-v)=\min_{\substack{S\subseteq V,\\ |S|=1}}\iota(G-S).$$

Now let $H=G-v$, then H is a proper subgraph with $n-1$ vertices and $\iota(H)<\iota(G)$. □

Moreover, the definition of integrity gives rise to the following inequality.

Theorem 3.12. *[3] Let G be a simple graph. Then, for any subgraph H,*

$$\iota(H)\leq\iota(G).$$

Proof. Let $G=(V,E)$. For any subgraph $H=(V',E')$, we have

$$m(H-S)\leq m(G-S).$$

By Definition 2.39, we get

$$\iota(H)=\min_{S\subseteq V'}\{|S|+m(H-S)\}\leq\min_{S\subseteq V'}\{|S|+m(G-S)\}\leq\min_{S\subseteq V}\{|S|+m(G-S)\}=\iota(G).$$

□

By utilizing the clique number, we can establish the following equality for the integrity.

Proposition 3.13. *For every graph G = (V, E), let $\omega(G)$ denote the clique number of G. Then, for any $0 \leq k \leq \omega(G)$,*

$$\iota(G) = \min_{\substack{S \subseteq V \\ m(G-S) \geq k}} \{|S| + m(G-S)\}.$$

Proof. As $0 \leq k \leq \omega(G)$, there exists a clique of size k. Let the vertices in this clique form the set C. Then, for any set $T \subseteq V \setminus C$, we have $m(G-T) \geq k$. To lower the $m(G-T)$ by an integer x, we need to at least remove x vertices from this clique. Let $U \subseteq C$. This implies that $m(G-(T \cup U)) \geq k - |U|$. Thus, $|T \cup U| + m(G-(T \cup U)) \geq |T| + k$. Note that, $|T| + m(G-T) \geq |T| + k$. Hence,

$$\iota(G) = \min_{\substack{S \subseteq V \\ m(G-S) \geq k}} \{|S| + m(G-S)\}.$$

□

Proposition 3.13 will be highly valuable in Chapter 5. This proposition enables a reduction in the search space, allowing us to explore fewer possibilities for the set S. As a result, it enhances our ability to establish lower bounds for the integrity.

By utilizing the independence number, we can establish the following upper bounds for the integrity.

Proposition 3.14. *For any simple graph $G = (V, E)$ with n vertices, let $\alpha(G)$ denote the independence number of G. Then,*

$$\iota(G) \leq n - \alpha(G) + 1.$$

Proof. Let T be an independent set in G of size $\alpha(G)$, and let $S = V \setminus T$. As T is an independent set, we have $m(G-S) = 1$. As $|S| = n - \alpha(G)$, by Definition 2.39, we obtain

$$\iota(G) \leq |S| + m(G-S) = n - \alpha(G) + 1.$$

□

Remark 3.15. *Some cases for which this bound is strict are the complete graph K_n, the empty graph $\overline{K}_n$, the complete bipartite graph $K_{m,n}$, and the star $K_{1,n}$.*

The upper bound presented in Proposition 3.14 is particularly interesting because it applies to any graph. It raises the intriguing question of which graphs satisfy this bound strictly.

3.1. Graph Unions

For graph unions, the most relevant aspect for this thesis is the integrity of the union of copies of the same graph.

Now redefine the integrity measure to specifically focus on determining the integrity of copies of the same graph. This will simplify the process of finding the value for the integrity in such cases.

Theorem 3.16. *Let H = (V, E) be a simple connected graph. Then, for any $k \geq 1$,*

$$\iota(kH) = \min_{S \subseteq V} \{k|S| + m(H-S)\}.$$

Proof. Let kH = (V', E'), where H has n vertices. All connected components have exactly n vertices, thus $m(kH) = n$. To lower $m(kH)$, a vertex must be removed from all components. To minimize the

amount of removed vertices, we will remove exactly x vertices in each component H^i. Thus $|S| = kx$, with $0 \leq x \leq n$. Let $S_i = \{v \in S : v \in H^i\}$. As $|S_i| = x$, by Definition 2.39, we have

$$\begin{aligned}\iota(kH) &:= \min_{S \subseteq V'} \{|S| + m(kH - S)\} \\ &= \min_{S \subseteq V'} \left\{|S| + \max_{1 \leq i \leq k} m(H^i - S_i)\right\} \\ &= \min_{S \subseteq V} \{k|S| + m(H - S)\}.\end{aligned}$$

□

Consider the case where the number of graph copies exceeds the number of vertices in the largest connected component, we can arrive at the following equality.

Corollary 3.17. *[3] Let H be a simple graph. Then, for any $k \geq 1$ such that $m(H) < k$,*

$$\iota(kH) = m(H).$$

Proof. As $m(H - S) \leq m(H) < k$, by Theorem 3.16, we have

$$\iota(kH) = \min_{S \subseteq V} \{k|S| + m(H - S)\} < \min_{S \subseteq V} \{k|S| + k\} = k.$$

As $k|S| + m(H - S) < k$, we get that $|S| = 0$, thus S has to be the empty set. Hence, $\iota(kH) = m(H)$.

□

The following results will be used to prove exact values and bounds for various graph families in Chapter 5. First, we determine the value of the integrity for the union copies of paths.

Theorem 3.18. *Let kP_l be a disjoint union of k copies of the path P_l, and let $x = \max\left\{0, \frac{-k+\sqrt{k(l+1)}}{k}\right\}$. Then,*

$$\iota(kP_l) = \min\left\{k\lceil x\rceil + \left\lceil \frac{l - \lceil x\rceil}{\lceil x\rceil + 1}\right\rceil, k\lfloor x\rfloor + \left\lceil \frac{l - \lfloor x\rfloor}{\lfloor x\rfloor + 1}\right\rceil\right\}.$$

Proof. Let $P_l = (V, E)$ and $S \subseteq V$. As $\Delta(P_l) = 2$, $\sum_{s \in S} \deg_{P_l}(s) \leq 2|S|$. By Lemma 3.5, we get

$$m(P_l - S) \geq \frac{l - |S|}{1 + |S|}.$$

By Theorem 3.16, we obtain

$$\begin{aligned}\iota(kP_l) &= \min_{S \subseteq V} \{k|S| + m(P_l - S)\} \\ &\geq \min_{0 \leq x \leq l} \left\{kx + \frac{l - x}{x + 1}\right\} \\ &= \min\left\{k\lceil x\rceil + \left\lceil \frac{l - \lceil x\rceil}{\lceil x\rceil + 1}\right\rceil, k\lfloor x\rfloor + \left\lceil \frac{l - \lfloor x\rfloor}{\lfloor x\rfloor + 1}\right\rceil\right\},\end{aligned}$$

with $x = \max\left\{0, \frac{-k+\sqrt{k(l+1)}}{k}\right\}$.

For the upper bound, let $S, T \subseteq V$ such that $|S| = \lceil x\rceil$, $|T| = \lfloor x\rfloor$, and $m(P_l - S)$ and $m(P_l - T)$ are minimized. Then, we have

$$m(P_l - S) = \left\lceil \frac{l - \lceil x\rceil}{\lceil x\rceil + 1}\right\rceil \wedge m(P_l - T) = \left\lceil \frac{l - \lfloor x\rfloor}{\lfloor x\rfloor + 1}\right\rceil$$

Then, by Theorem 3.16 we obtain

$$\iota(kP_l) = \min_{S\subseteq V}\{k|S| + m(P_l - S)\} \leq \min\left\{k\lceil x\rceil + \left\lceil\frac{l-\lceil x\rceil}{\lceil x\rceil+1}\right\rceil, k\lfloor x\rfloor + \left\lceil\frac{l-\lfloor x\rfloor}{\lfloor x\rfloor+1}\right\rceil\right\}.$$

□

Second, we explore the value of the integrity for the union copies cycles.

Theorem 3.19. *Let kC_l be a disjoint union of k copies of the cycle C_l, and let $x = \max\left\{1, \sqrt{\frac{l}{k}}\right\}$. Then,*

$$\iota(kC_l) = \min\left\{l, k\lceil x\rceil + \left\lceil\frac{l-\lceil x\rceil}{\lceil x\rceil}\right\rceil, k\lfloor x\rfloor + \left\lceil\frac{l-\lfloor x\rfloor}{\lfloor x\rfloor}\right\rceil\right\}.$$

Proof. Let $C_l = (V, E)$ and $S \subseteq V$. Let $|S| \geq 1$ and $v \in S$. As $C_l - v = P_{l-1}$ and $\Delta(P_{l-1}) = 2$, $\sum_{s\in S}\deg_{P_{l-1}}(s) \leq 2|S|$. By Lemma 3.3, we get

$$c(C_l - S) = c(C_l - v - (S \setminus v)) = c(P_{l-1} - (S \setminus v)) \leq 1 + |S \setminus v| = |S|.$$

By Lemma 3.1, we obtain

$$m(C_l - S) \geq \frac{n - |S|}{|S|}.$$

By Theorem 3.16, we get

$$\begin{aligned}\iota(kC_l) &= \min_{S\subseteq V}\{k|S| + m(C_l - S)\}\\ &\geq \min\left\{l, \min_{1\leq x\leq l}\left\{kx + \frac{l-x}{x}\right\}\right\}\\ &= \min\left\{l, k\lceil x\rceil + \left\lceil\frac{l-\lceil x\rceil}{\lceil x\rceil}\right\rceil, k\lfloor x\rfloor + \left\lceil\frac{l-\lfloor x\rfloor}{\lfloor x\rfloor}\right\rceil\right\},\end{aligned}$$

with $x = \max\left\{1, \sqrt{\frac{l}{k}}\right\}$.

For the upper bound, let $S, T \subseteq V$ such that $|S| = \lceil x\rceil$, $|T| = \lfloor x\rfloor$, and $m(C_l - S)$ and $m(C_l - T)$ are minimized. Then, we have

$$m(C_l - \emptyset) = l \ \wedge\ m(C_l - S) = \left\lceil\frac{l-\lceil x\rceil}{\lceil x\rceil}\right\rceil \ \wedge\ m(C_l - T) = \left\lceil\frac{l-\lfloor x\rfloor}{\lfloor x\rfloor}\right\rceil$$

Then, by Theorem 3.16 we obtain

$$\iota(kC_l) = \min_{S\subseteq V}\{k|S| + m(C_l - S)\} \leq \min\left\{l, k\lceil x\rceil + \left\lceil\frac{l-\lceil x\rceil}{\lceil x\rceil}\right\rceil, k\lfloor x\rfloor + \left\lceil\frac{l-\lfloor x\rfloor}{\lfloor x\rfloor}\right\rceil\right\}.$$

□

Lastly, apart from the union of copies of paths and cycles, it is crucial to consider the union of copies of the complete graph.

Theorem 3.20. *Let kK_n be a disjoint union of k copies of the complete graph K_n. Then,*

$$\iota(kK_l) = n.$$

Proof. By Theorem 3.16, we get

$$\iota(kK_n) = \min_{S\subseteq V}\{k|S| + m(K_n - S)\} = \min_{S\subseteq V}\{k|S| + n - |S|\} = n.$$

□

3.2. Graph Joins

For the graph join, we find that we can evaluate the integrity based on the integrity values of its constituent parts. This property allows us to determine the resilience of the resulting graph by considering the integrity of the graphs being joined together. Moreover, this property is extendable to cases involving more than two graph joins, providing a convenient approach for evaluating the integrity of large-scale network structures.

Theorem 3.21. *[3] For any simple graphs G and H,*

$$\iota(G + H) = \min\{\iota(G) + |H|, \iota(H) + |G|\}.$$

Similar to Section 3.1, the following results are also useful in proving exact values and bounds for various graph families in Chapter 5. First, we determine the value of the integrity for graphs joined with the complete graph.

Corollary 3.22. *[3] For any simple graph H,*

$$\iota(K_n + H) = n + \iota(H).$$

Proof. By Theorem 2.40, $\iota(K_n) = |K_n| = n$. As $|K_n| = n$, by Theorem 3.21, we have

$$\iota(K_n + H) = \min\{\iota(K_n) + |H|, \iota(H) + |K_n|\} = n + \iota(H).$$

□

Lastly, apart from graphs joined with the complete graph, graphs joined with the empty graph are also crucial to consider.

Corollary 3.23. *For any simple graph H,*

$$\iota(\overline{K}_n + H) = \min\{1 + |H|, n + \iota(H)\}.$$

Proof. By Theorem 2.40, $\iota(\overline{K}_n) = 1$. As $|\overline{K}_n| = n$, by Theorem 3.21, we get

$$\iota(K_n + H) = \min\{\iota(K_n) + |H|, \iota(H) + |K_n|\} = \min\{1 + |H|, n + \iota(H)\}.$$

□

3.3. Spectral Bounds

Alon et al. [2] introduced a graph property that exhibits similarities with the independence number and provides bounds for the integrity measure. This property can be bounded using spectral graph theory, which offers new approaches and techniques for studying the integrity of graphs.

Definition 3.24. *[2] For graph G, let $z(G)$ denote the largest integer z such that there are two disjoint sets of vertices in G, each of size z, with no edge between them.*

Proposition 3.25. *[2] For every graph G on n vertices,*

$$n - 2z(G) \leq \iota(G) \leq n - z(G).$$

Proof. Let $G = (V, E)$ be a graph. For the upper bound, let $A, B \subseteq V$ be two disjoint sets of size z with no edge between then. Let $S = V \setminus (A \cup B)$. Then $G - S$ consists only of vertices from A and B. As there is no edge between A and B, the connected components in $G - S$ are either vertices contained only in A or B. Hence the size of a component is at most z. Therefore, we have

$$\iota(G) \leq |S| + m(G - S) = |S| + z = (n - 2z) + z = n - z.$$

For the lower bound, let $z = z(G)$ and let $S \subseteq V$ be of size ζ such that the largest size of a connected component in $G - S$ is $\eta = m(G - S)$. Choose ζ and η such that $\iota(G) = \zeta + \eta$. Let $G - S$ consist of the connected components $C_1, C_2, \dots, C_l$ with sizes $\eta = c_1 \geq c_2 \geq \dots \geq c_l$ and $l \in \mathbb{Z}_{\geq 0}$. Note that, since $n - \zeta = \sum_{i=1}^{l} c_i$ and $\eta = c_1$, we have $n - \iota(G) = (n - \zeta) - \eta = \sum_{i=2}^{l} c_i$. To prove the lower bound it suffices to show that $\sum_{i=2}^{l} c_i \leq 2z$. Also note that there are no edges between C_i and C_j for any $i \neq j$.

We have two cases to prove this $2z$ upper bound. Case 1 is when $c_1 \geq z + 1$. Then by the maximality of z the size of $C_2 \cup \dots \cup C_l$ is at most z and the upper bound holds. Case 2 is when $c_1 \leq z$. For the sake of contradiction, we assume that $\sum_{i=2}^{l} c_i \geq 2z + 1$. Let $2 \leq k \leq l$ be the largest index for which $c_k + \dots + c_l \geq z + 1$. As k is the largest index, we have $c_k + \dots + c_{l-1} \leq z$. This implies that $c_k + \dots + c_l \leq z + c_l$. Since $c_l \leq c_1$, it follows that $c_k + \dots + c_l \leq z + c_1$. Therefore $c_2 + \dots + c_{k-1} \geq 2z + 1 - (z + c_1) = z + 1 - c_1$. Let $X = C_1 \cup \dots \cup C_{k-1}$ and $Y = C_k \cup \dots \cup C_l$. Then both X and Y have a size of at least $z + 1$, which is a contradiction since they have no edge between them.

□

Proposition 3.26. *[2] For any d-regular graph G with n vertices and eigenvalue λ,*

$$z(G) \leq \frac{\lambda n}{d + \lambda}.$$

Utilizing the bound from Proposition 3.25, we can find the following lower bound for the integrity.

Theorem 3.27. *[2] For any d-regular graph G with n vertices and eigenvalue λ,*

$$\iota(G) \geq n \frac{d - \lambda}{d + \lambda}.$$

By utilizing the known eigenvalues of strongly regular graphs and combining them with the lower bound presented in Theorem 3.27, we obtain the following lower bound.

Lemma 3.28. *[32] Let G be a $\mathrm{srg}(n, d, a, b)$ graph, with $b \neq 0$, and let $D = (a - b)^2 + 4(d - b) > 0$. Then G has three distinct eigenvalues $d > \lambda_1 > \lambda_2$, where*

$$\lambda_{1,2} = \frac{1}{2}\left((a - b) \pm \sqrt{D}\right).$$

Corollary 3.29. *Let G be a $\mathrm{srg}(n, d, a, b)$ graph, with $b \neq 0$, and let $D = (a - b)^2 + 4(d - b) > 0$. Then,*

$$\iota(G) \geq n \frac{b - a + 2d - \sqrt{D}}{a - b + 2d + \sqrt{D}}.$$

Proof. By Theorem 3.27 and Lemma 3.28, we get

$$\iota(P(q)) \geq n \frac{d - \lambda}{d + \lambda} = n \frac{b - a + 2d - \sqrt{D}}{a - b + 2d + \sqrt{D}}.$$

□

4

Integer Linear Programming

In this chapter, we introduce our developed Integer Linear Programming (ILP) models. These models are tailored to tackle the computational challenge of efficiently determining the properties' values compared to exhaustive input evaluation. Leveraging the capabilities of ILP, our objective is to expedite the property evaluation process and establish a reliable method for calculating accurate values.

4.1. Largest size of a Connected Component

Firstly, we examined one of the properties used in the definition of integrity: the largest size of a connected component. This initial model was crucial to access the feasibility of constructing a model for integrity.

Problem definition: The problem is to determine the largest size of a connected component, denoted by $m(G)$.

Input: Let $G = (V, E)$ be an undirected graph. Let C_i denote the i-th component in G.

Variable definition: Let

$$c_v^i = \begin{cases} 1 & \text{if } v \in C_i \\ 0 & \text{otherwise} \end{cases} \quad 1 \leq i \leq n.$$

Model: Let

$$\begin{aligned} &\min M, \\ &\text{s.t.} \sum_{v \in V} c_v^i \leq M \quad 1 \leq i \leq n, && (4.1) \\ &\sum_{i=1}^{n} c_v^i = 1 \quad v \in V, && (4.2) \\ &c_u^i = c_v^i \qquad uv \in E, 1 \leq i \leq n, && (4.3) \\ &c_v^i \in \{0, 1\} \quad v \in V, 1 \leq i \leq n, \\ &M \in \mathbb{Z}_{\geq 0}. \end{aligned}$$

Model description: The goal is to minimize M, where Constraint (4.1) ensures that M is bigger than the size of each component. There are at most n components, as each vertex can be in its own component. Constraint (4.2) forces that each vertex can only be in exactly 1 component. Finally, Constraint (4.3) forces that if two vertices are adjacent then they should be in the same component.

4.2. Vertex Integrity of a Graph

Having confirmed the feasibility of constructing an ILP for the largest size of a connected component, we proceed to explore the integrity parameter.

Problem definition: The problem is to determine the vertex integrity of a Graph, denoted by $\iota(G)$.

Input: Let $G = (V, E)$ be an undirected graph and $S \subseteq V$ be a set of vertices. Let C_i denote the i-th component in G.

Variable definition: Let

$$s_v = \begin{cases} 1 & \text{if } v \in S \\ 0 & \text{otherwise} \end{cases}$$

$$c_v^i = \begin{cases} 1 & \text{if } v \in C_i \\ 0 & \text{otherwise} \end{cases} \quad 1 \leq i \leq n.$$

Model: Let

$$\begin{aligned}
&\min \sum_{v \in V} s_v + M, \\
&\text{s.t.} \sum_{v \in V} c_v^i \leq M && 1 \leq i \leq n, && (4.4)\\
&\sum_{i=1}^{n} c_v^i = 1 - s_v && v \in V, && (4.5)\\
&c_u^i - c_v^i \leq s_u + s_v && uv \in E, 1 \leq i \leq n, && (4.6)\\
&c_v^i - c_u^i \leq s_u + s_v && uv \in E, 1 \leq i \leq n, && (4.7)\\
&s_v, c_v^i \in \{0, 1\} && v \in V, 1 \leq i \leq n, \\
&M \in \mathbb{Z}_{\geq 0}.
\end{aligned}$$

Model description: The goal is to minimize the size of S plus M, where Constraint (4.4) ensures that M is bigger than the size of each component. There are at most n components, as each vertex can be in its own component. Constraint (4.5) forces that each vertex can only be in exactly 1 component or 0 components if the vertex is in S. Finally, Constraints (4.6) and (4.7) force that if two vertices are adjacent and both are not in S then they should be in the same component.

4.2.1. Additional Constraints

We implemented the ILP-model for the integrity in Python [31] with the Gurobi software [21] (see Appendix B for the code). In order to enhance the model, we attempted to add additional constraints.

Initially, we explored removing the symmetry of the constraints by adding the requirement that the size of the first component should be larger than the second component, and so on. This resulted in the following constraint

$$\sum_{v \in V} c_v^i \geq \sum_{v \in V} c_v^{i+1} \quad 1 \leq i \leq n-1. \tag{4.8}$$

However, after testing, we observed that Constraint (4.8) led to a noticeable increase in solving time without providing any significant improvement. Therefore, we decided to exclude this constraint from the model.

Example 4.1. *Consider the Paley graph with 41 vertices, where the integrity is 37. When computing the integrity without the inclusion of Constraint* (4.8)*, the computation time is 31.26 seconds (98.60 work units). However, when this constraint is included, the computation time increases significantly to 156.85 seconds (581.85 work units).*

Furthermore, we explored the possibility of limiting the size of S and the largest size of a connected component M. This led to the following two constraints, where X and Y are chosen values,

$$M \leq Y \tag{4.9}$$

$$\sum_{v \in V} s_v \geq X. \tag{4.10}$$

These constraints were added to graphs where we knew the desired size of S or the desired value of M. However, Constraint (4.9) did not improve the solving time. On the other hand, Constraint (4.10) improved the solving time in some cases. Specifically, when X was equal to or larger than the desired size of S that would provide the correct value for the integrity, it improved the solving time. However, in cases where the size of X was considerably smaller than the desired size of S, the addition of this constraint led to a noticeable increase in the solving time compared to not including it.

Example 4.2. *Consider the Paley graph with 49 vertices, where the integrity is 49 and the desired size of set S is 42. When computing the integrity without the inclusion of Constraint* (4.10)*, the computation time is 79.54 seconds (215.79 work units). However, when applying the constraint with $Y = 42$, the computation time significantly decreases to only 2.47 seconds (5.01 work units). On the other hand, when Constraint* (4.10) *is used with $Y = 32$, the solving time is increased to 623.62 seconds (2809.75 work units).*

4.3. Largest size of an empty balanced bipartite subgraph

Problem definition: The problem is to determine the value of $z(G)$ (see Definition 3.24).

Input: Let $G = (V, E)$ be an undirected graph and $A, B \subseteq V$ be two sets of vertices.

Variable definition: Let

$$a_v = \begin{cases} 1 & \text{if } v \in A \\ 0 & \text{otherwise} \end{cases}$$

$$b_v = \begin{cases} 1 & \text{if } v \in B \\ 0 & \text{otherwise} \end{cases}.$$

Model: Let

$$\begin{aligned} &\max z, \\ &\text{s.t.} \sum_{v \in V} a_v = z, && (4.11) \\ &\sum_{v \in V} b_v = z, && (4.12) \\ &a_u + b_v \leq 1 \quad uv \in E, && (4.13) \\ &a_v + b_u \leq 1 \quad uv \in E, && (4.14) \\ &a_v, b_v \in \{0, 1\} \quad v \in V, \\ &z \in \mathbb{Z}_{\geq 0}. \end{aligned}$$

Model description: The goal is to maximize z, subject to Constraints (4.11) and (4.11), which ensure that the sizes of A and B are both equal to z. Finally, Constraints (4.13) and (4.14) force that no edges are between vertices in set A and B.

4.4. Computational Complexity

A fundamental aspect of this study is the computational complexity of determining whether the integrity of a graph is smaller or equal to p, given the graph G and an integer p. Drange et al. [13] provide a comprehensive overview of the computational complexity of this problem and related aspects. Clark et al. [12] have established the problem as NP-complete, indicating that finding an exact solution using a polynomial-time algorithm is currently infeasible. As the size of the graph increases, the computational resources required to compute the integrity grow exponentially. Consequently, for larger graphs, the precise calculation of integrity becomes impractical. To address the computational challenge posed by the NP-completeness of the integrity problem, alternative approaches are necessary to find efficient solutions. One such approach is the use of the ILP (Integer Linear Programming) model. By formulating the problem as an ILP, we can take advantage of the efficient algorithms and techniques developed for solving linear programming problems. The ILP model offers a more structured and systematic approach to approximate the integrity values in a more efficient manner compared to the brute force method. However, it is important to note that even with the utilization of the ILP model, the problem remains NP-complete. Nonetheless, the ILP model provides a valuable tool for tackling the integrity problem and facilitating more manageable computations.

However, there are certain graph families where the problem is not NP-complete, as mentioned by Bagga et al. [3], and implied by the results from Kratsch et al. [24].

Furthermore, the concept of vertex integrity has been expanded by incorporating a weight function that assigns weights to individual vertices. Drange et al. [13] have investigated the computational complexity of determining the existence of a subset X of vertices satisfying the condition that the weight of X, combined with the weight of the heaviest component in $G - X$, is at most p. This problem takes as inputs a graph G with n vertices, a weight function $\omega : V(G) \to \mathbb{N}$, and an integer p. The study revealed that this problem is only NP-complete for co-bipartite graphs, even when each vertex has a weight of 1.

5

Integrity of Families of Graphs

In this chapter, we apply the concepts described in Chapter 3 to explore the integrity of various graph families. To determine the most suitable concepts to use, we employ the ILP-model for the integrity, as described in Chapter 4. Using this model, we are able to compute the integrity values for several graph families. Through the analysis of these values, we have identified patterns that provide insights into which concepts should be employed to establish exact values or bounds for the integrity.

5.1. Glued Paths

Definition 5.1. *[27] A **Glued Path**, denoted by $GP_{k,l}$ and depicted in Figure 5.1, consists of a vertex u with $k > 2$ internally disjoint paths of length $l-1$, denoted by P_{l-1}^{i}, joined at u. The vertex u is called the root vertex.*

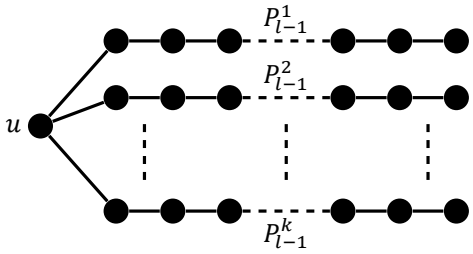


Figure 5.1: General form of a Glued Path.

Theorem 5.2. *Let $GP_{k,l}$ be a Glued Path, and let $x = \max\left\{0, \frac{-k+\sqrt{kl}}{k}\right\}$. Then,*

$$\iota(GP_{k,l}) = 1 + \min\left\{k\lceil x\rceil + \left\lceil \frac{l-1-\lceil x\rceil}{\lceil x\rceil+1}\right\rceil, k\lfloor x\rfloor + \left\lceil \frac{l-1-\lfloor x\rfloor}{\lfloor x\rfloor+1}\right\rceil\right\}.$$

Proof. Let $GP_{k,l} = (V, E)$, with u as the root vertex, and let $T = \{u\}$. As $GP_{k,l}$ is connected, by Corollary 3.10, we get

$$\begin{aligned}\iota(GP_{k,l}) &= 1 + \min\left\{\min_{v\in T}\iota(GP_{k,l}-v), \min_{v\in T^c}\min_{S\subseteq T^c\setminus v}\left\{|S| + m(GP_{k,l}-v-S)\right\}\right\}\\ &= 1 + \min\left\{\min_{v\in T}\iota(GP_{k,l}-v), -1 + \min_{\substack{S\subseteq T^c\\ |S|\geq 1}}\left\{|S| + m(GP_{k,l}-S)\right\}\right\}.\end{aligned}$$

As for any $t \in T$, $GP_{k,l} - t \cong kP_{l-1}$, we have

$$\iota(kP_{l-1}) = \min_{v \in T} \iota(GP_{k,l} - v).$$

For any $S \subseteq T^c$, $\Delta(GP_{k,l}, S) \leq 2$. By Corollary 3.7 and Theorem 3.18, we obtain

$$\begin{aligned} -1 + \min_{\substack{S \subseteq T^c \\ |S| \geq 1}} \{|S| + m(GP_{k,l} - S)\} &\geq -1 + \min_{\substack{S \subseteq T^c \\ |S| \geq 1}} \left\{|S| + \frac{|V| - |S|}{|S| + 1}\right\} \\ &\geq \left\lceil 2\sqrt{|V| + 1} \right\rceil - 3 \\ &= \left\lceil 2\sqrt{k(l-1) + 2} \right\rceil - 3 \\ &\geq \iota(kP_{l-1}). \end{aligned}$$

Hence we obtain

$$\iota(GP_{k,l}) = 1 + \iota(kP_{l-1}).$$

□

5.2. Generalized Theta Graphs

Definition 5.3. *[28] A **Theta graph**, consists of a pair of vertices u, v with 3 internally disjoint paths joining u to v.*

Definition 5.4. *[28] A **generalized Theta graph**, denoted by $\Theta_{k,l}$ and depicted in Figure 5.2, consists of a pair of vertices u, v with $k > 2$ internally disjoint paths of length $l - 2$, denoted by P_{l-2}^i, joining u to v. The vertices u and v are called the root vertices.*

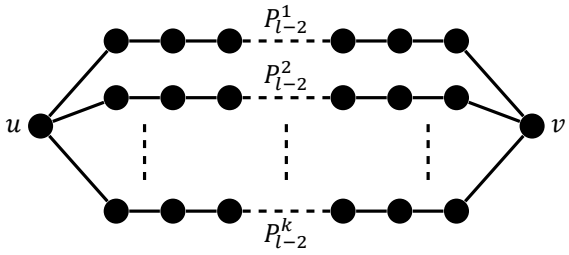


Figure 5.2: General form of a Theta graph.

Theorem 5.5. *Let $\Theta_{k,l}$ be a generalized Theta graph, and let $x = \max\left\{0, \frac{-k+\sqrt{k(l-1)}}{k}\right\}$. Then,*

$$\iota(\Theta_{k,l}) = 2 + \min\left\{k\lceil x \rceil + \left\lceil \frac{l - 2 - \lceil x \rceil}{\lceil x \rceil + 1} \right\rceil, k\lfloor x \rfloor + \left\lceil \frac{l - 2 - \lfloor x \rfloor}{\lfloor x \rfloor + 1} \right\rceil\right\}.$$

Proof. Let $\Theta_{k,l} = (V, E)$, with u and v as the root vertices, and let $T = \{u, v\}$. As $\Theta_{k,l}$ is connected, by Corollary 3.10, we get

$$\begin{aligned} \iota(\Theta_{k,l}) &= 1 + \min\left\{\min_{v \in T} \iota(\Theta_{k,l} - v), \min_{v \in T^c} \min_{S \subseteq T^c \setminus v} \{|S| + m(\Theta_{k,l} - v - S)\}\right\} \\ &= 1 + \min\left\{\min_{v \in T} \iota(\Theta_{k,l} - v), -1 + \min_{\substack{S \subseteq T^c \\ |S| \geq 1}} \{|S| + m(\Theta_{k,l} - S)\}\right\}. \end{aligned}$$

As for any $t \in T$, $\Theta_{k,l} - t \cong GP_{k,l}$, we have

$$\iota(GP_{k,l}) = \min_{v \in T} \iota(\Theta_{k,l} - v).$$

For any $S \subseteq T^c$, $\Delta(\Theta_{k,l}, S) \leq 2$. By Corollary 3.7 and Theorem 3.18, we obtain

$$\begin{aligned} -1 + \min_{\substack{S \subseteq T^c \\ |S| \geq 1}} \{|S| + m(\Theta_{k,l} - S)\} &\geq -1 + \min_{\substack{S \subseteq T^c \\ |S| \geq 1}} \left\{ |S| + \frac{|V| - |S|}{|S| + 1} \right\} \\ &\geq \left\lceil 2\sqrt{|V| + 1} \right\rceil - 3 \\ &= \left\lceil 2\sqrt{k(l-2) + 3} \right\rceil - 3 \\ &\geq \iota(GP_{k,l-1}). \end{aligned}$$

Hence we obtain

$$\iota(\Theta_{k,l}) = 1 + \iota(GP_{k,l-1}).$$

□

5.3. (Double) Cone Graphs

Definition 5.6. *The **Double Cone graph**, denoted by $K_1 + 2C_n$, is a graph obtained by joining two disjoint cycles C_n of order n with a single point K_1. Figure 5.3 depicts the Double Cone graph $K_1 + C_6$.*

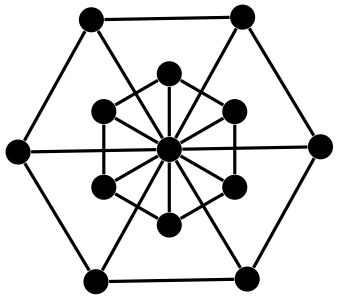


Figure 5.3: Graph drawing of the Double Cone graph $K_1 + 2C_6$, the graph join of the complete graph K_1 and two disjoint copies of the cycle C_6.

Theorem 5.7. *Let $K_1 + 2C_n$ be a Double Cone graph, and let $x = \max\left\{1, \sqrt{\frac{n}{2}}\right\}$. Then,*

$$\iota(K_1 + 2C_n) = 1 + \min\left\{n, 2\lceil x \rceil + \left\lceil \frac{n - \lceil x \rceil}{\lceil x \rceil} \right\rceil, 2\lfloor x \rfloor + \left\lceil \frac{n - \lfloor x \rfloor}{\lfloor x \rfloor} \right\rceil \right\}.$$

Proof. By Corollary 3.22 and Theorem 3.19, we have

$$\iota(K_1 + 2C_n) = 1 + \iota(2C_n).$$

□

Definition 5.8. *[10] The **Cone graph**, denoted by $\overline{K}_n + C_m$ and also known as the **generalized Wheel graph**, is a graph obtained by joining a cycle C_m of order m with the empty graph $\overline{K}_n$ of order n. Figure 5.4 depicts the Cone graph $\overline{K}_3 + C_3$.*

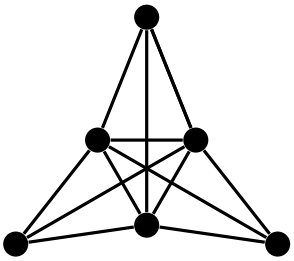


Figure 5.4: Graph drawing of the Cone graph $\overline{K}_3 + C_3$, the graph join of the empty graph $\overline{K}_3$ and the cycle C_3.

Theorem 5.9. *Let $\overline{K}_n + C_m$ be a Cone graph. Then,*

$$\iota(\overline{K}_n + C_m) = \min\left\{1 + m, \left\lceil 2\sqrt{m} \right\rceil + n - 1\right\}.$$

Proof. By Corollary 3.23 and Theorem 2.40, we get

$$\iota(\overline{K}_n + C_m) = \min\{1 + m, n + \iota(C_m)\}.$$

□

5.4. Fan Graphs

Definition 5.10. *The **Fan graph**, denoted by $F_{n,m}$, is a graph obtained by joining the path P_m of order m with the empty graph $\overline{K}_n$ of order n. Therefore $F_{n,m} \cong \overline{K}_n + P_m$. Figure 5.5 depicts the Fan graph $F_{3,4}$.*

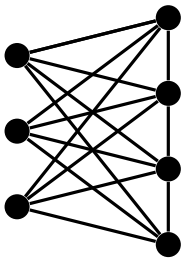


Figure 5.5: Graph drawing of the Fan graph $F_{3,4}$, the graph join of the empty graph $\overline{K}_3$ and the path P_4.

Theorem 5.11. *Let $F_{n,m}$ be a Fan graph. Then,*

$$\iota(F_{n,m}) = \min\left\{1 + m, \left\lceil 2\sqrt{m+1} \right\rceil + n - 2\right\}.$$

Proof. By Corollary 3.23 and Theorem 2.40, we obtain

$$\iota(F_{n,m}) = \iota(\overline{K}_n + P_m) = \min\{1 + m, n + \iota(P_m)\}.$$

□

5.5. Lollipop Graphs

Definition 5.12. *[11, 29] A **bridge** of a connected graph is a graph edge whose removal disconnects the graph. [22] More generally, a bridge is an edge of a not-necessarily-connected graph G whose removal increases the number of components of G.*

Definition 5.13. *[19] The **Lollipop graph**, denoted by L_n^l and with $n > 2$, is a graph connecting the complete graph K_n and the path P_l by a bridge. Figure 5.6 depicts the Lollipop graph L_5^3.*

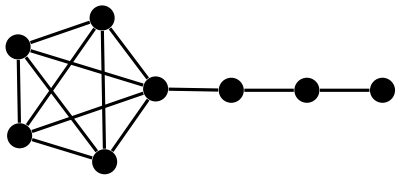


Figure 5.6: Graph drawing of the Lollipop graph L_5^3, the graph connecting the complete graph K_5 and the path P_3 by a bridge.

Theorem 5.14. *Let L_n^l be a Lollipop graph and let $k = \max\left\{0, \left\lfloor\sqrt{l+n}\right\rfloor - n\right\}$. Then,*

$$\iota(L_n^l) = \left\lfloor\frac{l-k}{n+k}\right\rfloor + n + k.$$

Proof. Let $L_n^l = (V, E)$. As K_n is a subgraph of L_n^l, $\omega(L_n^l) \geq n$. By Proposition 3.13, we have

$$\iota(L_n^l) = \min_{\substack{S\subseteq V \\ m(L_n^l - S)\geq n-1}} \left\{|S| + m(L_n^l - S)\right\}.$$

Let $S \subseteq V$, and let $0 \leq k \leq l+1$. Label the vertex in K_n connected to the bridge as 0, and label the vertices along the path P_l starting from this vertex as $1, 2, \ldots, l$, where the i-th vertex is adjacent to the $(i+1)$-th vertex for all $i = 0, \ldots, l-1$.

We will remove the k-th vertex, so the k-th vertex is in S. If $m(L_n^l - S) = (n-1) + k$, we have

$$|S| \geq \left\lfloor\frac{l-k}{(n-1)+k+1}\right\rfloor + 1 = \left\lfloor\frac{l-k}{n+k}\right\rfloor + 1.$$

By Proposition 3.13, we obtain

$$\begin{aligned}\iota(L_n^l) &= \min_{\substack{S\subseteq V \\ m(L_n^l - S)\geq n-1}} \left\{|S| + m(L_n^l - S)\right\} \\ &\geq \min_{0\leq k\leq l}\left\{\left\lfloor\frac{l-k}{n+k}\right\rfloor + 1 + (n-1) + k\right\} \\ &= \left\lfloor\frac{l-k}{n+k}\right\rfloor + n + k,\end{aligned}$$

with $k = \max\left\{0, \left\lfloor\sqrt{l+n}\right\rfloor - n\right\}$.

For the upper bound, let $S \subseteq V$ such that $m(L_n^l - S) = (n-1) + k$, and the size of S is minimized. Then, we have

$$|S| = \left\lfloor\frac{l-k}{n+k}\right\rfloor + 1$$

Then, by Proposition 3.13 we get

$$\iota(L_n^l) = \min_{\substack{S\subseteq V \\ m(L_n^l - S)\geq n-1}} \left\{|S| + m(L_n^l - S)\right\} \leq \left\lfloor\frac{l-k}{n+k}\right\rfloor + n + k.$$

□

5.6. Generalized Barbell Graphs

Definition 5.15. *[19] The **Barbell graph**, denoted by B_n and with $n > 2$, is a graph connecting two copies of a complete graph K_n by a bridge.*

Definition 5.16. *The **generalized Barbell graph**, denoted by B_n^l and with $n > 2$, is a graph connecting the path P_l and on both sides a complete graphs K_n by bridges. If the length of the path is 0, we have $B_n^0 \cong B_n$. Figure 5.7 depicts the Barbell graph B_6^4.*

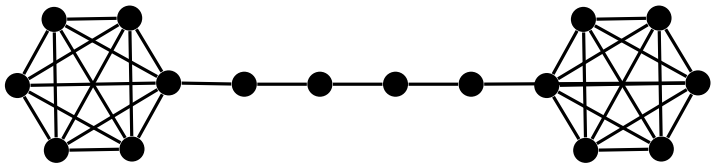


Figure 5.7: Graph drawing of the Barbell graph B_6^4, the graph connecting the path P_4 and on both sides a complete graphs K_6 by bridges.

Theorem 5.17. *Let B_n^l be a generalized Barbell graph and let $k = \max\left\{0, \left\lfloor\sqrt{l+2n}\right\rfloor - n\right\}$. Then,*

$$\iota(B_n^l) = \left\lfloor \frac{l-2k}{n+k} \right\rfloor + n + k + 1.$$

Proof. Let $B_n^l = (V, E)$. As K_n is a subgraph of B_n^l, $\omega(B_n^l) \geq n$. By Proposition 3.13, we have

$$\iota(B_n^l) = \min_{\substack{S \subseteq V \\ m(B_n^l - S) \geq n-1}} \left\{ |S| + m(B_n^l - S) \right\}.$$

Let $S \subseteq V$, and let $0 \leq k \leq l+1$. Label the vertex in one of the K_n connected to one of the bridges as 0, and label the vertices along the path P_l starting from this vertex as $1, 2, \ldots, l+1$, where the i-th vertex is adjacent to the $(i+1)$th vertex for all $i = 0, \ldots, l$. Note that the $(l+2)$-th vertex is in the other K_n connected to the other bridge.

We will remove the k-th and the $(l+1-k)$-th vertex, so the k-th and $(l+1-k)$-th vertices are in S. If $m(B_n^l - S) = (n-1) + k$, we have

$$|S| \geq \left\lfloor \frac{l-2k}{(n-1)+k+1} \right\rfloor + 2 = \left\lfloor \frac{l-2k}{n+k} \right\rfloor + 2.$$

By Proposition 3.13, we obtain

$$\begin{aligned} \iota(B_n^l) &= \min_{\substack{S \subseteq V \\ m(B_n^l - S) \geq n-1}} \left\{ |S| + m(B_n^l - S) \right\} \\ &\geq \min_{0 \leq k \leq l+1} \left\{ \left\lfloor \frac{l-2k}{n+k} \right\rfloor + 2 + (n-1) + k \right\} \\ &= \left\lfloor \frac{l-2k}{n+k} \right\rfloor + n + k + 1, \end{aligned}$$

with $k = \max\left\{0, \left\lfloor\sqrt{l+2n}\right\rfloor - n\right\}$.

For the upper bound, let $S \subseteq V$ such that $m(B_n^l - S) = (n-1) + k$, and the size of S is minimized. Then, we have

$$|S| = \left\lfloor \frac{l-2k}{n+k} \right\rfloor + 1$$

Then, by Proposition 3.13 we get

$$\iota(B_n^l) = \min_{\substack{S\subseteq V \\ m(B_n^l - S)\geq n-1}} \{|S| + m(B_n^l - S)\} \leq \left\lfloor \frac{l-2k}{n+k} \right\rfloor + n + k + 1.$$

□

5.7. (Dutch) Windmill Graphs

Definition 5.18. *[17] The **Dutch Windmill graph**, denoted by D_3^m and also known as the **Friendship Graph**, is a graph obtained by taking m copies of the cycle C_3 with a vertex u in common, where u is called the root vertex. This definition can be extended to D_n^m, consisting of m copies of C_n. Figure 5.8 depicts the Dutch Windmill graph D_4^3.*

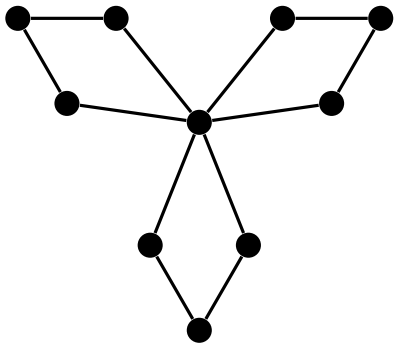


Figure 5.8: Graph drawing of the Dutch Windmill graph D_4^3, the graph graph obtained by taking 3 copies of the cycle C_4 with a vertex in common.

Theorem 5.19. *Let D_n^m be a Dutch Windmill graph, and let $x = \max\left\{0, \frac{-m+\sqrt{mn}}{m}\right\}$. Then,*

$$\iota(D_n^m) = 1 + \min\left\{ m\lceil x\rceil + \left\lceil \frac{n-1-\lceil x\rceil}{\lceil x\rceil + 1} \right\rceil, m\lfloor x\rfloor + \left\lceil \frac{n-1-\lfloor x\rfloor}{\lfloor x\rfloor + 1} \right\rceil \right\}.$$

Proof. Let $D_n^m = (V, E)$, with u and v as the root vertices, and let $T = \{u, v\}$. As D_n^m is connected, by Corollary 3.10, we get

$$\begin{aligned}\iota(D_n^m) &= 1 + \min\left\{ \min_{v\in T} \iota(D_n^m - v), \min_{v\in T^c} \min_{S\subseteq T^c\setminus v} \{|S| + m(D_n^m - v - S)\} \right\} \\ &= 1 + \min\left\{ \min_{v\in T} \iota(D_n^m - v), -1 + \min_{\substack{S\subseteq T^c \\ |S|\geq 1}} \{|S| + m(D_n^m - S)\} \right\}.\end{aligned}$$

As for any $t \in T$, $D_n^m - t \cong mP_{n-1}$, we have

$$\iota(mP_{n-1}) = \min_{v\in T} \iota(D_n^m - v).$$

For any $S \subseteq T^c$, $\Delta(D_n^m, S) \leq 2$. By Corollary 3.7 and Theorem 3.18, we obtain

$$\begin{aligned}-1 + \min_{\substack{S\subseteq T^c \\ |S|\geq 1}} \{|S| + m(D_n^m - S)\} &\geq -1 + \min_{\substack{S\subseteq T^c \\ |S|\geq 1}} \left\{ |S| + \frac{|V| - |S|}{|S| + 1} \right\} \\ &\geq \left\lceil 2\sqrt{|V| + 1} \right\rceil - 3 \\ &= \left\lceil 2\sqrt{m(n-1) + 2} \right\rceil - 3 \\ &\geq \iota(mP_{n-1}).\end{aligned}$$

Hence we obtain

$$\iota(D_n^m) = 1 + \iota(mP_{n-1}).$$

□

Definition 5.20. *[17] The **Windmill graph**, denoted by W_n^m, is a graph obtained by taking m copies of the complete graph K_n with a vertex u in common, where u is called the root vertex. Therefore $W_n^m \cong K_1 + mK_{n-1}$. Figure 5.9 depicts the Windmill graph W_5^4.*

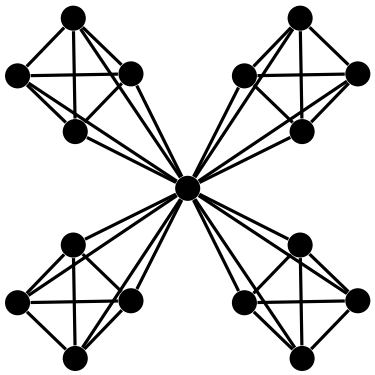


Figure 5.9: Graph drawing of the Windmill graph W_5^4, the graph graph obtained by taking 4 copies of the complete graph K_5 with a vertex in common.

Theorem 5.21. *Let W_n^m be a Windmill graph. Then,*

$$\iota(W_n^m) = n$$

Proof. By Corollary 3.22 and Theorem 3.20, we have

$$\iota(W_n^m) = \iota(K_1 + mK_{n-1}) = 1 + \iota(mK_{n-1}) = 1 + (n-1) = n.$$

□

5.8. Paley Graphs

Definition 5.22. *The **Paley graph**, denoted by $P(q)$ with q as a prime power, is a graph on q vertices. In this graph, two vertices are adjacent if and only if their difference is a square in the finite field $\mathrm{GF}(q)$. Figure 5.10 depicts the Paley graph $P(13)$.*

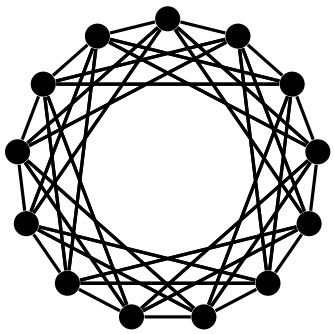


Figure 5.10: Graph drawing of the Paley graph $P(13)$ with 13 vertices.

Proposition 5.23. *[20] The Paley graph $P(q)$ is strongly regular with parameters $(q, \frac{q-1}{2}, \frac{q-5}{4}, \frac{q-1}{4})$.*

Theorem 5.24. *For any Paley graph $P(q)$,*

$$\iota(P(q)) \geq \frac{q\sqrt{q}}{\sqrt{q}+2}.$$

Proof. By Proposition 5.23 and Theorem 3.29, we get

$$\iota(P(q)) \geq \frac{q\sqrt{q}}{\sqrt{q}+2}.$$

□

Proposition 5.25. *[9] For any Paley graph $P(q)$, with q an even power of a prime,*

$$\chi(P(q)) = \omega(P(q)) = \sqrt{q}.$$

Theorem 5.26. *For any Paley graph $P(q)$, with q an even power of a prime,*

$$\alpha(P(q)) = \sqrt{q}.$$

Proof. As $P(q)$ is self-complementary, by Proposition 5.25, we obtain

$$\alpha(P(q)) = \omega(P(q)^c) = \omega(P(q)) = \sqrt{q}.$$

□

Corollary 5.27. *For any Paley graph $P(q)$, with q an even power of a prime,*

$$\iota(P(q)) \leq q - \sqrt{q} + 1.$$

Proof. Substituting Theorem 5.26 in Proposition 3.14 gives

$$\iota(P(q)) \leq q - \alpha(G) + 1 = q - \sqrt{q} + 1.$$

□

5.9. Kneser Graphs

Definition 5.28. *[7] The **Odd graph**, denoted by $O(n)$, is a graph whose vertices represent the $(n-1)$-subsets of $\{1, \ldots, 2n-1\}$, in which two vertices are connected if and only if they correspond to disjoint subsets. Therefore O(n) has $\binom{2n-1}{n-1}$ vertices and is n-regular.*

Definition 5.29. *[26] The **Kneser graph**, denoted by $K(n,k)$ is a generalization of the Odd graph. Its vertices represent the k-subsets of $\{1, \ldots, n\}$, in which two vertices are connected if and only if they correspond to disjoint subsets. Therefore K(n, k) has $\binom{n}{k}$ vertices and is $\binom{n-k}{k}$-regular. Figure 5.11 depicts the Kneser graph $K(5,2)$.*

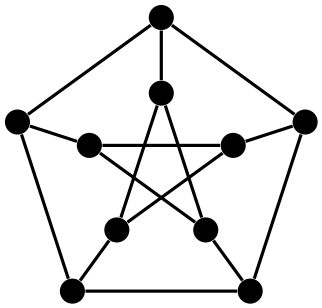


Figure 5.11: Graph drawing of the Kneser graph $K(5,2)$, also known as the Petersen graph, with 10 vertices.

Definition 5.30. *[4] The **Ladder Rung graph**, denoted by kP_2, consists of the graph union of k copies of the path P_2.*

Remark 5.31. *For any two positive integers k and n, we have special cases*

$$K(n,1) = K_n,$$
$$K(n,n) = K_1,$$
$$K(n,n-1) = \overline{K}_n,$$
$$K(2n,n) = \frac{1}{2}\binom{n}{k}P_2,$$
$$K(2n-1,n-1) = O(n).$$

Remark 5.32. *For any two positive integers k and n, the integrity of $K(n,k)$ is (still) unknown only in the case where $1 < k < \frac{n}{2}$.*

Proposition 5.33. *[1] For any non-empty Kneser graph $K(n,k)$,*

$$\alpha(K(n,k)) = \binom{n-1}{k-1}.$$

Corollary 5.34. *For any non-empty Kneser graph $K(n,k)$,*

$$\iota(K(n,k)) \leq \binom{n}{k} - \binom{n-1}{k-1} + 1.$$

Proof. Substituting Proposition 5.33 in Proposition 3.14 gives

$$\iota(K(n,k)) \leq \binom{n}{k} - \alpha(G) + 1 = \binom{n}{k} - \binom{n-1}{k-1} + 1.$$

□

6

Conclusion and Open Problems

The goal of this thesis was to investigate the integrity of specific graph families. In Chapter 3, we introduced general methods and concepts that enable us to establish boundaries or determine exact values of the integrity. Chapter 4 described our ILP models for evaluating various properties of a graph, including the integrity. We applied our integrity ILP model to determine the integrity for different graph families, where we discovered interesting patterns and structures. These findings allowed us to establish boundaries or determine exact values of the integrity for those graph families. The graph families analyzed include Glued Paths, generalized Theta graphs, (Double) Cone graphs, Fan graph, Lollipop graphs, generalized Barbell graphs, (Dutch) Windmill graphs, Paley graphs, and Kneser graphs.

During our research on integrity, intriguing open problems emerged, particularly in the context of Paley graphs (see Definition 5.22) and Kneser graphs (see Definition 5.29). Within these families, we uncovered compelling patterns and structures, which led us to formulate two conjectures. These conjectures provide exciting challenges and promising avenues for further research in graph theory and vertex integrity.

The first conjecture centers around the integrity for Paley graphs. Figure 6.1 illustrates the integrity values of Paley graphs with q vertices, highlighting a recurring pattern. The exact values can be found in Table A.1.

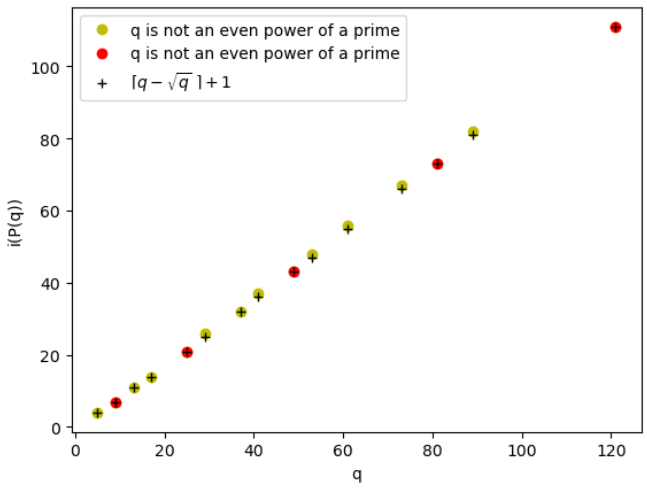


Figure 6.1: Integrity of Paley graphs with q vertices. This plot showcases the integrity values of Paley graphs with q vertices. The x-axis represents the values of q, while the y-axis displays the corresponding integrity values. The red points on the plot represent cases where q is an even power of a prime (Paley graphs), while the yellow points correspond to other values of q. Additionally, the plot includes a '+' symbol, indicating the value of $\lceil q - \sqrt{q} \rceil + 1$.

Conjecture 6.1. *For any Paley graph $P(q)$,*

$$\iota(P(q)) \geq \lceil q - \sqrt{q} \rceil + 1.$$

Combining the results from Conjecture 6.1 and Corollary 5.27, we obtain the following equality.

Theorem 6.2. *If Conjecture 6.1 is true, then for any Paley graph $P(q)$, where q is an even power of a prime,*

$$\iota(P(q)) = q - \sqrt{q} + 1.$$

The second conjecture centers around the integrity for Kneser graphs. Figure 6.2 illustrates the integrity values of Kneser graphs with parameters n and k, highlighting a recurring pattern. The exact values can be found in Table A.2.

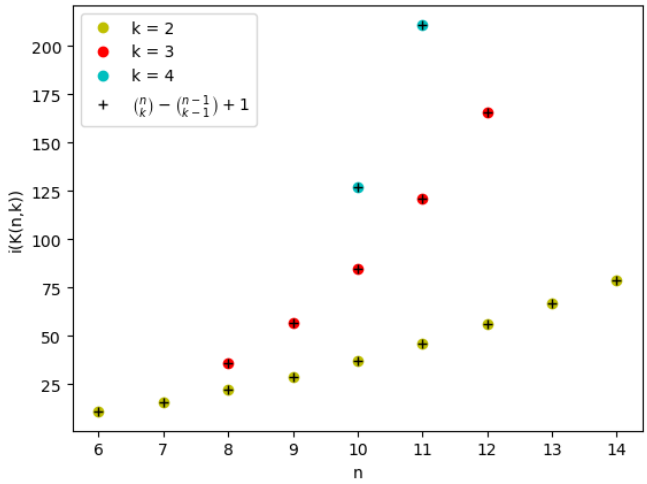


Figure 6.2: Integrity of Kneser graphs with parameters n and k. This plot showcases the integrity values of Kneser graphs with parameters n and k. The x-axis represents the values of n, while the y-axis displays the corresponding integrity values. The colour of the points represents the value of k. Additionally, the plot includes a '+' symbol, indicating the value of $\binom{n}{k} - \binom{n-1}{k-1} + 1$.

Conjecture 6.3. *For any non-empty Kneser graph $K(n,k)$, with $1 < k < \lfloor \frac{n}{2} \rfloor$,*

$$\iota(K(n,k)) = \binom{n}{k} - \binom{n-1}{k-1} + 1.$$

Remark 6.4. *If Conjecture 6.3 is assumed to be true, then the integrity values for all Kneser graphs, except for the Odd graphs, are known.*

Paley and Kneser graphs are well-known for their high symmetry and have been extensively studied. The conjectures formulated in this study carry great significance, as they highlight the observation that certain Paley and Kneser graphs showcase the strictness of the upper bound proposed by Proposition 3.14, which applies to all graphs.

References

[1] M. Aigner, G. M. Ziegler, M. Aigner, and G. M. Ziegler. The chromatic number of Kneser graphs. *Proofs from THE BOOK*, pages 251–255, 2010.

[2] N. Alon, A. Bishnoi, S. Das, and A. Neri. Strong blocking sets and minimal codes from expander graphs. *arXiv preprint arXiv:2305.15297*, 2023.

[3] K. S. Bagga, L. W. Beineke, W. D. Goddard, M. J. Lipman, and R. E. Pippert. A survey of integrity. *Discrete Applied Mathematics*, 37-38:13–28, 1992. ISSN 0166-218X. doi: 10.1016/0166-218X(92)90122-Q. URL https://www.sciencedirect.com/science/article/pii/0166218X9290122Q.

[4] W. Ball and H. Coxeter. *Mathematical Recreations and Essays*. Dover Recreational Math Series. Dover Publications, 1987. ISBN 9780486253572. URL https://books.google.nl/books?id=9lJqNJhYc9oC.

[5] C. Barefoot, R. Entringer, and H. Swart. Integrity of trees and powers of cycles. *Congr. Numer*, 58:103–114, 1987.

[6] C. A. Barefoot, R. Entringer, and H. Swart. Vulnerability in graphs-a comparative survey. *J. Combin. Math. Combin. Comput*, 1(38):13–22, 1987.

[7] N. Biggs, N. L. Biggs, and B. Norman. *Algebraic graph theory*. Number 67. Cambridge university press, 1993.

[8] R. C. Bose. Strongly regular graphs, partial geometries and partially balanced designs. *Pacific Journal of Mathematics*, 13:389–419, 1963.

[9] I. Broere, D. Döman, and J. N. Ridley. The clique numbers and chromatic numbers of certain Paley graphs. *Quaestiones Mathematicae*, 11(1):91–93, 1988.

[10] F. Buckley and F. Harary. On the euclidean dimension of a wheel. *Graphs and Combinatorics*, 4 (1):23–30, 1988.

[11] G. Chartrand. Cut-vertices and bridges. *introductory graph theory*, pages 45–49, 1985.

[12] L. H. Clark, R. C. Entringer, and M. R. Fellows. Computational complexity of integrity. *J. Combin. Math. Combin. Comput*, 2:179–191, 1987.

[13] P. G. Drange, M. Dregi, and P. van't Hof. On the computational complexity of vertex integrity and component order connectivity. *Algorithmica*, 76:1181–1202, 2016.

[14] L. Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, pages 128–140, 1741.

[15] H. Frank and I. Frisch. Analysis and design of survivable networks. *IEEE Transactions on Communication Technology*, 18(5):501–519, 1970. doi: 10.1109/TCOM.1970.1090419.

[16] S. Freitas, D. Yang, S. Kumar, H. Tong, and D. H. Chau. Graph vulnerability and robustness: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 35(6):5915–5934, 2023. doi: 10.1109/TKDE.2022.3163672.

[17] J. A. Gallian. A dynamic survey of graph labeling. *Electronic Journal of combinatorics*, 1 (DynamicSurveys):DS6, 2018.

[18] M. R. Garey. Computers and intractability: A guide to the theory of np-completeness, freeman. *Fundamental*, 1997.

[19] A. Ghosh, S. Boyd, and A. Saberi. Minimizing effective resistance of a graph. *SIAM review*, 50 (1):37–66, 2008.

[20] C. Godsil and G. F. Royle. *Algebraic graph theory*, volume 207. Springer Science & Business Media, 2001.

[21] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023. URL https://www.gurobi.com.

[22] F. Harary. *Graph Theory*. Addison-Wesley Series in Mathematics. Addison-Wesley Longman, Incorporated, 1969. ISBN 9780201027877. URL https://books.google.nl/books?id=0A4rmgEACAAJ.

[23] R. M. Karp. *Reducibility among combinatorial problems*. Springer, 2010.

[24] D. Kratsch, T. Kloks, and H. Muller. Measuring the vulnerability for classes of intersection graphs. *Discrete Applied Mathematics*, 77(3):259–270, 1997.

[25] J. Liu, M. Zhou, S. Wang, and P. Liu. A comparative study of network robustness measures. *Frontiers of Computer Science*, 11:568–584, 2017.

[26] L. Lovász. Kneser's conjecture, chromatic number, and homotopy. *Journal of Combinatorial Theory, Series A*, 25(3):319–324, 1978.

[27] S. Sankaran and N. Chidambaram. On hop domination number of some generalized graph structures. *Ural Mathematical Journal*, 7:121–135, 12 2021. doi: 10.15826/umj.2021.2.009.

[28] G. Sathiamoorthy and T. Janakiraman. Graceful labeling of generalized theta graphs. *National Academy science letters*, 41:121–122, 2018.

[29] S. Skiena. *Implementing discrete mathematics: combinatorics and graph theory with Mathematica*. Addison-Wesley Longman Publishing Co., Inc., 1991.

[30] S. A. University. Königsberg bridges. [Image], n.d. URL https://mathshistory.st-andrews.ac.uk/Extras/Konigsberg/.

[31] G. Van Rossum and F. L. Drake Jr. *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

[32] D. B. West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001.

[33] H. Whitney. Congruent graphs and the connectivity of graphs. *American Journal of Mathematics*, 54:61–79, 1932.

A

Data

q	integrity	$\lvert S\rvert$	$\lceil q - \sqrt{q} \rceil + 1$
5	4	3	4
9	7	6	7
13	11	10	11
17	14	12	14
25	21	20	21
29	26	25	25
37	32	30	32
41	37	36	36
49	43	42	43
53	48	46	47
61	56	54	55
73	67	64	66
81	73	72	73
89	82	80	81
121	111	110	111

Table A.1: Integrity of Paley graphs with q vertices. Comparison of integrity values, the size of the set of removed vertices, and the lower bound based on q. The integrity values were calculated using the ILP model described in Chapter 4.

$n \setminus k$	2	3	4
5	6		
6	11		
7	16	17	
8	22	36	
9	29	57	57
10	37	85	127
11	46	121	211
12	56	166	
13	67		
14	79		

Table A.2: Integrity of Kneser graphs with parameters n and k. Comparison of integrity values for various combinations of n and k. The integrity values were computed using the ILP model described in Chapter 4.

B

Python Code

B.1. General Functions

[ ]:
```
def pline():
    print('=' * 109)

def log(col, value):
    if col == 'red':
        print('\033[91m' + str(value) + '\033[0m')

def header(value):
    pline()
    print()
    log('red', value)
    print()
    pline()
    print()

def block(value):
    print()
    log('red', value)
    print()
```

B.2. Graph Families

[ ]:
```
from sage.all import *

def ThetaGraph(k, l):
    node_s = -1
    node_f = k*(l-2) + 1

    P = graphs.PathGraph(l - 2)

    G = P
    for i in range(k - 1):
        G += P
```

```
    for i in range(k):
        j = i * (l - 2)
        G.add_edge(node_s, j)
        G.add_edge(j + l - 3, node_f)

    return G

def getThetaPos(G, k, l):
    result = dict()

    for v in G:
        pos = []
        if v == -1:
            pos.append((k - 1) / 2)
        elif v == len(G) - 1:
            pos.append((k - 1) / 2)
        else:
            pos.append(floor(v / (l - 2)))

        if v == -1:
            pos.append(-1)
        elif v == len(G) - 1:
            pos.append(l - 2)
        else:
            pos.append(v % (l - 2))

        result[v] = pos

    return result

def getPaleyPos(G, q):
    C = graphs.CycleGraph(q)
    C.relabel(list(G))
    return C.get_pos()

def getKneserPos(G, n):
    C = graphs.CycleGraph(n)
    C.relabel(list(G))
    return C.get_pos()
```

B.3. Graph Properties

[ ]:
```
import gurobipy as gp
from gurobipy import GRB

def m(G, Output = 1):
    G.relabel(range(1, len(G) + 1))

    A = G.adjacency_matrix()
    V = list(set(G))
    C = range(len(V))

    # Create a new model
    m = gp.Model()
```

```
    # Create variables
    M = m.addVar(name="M", vtype=GRB.INTEGER, lb=0, ub=len(V))

    Cv = {}
    for i in range(len(C)):
        c = C[i]
        for v in V:
            Cv[c, v] = m.addVar(name="c_%s,%s"%(c, v), vtype=GRB.BINARY)

    # Set objective function
    m.setObjective(M, GRB.MINIMIZE)

    # Add constraints
    for i in range(len(V)):
        m.addConstr(gp.quicksum([ Cv[C[i], v] for v in V ]) <= M)

        m.addConstr(gp.quicksum([ Cv[c, V[i]] for c in C ]) == 1)

        for c in C:
            for j in range(i + 1, len(V)):
                if A[i][j]:
                    m.addConstr(Cv[c, V[i]] == Cv[c, V[j]])

    m.setParam('OutputFlag', Output)

    # Solve it!
    m.optimize()

    return (int(m.objVal))

def a(G, Output = 1):
    G.relabel(range(1, len(G) + 1))

    A = G.adjacency_matrix()
    V = list(set(G))
    C = range(len(V))

    # Create a new model
    m = gp.Model()

    # Create variables
    Sv = {}
    for i in range(len(C)):
        Sv[V[i]] = m.addVar(name="s_%s"%(V[i]), vtype=GRB.BINARY)

    # Set objective function
    m.setObjective(gp.quicksum([ Sv[v] for v in V ]), GRB.MAXIMIZE)

    # Add constraints
    for i in range(len(V)):
        for j in range(i + 1, len(V)):
            if A[i][j]:
                m.addConstr(Sv[V[i]] + Sv[V[j]] <= 1)
```

```
    m.setParam('OutputFlag', Output)

    # Solve it!
    m.optimize()

    sets = [v for v in V if Sv[v].X == 1]

    return (int(m.objVal), sets)

def z(G, Output = 1):
    G.relabel(range(1, len(G) + 1))

    A = G.adjacency_matrix()
    V = list(set(G))
    C = range(len(V))

    # Create a new model
    m = gp.Model()

    # Create variables
    M = m.addVar(name="M", vtype=GRB.INTEGER, lb=0, ub=len(V))

    Av = {}
    Bv = {}
    for i in range(len(C)):
        Av[V[i]] = m.addVar(name="a_%s"%(V[i]), vtype=GRB.BINARY)
        Bv[V[i]] = m.addVar(name="b_%s"%(V[i]), vtype=GRB.BINARY)

    # Set objective function
    m.setObjective(M, GRB.MAXIMIZE)

    m.addConstr(gp.quicksum([ Av[v] for v in V ]) == M)
    m.addConstr(gp.quicksum([ Bv[v] for v in V ]) == M)

    # Add constraints
    for i in range(len(V)):
        m.addConstr(Av[V[i]] + Bv[V[i]] <= 1)

        for j in range(i + 1, len(V)):
            if A[i][j]:
                m.addConstr(Av[V[i]] + Bv[V[j]] <= 1)
                m.addConstr(Av[V[j]] + Bv[V[i]] <= 1)

    m.setParam('OutputFlag', Output)

    # Solve it!
    m.optimize()

    sets = ([v for v in V if Av[v].X == 1], [v for v in V if Bv[v].X == 1])

    return (int(m.objVal), sets)

def i(G, Output = 1):
    G.relabel(range(1, len(G) + 1))
```

```
    A = G.adjacency_matrix()
    V = list(set(G))
    C = range(len(V))

    # Create a new model
    m = gp.Model()

    # Create variables
    M = m.addVar(name="M", vtype=GRB.CONTINUOUS, lb=0, ub=len(V))

    Cv = {}
    Sv = {}
    for i in range(len(C)):
        Sv[V[i]] = m.addVar(name="s_%s"%(V[i]), vtype=GRB.BINARY)

        c = C[i]
        for v in V:
            Cv[c, v] = m.addVar(name="c_%s,%s"%(c, v), vtype=GRB.BINARY)

    # Set objective function
    m.setObjective(gp.quicksum([ Sv[v] for v in V ]) + M, GRB.MINIMIZE)

    # Add constraints
    for i in range(len(V)):
        m.addConstr(gp.quicksum([ Cv[C[i], v] for v in V ]) <= M)

        m.addConstr(gp.quicksum([ Cv[c, V[i]] for c in C ]) == 1 -
 ↪Sv[V[i]])

#         if i < len(V) - 1:
#             m.addConstr(gp.quicksum([ Cv[C[i], v] for v in V ]) <= gp.
 ↪quicksum([ Cv[C[i + 1], v] for v in V ]))

        for c in C:
            for j in range(i + 1, len(V)):
                if A[i][j]:
                    m.addConstr(Cv[c, V[i]] - Cv[c, V[j]] <= Sv[V[i]] +
 ↪Sv[V[j]])
                    m.addConstr(Cv[c, V[j]] - Cv[c, V[i]] <= Sv[V[i]] +
 ↪Sv[V[j]])

    m.setParam('OutputFlag', Output)

    # Solve it!
    m.optimize()

    sets = [v for v in V if Sv[v].X == 1]

    return (int(m.objVal), sets)
```

B.4. File Management

[ ]:
```
import os

import pandas as pd

cwd = './Files/'

def readFile(file):
    path = os.path.join(cwd, file)
    df = pd.read_excel(path, index_col=0)

    cols = list(df.columns)

    return (df, cols)

def writeFile(file, cols, values):
    data = {i: dict(zip(cols, vals)) for i, vals in enumerate(values)}

    df = pd.DataFrame(data=data).T

    path = os.path.join(cwd, file)
    df.to_excel(path)

    display(df)

def addToFile(file, values, sort = None):
    df, cols = readFile(file)

    data = {i: dict(zip(cols, vals)) for i, vals in enumerate(values)}

    df_n = pd.DataFrame(data=data).T

    df = pd.concat([df, df_n], ignore_index=True)
    if sort:
        df.sort_values(list(sort), inplace=True)

    path = os.path.join(cwd, file)
    df.to_excel(path)

    display(df)
```

B.5. Paley Graphs

[ ]:
```
#
# Modules
#

import os
import random
import time
import math
import numpy as np
```

```
import matplotlib.pyplot as plt
from IPython.display import display
from ast import literal_eval

#
# Module Files
#

sys.path.append('./Modules')

import Functions as f
import GraphFamilies as gf
import GraphProperties as gp
import FileManagement as fm

from importlib import reload

reload(f)
reload(gf)
reload(gp)
reload(fm)
```

[ ]:
```
def checkPaleyGraph(q):
    if q**0.5 == int(q**0.5):
        return 'Quadratic PaleyGraph(q)'
    return
```

[ ]:
```
# i(G)

# For other properties, replace all i's with the for example a or z.

df, cols = fm.readFile('paley-i.xlsx')

reset = False

df_v, cols_v = fm.readFile('paley-values.xlsx')

Lz = []
for i, q in enumerate(df_v[cols_v[0]].head(int(40))):
    if not checkPaleyGraph(q):
        continue

    if not reset and len(df[df[cols[0]] == q]) > 0:
        continue

    G = graphs.PaleyGraph(q)

    f.header('P(' + str(q) + ')')

    i_solve, sets = gp.i(G)

    Li.append((G, q, i_solve, sets))

    f.block('i(G) = ' + str(i_solve))
```

```
#     pos = gf.getPaleyPos(G, q)
#     G.show(vertex_colors={ '#FF0000': list(sets) }, pos=pos)
```

[ ]:
```
file = 'paley-i.xlsx'

cols = ('q', 'i(G)', '|S|', 'S')
sort = ('q')

values = [(q, i_value, len(sets), sets) for (G, q, i_value, sets) in Li]

if reset:
    fm.writeFile(file, cols, values, sort)
else:
    fm.addToFile(file, values, sort)
```

[ ]:
```
df, cols = fm.readFile('paley-i.xlsx')

fig, ax = plt.subplots()

ax.plot(df[cols[0]], df[cols[1]], 'bo', label=cols[1])

plt.xlabel(cols[0])
plt.ylabel(cols[1])

plt.legend()

plt.show()

display(df[cols[0:2]].T)
```