The School of Mathematics

THE UNIVERSITY
of EDINBURGH

Computational Analysis of Quadratic Reformulations Impact on Semi-definite Programming Relaxation: A Study on the Graph Coloring Problem

by

Ahmed A. Al-Ali

Dissertation Presented for the Degree of MSc in Operational Research with Data Science

Sep/2023

Supervised by
prof. Dr. Emre Alper Yildirim

Own Work Declaration

This dissertation is submitted in partial fulfilment of the requirements for the degree of MSc in Operational Research with Data Science. I Declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Ahmed A. Al-Ali

Word Count: 22248 words

Dedication

To my younger self,

This work is a testament to your dreams, your relentless curiosity, and your unwavering resilience. It serves as a tangible proof of the journeys you embarked on, the late nights you spent studying, and the questions you dared to ask.

You might have stumbled and faltered along the way, but you always found the strength to rise and press on. The path was often steep, yet you climbed. The journey was long, yet you persisted. The problems were complex, yet you solved them.Through every challenge and setback, you discovered more about yourself and the world around you. You learned the importance of grit, determination, and passion. You realized the power of knowledge, the joy of discovery, and the satisfaction that comes from hard work.

So, here's to you, for embarking on a journey filled with knowledge and discovery, for embracing challenges with an open heart, and for never letting the flame of curiosity die out. Your dedication and perseverance made this all possible.May you always remember the journey that led to this achievement and may it continue to inspire you to explore, to question, and to innovate.

With pride and affection,
The 22 years old Ahmed

Acknowledgments

I would like to extend my sincerest gratitude to Dr. Emre Alper Yildirim for his invaluable guidance, expertise, and mentorship throughout the course of this dissertation. His unwavering support and insightful critiques have been instrumental in shaping this research. I am particularly thankful for the countless hours he devoted to discussing the nuances of the research topic, providing constructive feedback, and ensuring the highest quality of academic rigor. His encouragement and commitment to excellence have not only enhanced the quality of this work but have also made the journey an enriching educational experience for me. I feel privileged to have had the opportunity to work under his tutelage, and I am better positioned for future scholarly pursuits because of it.

I am grateful for the time and effort Dr. Yildirim invested in me and this research. His contributions have been a cornerstone of this academic endeavor, and for that, I am eternally grateful.

Abstract

The Graph Coloring Problem (GCP) poses significant computational challenges due to its combinatorial nature and NP-hard complexity. This dissertation investigates techniques to enhance solutions for the GCP, specifically through quadratic reformulations and semidefinite relaxation. Mixed-integer programming, linear programming relaxation, and semidefinite programming are implemented and compared on benchmark graph instances. Further, quadratic constraints are incrementally added to strengthen the semidefinite formulations. The study analyzes the impact of these relaxations and reformulations on lower bound tightness and computational demands. Results show semidefinite relaxation with quadratic enhancements can provide tighter bounds but incurs greater computational costs. The enhancements exhibit varied efficacy based on graph structure and density. Statistical analysis confirms their significant yet nuanced effects on optimizing performance metrics. This research contributes novel insights into leveraging quadratic expressions and semidefinite programming to achieve higher-quality solutions for computationally challenging problems like the GCP.

Contents

Chapter 1

Introduction

The chapter begins by highlighting the utility of mathematical optimization methods like Linear Programming (LP), Integer Programming (IP), and Mixed-Integer Programming (MIP) in solving complex problems across various fields. However, it notes that these conventional techniques often struggle with specific types of problems, such as combinatorial issues exemplified by the Graph Coloring Problem (GCP).

The chapter then provides a fundamental understanding of graph theory, explaining that graphs are composed of vertices (or nodes) and edges (or arcs). It elaborates on the objective of GCP, which is to assign unique colors to adjacent vertices or edges in a graph. Originating from Francis Guthrie's Four Color Problem, the GCP aims to color a map's neighboring regions with the least number of unique colors. Two main variants of GCP—vertex coloring and edge coloring—are introduced, both aiming to achieve conflict-free coloration of the entire graph.

The chapter also underscores the notorious computational complexity of GCP, categorizing it as an NP-hard problem. This implies that there is currently no known efficient algorithm to universally solve the GCP. The chapter concludes by outlining the motivations and objectives for exploring GCP and provides an overview of the dissertation's structure.

1.1 Problem Definition

Definition 1.1. The Graph Coloring is a problem on an undirected graph Let $G = (V, E)$ be an undirected simple graph and C be a set of colors. A coloring of G is a function $\omega : V \to C$ such that $\omega(u) = \omega(v)$ for every edge (u, v) of G. Given a coloring ω of G, the class color of $j \in C$, denoted by $\omega^{-1}(j)$, is the set of vertices v colored by j, i.e., such that $\omega(v) = j$. Clearly, each class color is a stable set of G and, therefore, any coloring can be thought of as a partition of vertices into stable sets. The active colors of a coloring f, denoted by A, are those ones assigned to some vertex, i.e.,

$$A = \{j \in C : \omega^{-1}(j) \neq \emptyset\}.$$

The GCP consists of finding a coloring of G with minimum cardinality of A. This minimum is called the chromatic number of G, denoted by $\chi(G)$, and it is known that obtaining this number is NP-hard for general graphs. Having defined the GCP, it is imperative to note that this problem has been the subject of extensive research due to its inherent complexity and practical relevance. The following theorems represent pivotal contributions that address different facets of the problem, including algorithmic approaches, computational tractability, and mathematical formulations.

Theorem 1.1 Branch-and-Cut for GCP Given a graph G = (V, E) and a coloring w, there exists an efficient branch-and-cut algorithm that solves the GCP to optimality (Padberg adn et al. Rinaldi 1991). The algorithm branches on the selection of vertices to color and applies cutting planes based on violated coloring constraints.

Theorem 1.2 Linear Programming Relaxation of GCP The integrality gap of the natural LP relaxation of the GCP is unbounded (Laurent et al. 2001). This suggests that purely linear approaches may struggle to solve the GCP optimally, motivating the use of additional techniques such as branch-and-cut, SDP, and quadratic reformulations.

Theorem 1.3 Semi-definite Programming in GCP: There exists a polynomial-time algorithm that colors the vertices of any n-vertex graph G with no more than $O(n/log(n))$ colors

(Arora et al. 1998). This algorithm works by applying SDP to approximate a solution to the GCP, followed by randomized rounding to obtain an integer solution.

Theorem 1.4 Quadratic Matrix Reformulation for GCP: Given a graph G = (V, E), a quadratic matrix reformulation of the GCP can be defined, whereby the optimal coloring of G can be found as the solution to a matrix optimization problem (Poljak et al. 1995). This provides an alternative, potentially more efficient way to solve the GCP.

Despite it hardness the need of obtaining concrete solutions to numerous applications motivated the development of this dissertation. Our study strives to merge these diverse techniques, thereby offering a more profound understanding of the impact on SDP relaxation and optimal value bounds concerning the GCP. This research aims to enrich the field's current understanding and ultimately propose a more efficient and comprehensive approach for the GCP and potentially other combinatorial optimization problems.

1.2 Motivations

The Graph Coloring Problem (GCP) is a key issue in combinatorial optimization, relevant both theoretically and practically in areas like scheduling and resource allocation. While solving large-scale GCP instances is computationally intense, the role of relaxation techniques in Mixed-Integer Programming (MIP) formulations, such as Linear Programming (LP) and Semi-Definite Programming (SDP), offers a way to simplify these complex problems. However, the effectiveness of these techniques in the context of the GCP is not yet fully explored. Another area of interest is the concept of problem reformulation, such as incorporating quadratic equations, which could potentially improve computational efficiency. Our research aims to explore the impact of these techniques and reformulations on the efficiency and quality of GCP solutions, thereby contributing to a better understanding of combinatorial optimization.

1.3 Research Questions and Objectives

This dissertation aims to contribute to the burgeoning body of literature on the Graph Coloring Problem (GCP) by systematically interrogating specific facets of optimization techniques. Given the inherent complexity and combinatorial nature of GCP, this study has been architectured to answer questions that exist at the confluence of theoretical soundness and empirical utility. The following objectives provide an integrative scaffolding that encapsulates our multidimensional approach to the GCP.

Objective 1: Comparative Efficacy of SDP Relaxations in Lower-Bound Approximation, with seminal works from Karp and Lovász suggesting the potential of SDP relaxations in tackling GCP (Karp and Lovász et al. 1972), one critical question arises: How effective are SDP relaxations in generating tighter lower bounds for GCP when introduced to enhancements by quadratic formulations? The objective aims to fill this empirical gap by conducting a meticulous comparative study.

Objective 2: Computational Footprint of LP and SDP Relaxations, Boyd and Vandenberghe emphasize the computational burden often associated with relaxation techniques (Boyd and Vandenberghe et al. 2004). As such, a compelling question to address is: What are the computational demands of LP and SDP relaxations in the specific context of GCP? This objective seeks to quantify these demands, thereby delineating their practical limitations.

Objective 3: Impact of Quadratic Reformulations on GCP Solutions, Savelsbergh in 1994 hinted at the utility of problem reformulations in optimization. Yet, the nuances of employing quadratic reformulations within GCP contexts are unclear (Savelsbergh et al. 1994). This objective investigates: How do quadratic constraints, when incrementally added, affect the quality of solutions and computational time in solving GCP?

Objective 4: Evaluating the Performance of Relaxation Techniques Versus Optimal Computation Methods, Padberg and Rinaldi's work in 1991 established the efficacy of the Branch-and-Cut algorithm in Mixed-Integer Programming (Padberg and Rinaldi et al. 1991).

The question then arises: How does this technique perform in relation to relaxation methods in the context of GCP, especially concerning achieving optimal bounds and computational efficiency? This objective is designed to provide an analytical comparison.

Collectively, these objectives are orchestrated to provide an expansive understanding of the targeted problem space. They are crafted not just to answer stand-alone questions but to build a robust and complementary analytical framework for future scholarly endeavors in the realm of combinatorial optimization problems such as GCP.

1.4 Outline

The dissertation is structured to provide a thorough examination of the Graph Coloring Problem (GCP) and its intersection with optimization techniques like Mixed-Integer Programming (MIP) and Semi-Definite Programming (SDP). It starts with an "Introduction" that outlines the research motivations, questions, and objectives. This is followed by "Advanced Techniques," which delves into quadratic reformulations, and relaxation techniques. The fundamentals of graph theory and the GCP are covered in "The Graph Coloring Problem" section. "Semi-definite Programming in Graph Theory and Coloring Problem" focuses on the application of SDP in GCP. A "Literature Review" offers an overview of existing research, while "Mathematical Dynamics" and "Computational Dynamics" explore the programming formulations and computational frameworks, respectively. The "Experimentation" section describes the experimental design and results. The paper concludes with an "Analysis and Discussion," summarizing findings and comparing optimization techniques, followed by a "Conclusion and Future Prospects" that suggests areas for future research. A comprehensive "Bibliography" supplements the document.

Chapter 2

Advanced Techniques

This chapter delves into the realm of Advanced Techniques, spotlighting two pivotal approaches: Quadratic Reformulations of Binary Variables and Relaxation Techniques. These methods hold crucial implications for the broader dissertation by addressing the challenge of solving Mixed-Integer Programming (MIP) and Semidefinite Programming (SDP) problems, which are known for their computational complexity and myriad of real-world applications. The challenge with MIP problems is that they are NP-hard (**Theorem 1.1**). Their solution space is non-convex, which means that there's no guarantee to a global minimum, and the number of possible solutions can grow exponentially with the problem size, making them difficult to solve. This is where relaxation techniques come in. A relaxation of an optimization problem is a new problem that is similar to the original in semantics but different in syntax of the formulation but is easier to solve (**Objective 1**). Two common forms of relaxation in integer programming are the linear programming relaxation and the SDP relaxation.

Specifically, the chapter offers an in-depth understanding of how quadratic reformulations can simplify complex non-linear equations into computationally tractable quadratic forms. Furthermore, it elucidates the role of relaxation techniques in solving NP-hard MIP problems by providing simpler, yet meaningful approximations. The techniques discussed not only contribute to solving individual problems but also enhance the current body of knowledge by offering generalizable tools and frameworks. By arming the reader with these advanced techniques, this chapter aims to broaden the scope and efficiency of optimization methods that can be applied to complex scenarios, thus serving as a cornerstone of the dissertation.

2.1 Quadratic Reformulations of Binary Variables

In many real-world problems, the objective functions or constraints involve product terms of binary variables. Quadratic reformulations enable us to simplify these non-linear equations into quadratic forms, thus offering computational advantages. Quadratic reformulations of binary variables are instrumental in optimization, enabling the mathematical modeling of problems in a quadratic format. This approach is particularly valuable for transforming problems with linear constraints or objectives into equivalent forms that provide potentially more accurate representations in terms of optimality and feasibility. The quadratic reformulation of binary variables is not just for simplification; it can also be used to capture interaction terms between variables, making them essential tools for MIP and SDP formulations.

Quadratic Terms

In MIP, it is often desirable to capture interaction terms between binary variables in both objective functions and constraints. A typical example is an interaction term $x_i \times x_j$. Quadratic reformulation allows us to replace this term with a new variable z_{ij} and impose specific constraints on it. As shown in the example below, Facility Location Problem.

$$\min \sum_i c_i \times x_i + \sum_{ij} c_{ij} \times z_{ij}$$

$$\begin{aligned} \text{s.t.} \quad & z_{ij} \leq x_i && (1)\\ & z_{ij} \leq x_j && (2)\\ & z_{ij} \geq x_i + x_j - 1 && (3)\\ & z_{ij} \geq 0 && (4) \end{aligned}$$

In the context of a facility location problem, assume that opening one facility i has a synergistic effect on another facility j. Quadratic reformulation can be used to capture this interaction effect by introducing variables z_{ij} to represent the synergy between facilities i and j. Quadratic reformulations not only serve as computational simplifications but also capture intricate interactions among variables. Through appropriate techniques, they offer effective pathways for solving complex problems in MIP and SDP, thereby widening the scope of problems that can be tackled efficiently.

2.2 Relaxation Techniques

Relaxations are an integral part of solving MIP problems. that is simpler and computationally more tractable. This process of relaxation provides an upper or lower bound on the optimal solution of the original MIP problem (**Objective 2**). One common relaxation technique is the LP relaxation, which involves transforming a MIP problem into a LP problem by relaxing the integrality constraints on the decision variables (Nemhauser and Wolsey et al. 1988). The result is a linear programming problem that can be solved efficiently using polynomial-time algorithms, like the simplex method or interior-point methods (Boyd and Vandenberghe et al. 2004).

$$\begin{aligned} &\min \quad \mathbf{c}^T\mathbf{x} && && \min \quad \mathbf{c}^T\mathbf{x} && \\ &\text{s.t. } \mathbf{Ax} \leq \mathbf{b} && (1) && \text{s.t. } \mathbf{Ax} \leq \mathbf{b} && (1) \\ &\quad \mathbf{x} \in \mathbb{Z}^n, :: \mathbf{x} \geq 0 && (2) && \quad \mathbf{x} \in \mathbb{R}^n, :: \mathbf{x} \geq 0 && (2) \end{aligned}$$

MIP Standard form **LP relaxed**

LP Relaxation, the integer or binary variables in the MIP are allowed to take on any real values within their defined bounds (**Theorem 1.2**). The LP relaxation often provides a useful lower bound on the optimal solution to the original MIP problem. If the solution to the LP relaxation happens to be integral, it is also the optimal solution to the MIP. However, in most cases, the solution to the LP relaxation is fractional.

$$\begin{aligned} \min \quad & \mathbf{c}^T\mathbf{x} \\ \text{s.t. } & \mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{a}^T\mathbf{x} \leq b && (1) \\ & \mathbf{x}_j \in \mathbb{Z}^n, : \text{for all } j \in J && (2) \\ & \mathbf{x} \geq 0 && (3) \end{aligned}$$

MIP Standard Form

$$\begin{aligned} \min \quad & \mathbf{c}^T\mathbf{x} \\ \text{s.t. } & \text{Tr}(\mathbf{Q}\mathbf{X}) + \mathbf{a}^T\mathbf{x} \leq b && (1) \\ & \mathbf{X} \succeq 0 && (2) \end{aligned}$$

SDP Relaxation of MIP

Semi-definite Relaxation, is often applied to problems with non-convex quadratic constraints or objectives (**Theorem 1.3**). These are typically challenging to solve due to the non-convex nature of the problem. Semi-definite relaxations converts these non-convex problems into SDP problems, which are convex and easier to solve. where Q is a non-convex quadratic form. The problem can be relaxed to an SDP problem through the following substitution $X = xx^T$. Here, $Tr(QX)$ is the trace of QX, and $X = xx^T$ is a rank-one constraint. This relaxation has made the problem convex (as SDP problems are convex) and potentially easier to solve, even though the solution may now be only approximately optimal for the original problem. The application of SDR is particularly useful in the field of signal processing and communications, where many optimization problems naturally contain non-convex quadratic forms (Zhang and Luo et al. 2010).

Relaxation techniques play a crucial role in solving MIP problems by providing useful bounds and feasible solutions (Objectives 1 and 2). They are at the heart of algorithms such as branch-and-bound and cutting-plane methods, and are essential tools in the toolbox of operations researchers and optimization professionals alike.

Chapter 3

The Graph Coloring Problem

In this chapter, we lay the groundwork for understanding the intricacies of graph coloring problems (GCP). We have introduced key definitions that are instrumental in comprehending the subject matter at a fundamental level. The chapter has also shed light on various formulations of the problem, delving into both classical and contemporary perspectives. By dissecting the core components of GCP, we have built a foundational understanding that will be essential for the subsequent discussions in this dissertation.

Chapters 6 and 7 will build upon this foundation by exploring the mathematical and computational dynamics of GCP, respectively. We will employ the principles, definitions, and formulations discussed in this chapter to address more complex issues, analyze algorithms, and investigate real-world applications. In doing so, we aim to offer a comprehensive understanding of graph coloring problems, their computational complexities, and their implications in various fields.

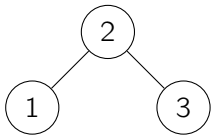
3.1 Graph Theory

Definition 3.1. Directed Graph In graph theory, a graph consists of a set of vertices V connected by a set of edges E. An edge e_{ij} is an ordered pair of vertices (v_i, v_j), where $v_i, v_j \in V$. In the context of directed graphs, e_{ij} is not equal to e_{ji}. Self-loops, represented by e_{ii}, are not included in the graph, meaning that $e_{ii} \notin E$ for all i.

A graph G is defined by its vertex set $V(G)$ and its edge set $E(G)$. An edge in the graph is essentially a connection between two specific vertices, represented as v_i and v_j, where $v_i, v_j \in V(G)$. If a graph H has its vertex set $V(H)$ and edge set $E(H)$ such that $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$, then H is considered a subgraph of G. If H is an induced subgraph, it contains only those edges that connect vertices in $V(H)$.

3.2 Graph Representations

In Graph theory, the effective representation of graphs is crucial for optimizing graph algorithms. There are two prevalent methods for this: the adjacency matrix and the adjacency list. Each has distinct time and space complexities, and the choice between them often depends on the specific nature of the graph problem at hand.



(a) Graph

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

(b) Adjacency Matrix

Vertex	Adjacency List
1	[2, 3]
2	[1, 3]
3	[1, 2]

(c) Adjacency Lists

Figure 3.1: 3 Node Graph and its Representations

Based on **Definition 3.1**, it follows that an adjacency matrix is a binary $n \times n$ matrix, with n being the number of vertices (see Figure 3.1). The cell at the i^{th} row and j^{th} column holds the value 1 if an edge exists between v_i and v_j, and 0 otherwise. For an undirected graphs, the matrix is symmetric. The time and space complexity for constructing an adjacency matrix are $O(n^2)$. This representation allows for $O(1)$ edge existence queries but is less

space-efficient for sparse graphs. On the other hand, an adjacency list is more compact, especially for sparse graphs. For each vertex v_i, a list L_i holds its adjacent vertices. The time complexity to build an adjacency list is $O(m)$, where m is the number of edges. The space complexity is $O(n+m)$. While more space-efficient, adjacency lists require $O(n)$ time for edge existence queries. For algorithms that involve dynamic programming or machine learning, the adjacency matrix's constant-time edge lookup can be beneficial. In contrast, for optimization frameworks, adjacency lists are generally more efficient.

3.3 Graph Coloring and Chromatic Number

A key problem in graph theory is Graph Coloring, in reference to *(Definition 1.1.)*, it is the process of assigning colours to the vertices of an undirected graph, such that no two adjacent vertices have the same colour. The objective is to utilise the smallest number of colours possible. The chromatic number, represented as $\chi(G)$, is the smallest number of colours needed to colour the graph G validly. Several examples help illustrate this concept. For instance, in a complete graph with n vertices, $\chi(Kn) = n$. In a bipartite graph, the maximum chromatic number is 2, and for a planar graph, the maximum is 4. This can be expressed as $\chi(\text{bipartite graph}) \leq 2$ and $\chi(\text{planar graph}) \leq 4$. The complexity associated with determining an approximate $\chi(G)$ within a factor of $n^{\frac{1}{7}-\epsilon}$ is known to be NP-hard *(Theorem 1.1)*. The best-known approximation ratio (Garey and Johnson et al. 1979) is $\frac{n}{(\log\log n)^2(\log n)^3}$.

3.4 Graph Coloring Problem Formulations

GCP can be approached using various optimization techniques, each with its own advantages and drawbacks. A standard MIP formulation uses binary decision variables to represent potential colors for each vertex (Mehrotra and Trick et al. 1996) it aims to minimize the number of colors while ensuring that each vertex is assigned exactly one color and adjacent vertices do not share the same color, it serves as a foundation for other more advanced formulations, including Semi-Definite Programming SDO and Quadratic formulations, thus providing a comprehensive way to analyze the GCP (Karger and Kochenberger et al. 2004).

Mixed-Integer Programming Formulation

The MIP formulation, initially proposed by Mehrotra and Trick, serves as a foundational approach for solving the GCP. It employs binary decision variables to assign colors to vertices while minimizing the overall number of colors used (Mehrotra and Trick et al. 1996).

$$\begin{aligned}
\min \quad & \sum_{i=1}^{k} y_i \\
\text{s.t.} \quad & \sum_{i=1}^{k} x_{vi} = 1 \quad \forall v \in V && (1) \\
& x_{ui} + x_{vi} \leq y_i \quad \forall (u,v) \in E, \forall i && (2) \\
& x_{vi} \in \{0,1\} \quad \forall v \in V, \forall i && (3) \\
& y_i \in \{0,1\} \quad \forall i && (4)
\end{aligned}$$

Semi-Definite Programming Formulation

The SDP formulation, introduced by Karger, Motwani, and Sudan in 1998, provides an alternative that may be computationally more efficient for some graph structures. Unlike MIP, SDP allows for a relaxation of the constraints, potentially yielding better approximations in some instances (Karger et al. 1998). This approach transforms the non convexity introduced by quadratic constraints to convex allowing for more intricate relationships within the graph structure that are also not easily captured by linear constraints (Belotti et al. 2009).

$$\begin{aligned}
\min \quad & \text{Tr}(X) \\
\text{s.t.} \quad & X \succeq 0 && (1) \\
& \text{Tr}(X_{vv}) = 1 \quad \forall v \in V && (2) \\
& \text{Tr}(X_{uv}) = -\frac{1}{k-1} \quad \forall (u,v) \in E && (3)
\end{aligned}$$

The SDP relaxation approach offers several advantages that motivate its application to the GCP. By relaxing the integer constraints, SDP converts the non-convex coloring problem into a convex optimization, enabling efficient solution methods to find strong lower bounds (Vandenberghe and Boyd et al. 1996). The SDP relaxation provides high-quality bounds on the chromatic number, being exact for certain graphs (Grötschel et al. 1988). The semidefinite solutions can also be rounded into good feasible colorings (Goemans et al. 1995). Extra side constraints can flexibly be incorporated into the SDP formulation (Laurent et al.

2003). Importantly, the trace operation conveniently represents each coloring constraint as a sparse matrix product that can be included or excluded as needed. This linearizes the quadratic constraints into a computationally tractable form, leveraging sparse matrix techniques which captures the essence of graph coloring through its positive semidefinite matrix variables. So, the modeling versatility, computational tractability, quality bounds, and geometric interpretability of SDP make it a compelling technique for tackling large GCP's. Each of these formulations contributes uniquely to our understanding of the GCP. MIP offers a basic yet robust framework, SDP provides a more tractable alternative with relaxed constraints, and quadratic reformulations allows for a richer set of feasible solutions through quadratic terms. These insights lay the groundwork for the development of our methodological framework.

Chapter 4

Literature Review

Graph theory, a foundational pillar in the realm of combinatorial optimization, has a myriad of applications that span from operational research to computer science, and from logistics to social network analysis. One of the most intriguing yet challenging problems in graph theory is the Graph Coloring Problem (GCP). This chapter aims to explore the rich tapestry of literature surrounding both graph theory and GCP, systematically reviewing various methodologies, problem formulations, and relaxations that have evolved over the years.

The chapter starts by exploring the origins of graph theory and the subsequent formalization of the Graph Coloring Problem (GCP), emphasizing key milestones like chromatic number and map coloring. It then moves on to review various methods developed to solve GCP, offering insights into their scope and limitations. The second section of the chapter is dedicated to relaxation techniques used in GCP, specifically Linear Programming and Semi-definite Programming. These methods are critically evaluated and compared to identify their respective merits and drawbacks. The chapter concludes by pointing out existing gaps in the current research on relaxation techniques for GCP, suggesting avenues for future studies.

By the end of this chapter, the reader will have a comprehensive understanding of the Graph Coloring Problem as discussed in the literature, encompassing its historical context, various methodologies, and the state-of-the-art in relaxation techniques. This foundation will set the stage for the Dynamics of the GCP both Mathematically and Coputationally, and Experimentation detailed in subsequent chapters.

4.1 Graph Theory and Graph Coloring Problem in Literature

4.1.1 Historical Overview of Graph Theory

Graph theory, a fundamental branch of mathematics, deals with the study of graphs and their properties. A graph is a mathematical representation of a set of objects, where some pairs of objects are connected by links, called edges. The origins of graph theory can be traced back to the 18th century, with foundational contributions from several prominent mathematicians. The seminal work is often considered the starting point of graph theory, In which Euler tackled the famous "Seven Bridges of Königsberg" problem, which involved finding a walk through the city of Königsberg that would cross each of its seven bridges exactly once and return to the starting point. His approach from *Solutio problematis ad geometriam situs pertinentis* to solve this problem introduced the concept of a graph and laid the groundwork for graph theory (Leonhard Euler et al. 1736). He represented the landmasses and bridges as vertices and edges, respectively, and showed that such a walk was not possible due to the existence of vertices with an odd number of adjacent bridges.

The rise of graph theory gained momentum throughout the 19th century, with various mathematicians contributing to its development. Augustin-Louis Cauchy made significant strides in the theory of polyhedra, which can be represented as planar graphs. Introduced by Sir William Rowan Hamilton, the concept of Hamiltonian cycles, a path that visits every vertex in a graph exactly once and returns to the starting vertex, which became a prominent topic in graph theory in literature proceeding it (Hamilton et al. 1859). The Four Color Theorem, one of the most famous problems in graph theory, captured the attention of mathematicians for over a century. The theorem, which states that any map on a plane can be colored using only four colors in such a way that no two adjacent regions have the same color, was first conjectured by Francis Guthrie in 1852. However, the first published proof of this theorem was provided by Kenneth Appel and Wolfgang Haken in 1976, relying heavily on computer-assisted methods (Appel et al. 1977).

In the 20th century, the field of graph theory saw significant advancements and applications in various disciplines. Notably, the concept of graph isomorphism, introduced by Hassler

Whitney in the 1930s, deals with determining whether two graphs are structurally identical. This concept has found applications in chemistry, computer science, and cryptography, among others (Whitney et al. 1932) this formed a fundamental base towards more recent literature in Machine Learning, Cryptography and Quantum Optimization. The study of trees and connectivity in graphs emerged as a central topic by the pioneering work of Arthur Cayley and Otakar Borůvka which laid the foundation for understanding tree structures and network connectivity. The concepts of minimum spanning trees and shortest paths, essential in network optimization and transportation planning, were also developed during this time (Cayley et al. 1889).

The introduction of matrix representation of graphs (Kőnig et al. 1936) particularly the adjacency matrix and incidence matrix, facilitated the analysis of graph properties using linear algebraic methods. Hungarian mathematician Dénes Kőnig made significant contributions in this area during the mid-20th century, in which formed the basis of Semi-definite relaxations for MIP in recent developments. During the latter half of the 20th century, graph theory found numerous applications in computer science, operations research, and other fields. The study of directed graphs and their applications in modeling networks, such as computer networks and social networks, gained prominence. The theory of planar graphs and graph embeddings became relevant in circuit design and graph visualization (Harary et al. 1969). The growth of the internet and the emergence of social media platforms in the late 20th and early 21st centuries brought renewed interest in graph theory. Researchers began studying large-scale graphs, such as web graphs and social networks, leading to the development of algorithms for efficient graph processing and analysis (Barabási et al. 1999).

4.1.2 Evolution of Graph Coloring

The GCP has a rich history that spans several centuries, and it has evolved into a prominent combinatorial optimization challenge with diverse real-world applications. From its humble beginnings with the Seven Bridges problem to its modern-day complexities, the GCP has intrigued mathematicians, computer scientists, and researchers alike. This subsection explores the historical development of the GCP, its various formulations, and its relevance in solving

real-world scenarios.

Chromatic Number and Map Coloring

One of the most captivating problems within the problem of graph coloring is the associated concept of the chromatic number. The chromatic number of a graph, denoted as $\chi(G)$, is the minimum number of colors needed to color the vertices of a graph such that no two adjacent vertices share the same color (**Definition 3.1**). The concept of the chromatic number can be traced back to the Four Color Theorem. This theorem states that any map in a plane can be colored using just four colors in such a way that no two adjacent regions share the same color. It was first conjectured by Francis Guthrie in 1852, and its proof has been one of the most significant milestones in the history of graph theory (Guthrie et al. 1852). Despite its simple formulation, the Four Color Theorem has a rich history filled with numerous failed proofs and counterexamples, illustrating the depth and complexity of graph coloring.

The first published proof of the Four Color Theorem, which heavily relied on computer-assisted methods, was not given until 1976 by Kenneth Appel and Wolfgang Haken (Appelwt al. 1977). However, their proof was highly controversial because it was not feasible for human verification due to the massive computational work involved. Later, Robertson, Seymour, Sanders, and Thomas simplified Appel and Haken's proof using the concept of "discharging," which was originally introduced by Heesch in the 1960s. The revised proof, while still relying on computational verification, significantly reduced the complexity and was more widely accepted by the mathematical community (Robertson et al. 1997). The problem of graph coloring and the chromatic number extends beyond planar graphs and into the realm of graph theory at large. It has many applications, from scheduling and timetabling problems to frequency assignment in cellular networks. However, finding the chromatic number of a graph is an NP-hard problem (**Theorem 1.1**), highlighting the computational complexity inherent in graph coloring (Garey et al. 1979). The chromatic number is closely related to several other parameters in graph theory, such as the clique number and the chromatic index. For instance, the clique number of a graph, denoted by $\omega(G)$, is the maximum number of vertices of a complete subgraph. It is clear that $\omega(G)\chi(G)$ since each vertex in a clique must be colored differently.

In recent years, the advent of sophisticated computational tools and heuristics has led to numerous advancements in graph coloring. Techniques like the greedy algorithm, backtracking, and simulated annealing have been used to find approximate solutions to the GCP. The development of integer programming formulations and the use of metaheuristics for finding optimal colorings have also been significant in recent literature (Malaguti et al. 2010). As we move further into the 21st century, the field of graph coloring continues to flourish. Despite the significant progress made over the years, many questions related to graph coloring and the chromatic number remain unsolved, such as the Hadwiger conjecture and the total coloring conjecture. These challenges, coupled with the rich application potential of graph coloring, promise an exciting future for this area of study.

Formalization of Graph Coloring

The GCP involves the formal task of assigning a color to each vertex of an undirected graph such that no two adjacent vertices share the same color (See **Definition 3.1**). Ramsey, in his 1930 work, formalized the problem of complete coloring, which led to the development of Ramsey Theory (Ramsey, 1930). He proposed that for a given integer c, there exists a number, $R(c)$, such that in any complete graph of $R(c)$ vertices, it either contains a complete subgraph of c vertices all of the same color, or an independent set of c vertices. Welsh and Powell proposed the 'Greedy Coloring Algorithm', which is a systematic approach to color a graph. Starting from a vertex, it assigns the smallest legal color to it and then proceeds to the next uncolored vertex, repeating the process until all vertices are colored. Although this algorithm doesn't always find the optimal coloring, it provides a simple heuristic that has been utilized in many coloring algorithms (Welsh et al. 1967).

The formalization of graph coloring saw a significant development with Lovász's work on the perfect graph theorem (Lovász et al. 1972). He demonstrated that for every induced subgraph G', the chromatic number $\chi(G')$ equals the maximum clique size $\chi(G')$, expressed as $\chi(G') = \omega(G')$. This brought about an intersection of combinatorial graph theory with linear programming, significantly expanding the scope of graph coloring. In 1976, Appel and Haken resolved the Four Color Theorem, one of the most iconic problems in graph coloring.

They proved that every planar graph's chromatic number is at most 4, stated mathematically as $\chi(G) \leq 4$ for all planar G (Appel et al. 1976). This computer-assisted proof pushed the boundaries of computational approaches in mathematics. The field of graph coloring saw further formalizations with the advent of SDP in the 1990s. Goemans and Williamson, introduced SDP relaxations as a tool for combinatorial optimization problems including graph coloring. This innovative approach allowed for an approximation of the chromatic number in polynomial time, which was a significant improvement over previous methods (Goemans et al. 1995).

4.1.3 Exploring methods and innovations in Graph Coloring

Graph coloring problems have been a central topic in combinatorial optimization, leading to the development of various methods to tackle the inherent complexities associated with them. These methods traditionally fall into two primary categories: heuristic algorithms, known for their computational efficiency, and exact algorithms, which provide guarantees of optimality for small instances. In recent years, advances in computing and machine learning have stimulated the development of more sophisticated techniques, integrating the strengths of traditional approaches with novel technologies and strategies. Heuristic Algorithms focus on balancing efficiency and solution quality while exact algorithms for small instances guarantees optimality.

The foundation for current graph coloring methodologies lies in the traditional approaches developed between 1970 and 1999 which was marked by the development of several key algorithms and heuristics for solving the GCP. These solutions predominantly relied on deterministic methods, probabilistic techniques, and heuristic strategies. The Greedy algorithm, introduced by Welsh and Powell, was among the earliest simplistic solutions. This algorithm assigns to each vertex the lowest possible color not yet assigned to its neighboring vertices. The coloring outcome, however, greatly relies on the order in which vertices are visited, and the method does not assure the most efficient solution (Welsh et al. 1967). Despite its simplicity and efficiency for small-scale graphs, it often results in suboptimal solutions for larger, more complex graphs. This method served as a foundation for the development of

more sophisticated algorithms. The exhaustive search procedure, backtracking, provided a fundamental approach to the problem (Brown et al. 1970). By exploring all potential color assignments and reverting upon reaching an incompatible configuration, it ensured an optimal solution. However, due to its exponential computational complexity, it proved inefficient for large graphs. Heuristic methods rose to prominence in the late 1970s and 1980s, the DSATUR algorithm rose. DSATUR selects vertices with the highest saturation degree and, in case of a tie, the one with the highest degree (Brelaz et al. 1979). Although not guaranteeing an optimal solution, it improved efficiency over the Greedy algorithm.

In the 1990s, metaheuristic methods, including Genetic Algorithms (GA), were applied to the GCP (Morgenstern et al. 1996). Inspired by evolutionary principles like mutation, crossover, and selection, GA manipulates a population of colorings to seek satisfactory solutions. It demonstrated effectiveness in handling large-scale problems, even though it might not always yield the optimal solution. Simulated Annealing, another significant contribution of this period, used a probabilistic technique inspired by metallurgy (Johnson et al. 1991). It simulates a heating and cooling process to reach an optimal solution, occasionally allowing uphill moves to escape local optima. This technique improved the quality of solutions by exploring more of the search space. The advent of the new millennium marked significant progress in graph coloring techniques, with computational advancements, innovative heuristics, and hybrid methods. Tabu Search became a widely used metaheuristic technique from the early 2000s (Hertz and De Werra et al. 2001). By implementing a short-term memory structure to discourage recently explored solutions, it managed to escape local optima and effectively explored the solution space. Ant Colony Optimization (ACO), inspired by the foraging behavior of ants, was applied to graph coloring by (Costa and Hertz et al. 2007). The algorithm guides a population of artificial ants to construct solutions by communicating through pheromones, ensuring a more global exploration of the solution space. Hybrid methods that combine the advantages of multiple algorithms started to surface in the 2010s. One notable example is the Hybrid Evolutionary Algorithm by (Galinier and Hao et al. 2011). This approach, by integrating local search, a crossover operator, and a diversification procedure, significantly improved both the quality of solutions and computational time for GCPs.

Quantum computing, though still in its early stages, offered a groundbreaking approach to the GCP (Hadfield et al. 2018). The Quantum Approximate Optimization Algorithm (QAOA) leverages quantum superposition and interference principles to seek optimal solutions. As quantum computing technology advances, it has the potential to redefine the landscape of solutions for NP-hard problems, including graph coloring. Another significant development has been the use of Graph Neural Networks (GNNs) for graph coloring. These networks, effective in learning graph representations and properties, predict vertex colors based on the graph structure. This allows for more efficient coloring, improving time efficiency and solution quality.

4.2 Relaxation Techniques in context of Graph Coloring Problem

4.2.1 Linear Programming Relaxation

Linear programming relaxation has proven to be an effective method in providing tight lower bounds for this challenging optimization problem. We explore the foundations and principles of linear programming relaxation and its relevance in the context of graph coloring. LP, as it stands, indeed has its roots traced back to World War II, where it was developed to optimize logistical operations and resource allocation (Dantzig et al. 1963). This strategic tool, soon after the war, found its purpose redefined and expanded into various civil and industrial applications. LP Relaxation is an elegant technique stemming from the core principles of LP. In essence, LP relaxation replaces the integer constraints with continuous ones, thereby "relaxing" the requirement for integer solutions. This modification, while seemingly minor, is pivotal as it transforms a potentially intractable problem into one solvable in polynomial time, providing a window into feasible solutions or tight lower bounds (Wolsey et al. 1998). The mathematical rendition of the GCP, at its core, seeks the fewest colors to assign to a graph's vertices such that adjacent vertices aren't colored the same (**Definition 3.1**). The prowess of LP relaxation, especially in graph coloring, has been demonstrated by numerous scholars. A pivotal paper by (Lovász et al. 1979) delved into this relationship, showcasing how LP relaxation could guide heuristic techniques to find near-optimal colorings for various graph instances. However, perfection remains elusive. In certain instances, LP relaxation might yield bounds that are far from tight, which translates to derived solutions being non-optimal

(Nemhauser and Wolsey et al. 1988). This brings to light the importance of understanding the nuances and intricacies of the problem at hand. Broadly, LP relaxation's influence in graph coloring stretches across numerous domains: task scheduling, where tasks (vertices) are slotted into non-overlapping time frames (colors) (Pinedo et al. 2012), or frequency allocation in telecommunications, ensuring signal channels (colors) are designated without crosstalk or interference (Hale et al. 1980).

A dive into the core mechanics of LP relaxation reveals a fascinating interplay of algebra and geometry. Essentially, an Integer Linear Programming problem is transformed into its LP counterpart by omitting the integer constraints. This creates a polyhedral structure, where the vertices often represent feasible integer solutions. This structure is key to understanding how LP relaxation helps in identifying promising regions of the solution space (Schrijver et al. 1986). Over time, the applications of LP relaxation have burgeoned, penetrating diverse sectors with environmental conservation becoming a global focus, tools to manage and allocate resources efficiently are indispensable.

On the theoretical spectrum, the application of LP relaxation to GCP was groundbreaking. Chvátal's work was one of the pioneers in this direction. However, subsequent research endeavours, like those by (Karger et al. 1993), delved deeper, investigating the chromatic number of a graph using LP relaxation techniques and offering new insights. The continued adaptation and integration of LP relaxation, fortified by the advancements in computational power and algorithms, are pushing the horizons of what's achievable. It offers a lens to look into intricate problems, bridging the gap between theoretical robustness and practical necessities.

4.2.2 Semi-definite Programming Relaxation

Continuing our exploration of relaxation techniques, we dive into SDP relaxation. This advanced method has garnered significant attention due to its ability to address the complexities of GCPs and offer even tighter lower bounds. We unravel the mathematical underpinnings of SDP relaxation and its application in the domain of graph coloring, witnessing its potential in pushing the boundaries of solution quality.

The roots of SDP relaxation can be traced back to the study of linear matrix inequalities, positioning itself as a natural extension and generalization of linear programming and convex quadratic programming. While several researchers have contributed to its development, one of the pioneers in this field is Alizadeh, whose seminal work established the foundation of SDP's application in combinatorial optimization (Alizadeh et al. 1995). His explorations illuminated the versatility of SDP and provided a solid platform for subsequent investigations into its vast potential. The rise of computational capabilities in the late 20th century was a boon for mathematical programming. As computational techniques became more advanced, the 1990s saw an unprecedented interest in the properties and applications of SDP. One monumental contribution from this era was the introduction of the first polynomial-time algorithms for SDP by Nesterov and Nemirovski (Nesterov and Nemirovski et al. 1994). Their revolutionary algorithms not only streamlined the solution process for SDPs but also laid down a robust framework that would inspire countless subsequent methodologies. The growing fascination with SDP was further enriched by Goemans and Williamson. Their innovative SDP-based algorithm, designed specifically for the GCP, was not only efficient but also offered a remarkable approximation guarantee (Goemans and Williamson et al. 1995). Their research spotlighted the untapped potential of SDP relaxations and reinforced its relevance in combinatorial optimization problems.

SDP relaxation's prowess isn't limited solely to academia or specific mathematical challenges. Its tentacles have spread into real-world applications spanning numerous domains. In control theory, the intrinsic properties of SDP have assisted in designing optimal control systems, ensuring stability and performance. The realm of quantum mechanics has also benefitted from SDP, especially in the optimization of quantum states and processes. Furthermore, in the age of data and AI, machine learning algorithms are leveraging SDP to design and tune models, especially in areas like kernel machines and neural network training. Yet, its significance in graph coloring remains unparalleled. The impressive results yielded by the technique are testament to its capability. A notable mention here is the work of Karger, Motwani, and Sudan, who harnessed the power of SDP relaxation to craft approximation algorithms adept at determining the chromatic number of a graph (Karger et al. 1993). The burgeoning field of massive graph datasets has also seen the impact of SDP, the research by Bose and Joshi

stands out, where they delved into the scalability of SDP for vast graph datasets. Their findings were promising, demonstrating that SDP relaxation could be efficiently scaled to large instances without compromising the integrity and quality of solutions (Bose and Josh et al. 2017). The journey of SDP relaxation, from its historical roots to its modern-day applications, is a testament to its enduring relevance and potency. As computational capacities grow and new challenges emerge, it is poised to remain at the forefront of optimization techniques, continually pushing the boundaries of solution quality.

4.2.3 Comparative Studies of Relaxation Techniques In GCP

As we gather insights into both LP and SDP relaxations, we embark on a comparative analysis to understand their relative strengths and weaknesses. A thorough examination of their performances in the context of GCPs will guide us in comprehending the trade-offs between the two techniques and their implications on solution quality and computational efficiency.

Graph coloring, a classical combinatorial optimization problem, requires coloring the vertices of a graph such that no two adjacent vertices share the same color (**Definition 4.1**). Relaxation techniques have emerged as essential tools to tackle this problem, with LP and SDP relaxations being the most prominent. By delving deep into the comparative dynamics of these techniques, we gain clarity on their relative merits and limitations in the context of graph coloring. LP relaxations work by formulating the GCP into a set of linear inequalities. These inequalities typically represent constraints to ensure that adjacent vertices don't have the same color. As a consequence, LP solutions might not always yield integer solutions directly. However, appropriate rounding techniques or branching strategies can help derive feasible solutions. A groundbreaking study by (Grotschel et al. 1980) employed the ellipsoid method and cutting plane techniques on LP relaxations of various combinatorial problems, including graph coloring. Their findings indicated that while LP relaxations provided good bounds, the derived solutions required careful rounding to ensure feasibility, leading to approximation algorithms. SDP relaxations represent a more recent and advanced approach to the GCP. By considering a broader set of matrix constraints rather than just linear ones, SDP has the potential to provide tighter bounds than LP. This is particularly beneficial when seeking to minimize the number of colors used. Goemans and Williamson's landmark work in

the mid-90s (Goemans and Williamson et al. 1995) introduced a novel SDP-based approximation algorithm for the Max-Cut problem, which has implications for graph coloring. Their method gave rise to tighter bounds than LP-based methods. Following this, Karger, Motwani, and Sudan applied SDP relaxation for the GCP directly and made significant progress in approximation algorithms (Karger et al. 1993).

Comparing he dynamics of both LP and SDP relaxation methods both have their unique set of advantages. LP relaxations tend to be simpler and can be solved using a plethora of well-established algorithms. Additionally, LP's extensive history in operations research means that a lot of tools and methodologies are available to researchers and practitioners. SDP relaxations, on the other hand, can often produce tighter bounds due to their ability to work in the richer space of positive semidefinite matrices. This capability often translates to better approximation algorithms, especially in challenging graph instances. However, SDP also has its limitations. SDP problems generally demand more computational resources than LP, making them potentially unsuitable for very large graphs or real-time applications. A comprehensive study by Laurent (Laurent et al. 2001) compared the Sherali-Adams, Lovász-Schrijver, and Lasserre relaxations (all based on SDP) for 0-1 programming problems, which encompass the GCP. The results indicated that while SDP-based methods provided tighter bounds and better approximation guarantees, they were computationally more intensive than their LP counterparts.

In the realm of graph coloring, choosing between LP and SDP relaxations boils down to the specific requirements of the task at hand. If the primary goal is to get a reasonably good solution quickly, LP might be the preferred choice. However, if the objective leans towards achieving the best possible solution, especially for dense graphs or those with intricate structures, SDP could be more appropriate despite its higher computational demands, while both LP and SDP relaxations offer valuable tools in the arsenal against the GCP, they cater to different needs and scenarios. A balanced perspective, considering both solution quality and computational efficiency, is pivotal when choosing the most suitable approach for a given application.

The continual evolution of LP and SDP relaxation techniques has drawn significant research attention in the last two decades. Here, we highlight a few pivotal contributions that have shaped the contemporary understanding of these methods applied to the GCP. Enhancements in LP Techniques, (Johnson et al. 2010) mplemented advanced cutting-plane methods in LP relaxations for graph coloring, showcasing the potential of enhancing traditional LP techniques. Their experiments revealed that introducing specific classes of cuts could significantly improve the relaxation bounds, especially for sparse graphs. SDP's Robustness in Irregular Graphs in a comprehensive study by (Fernandez and Rajan et al. 2016), SDP relaxations were proven to be particularly robust when dealing with irregular graphs and those with high clique numbers. Their results accentuated SDP's power in cases where LP faced challenges, demonstrating the value of having both tools in the optimization toolbox. Hybrid Approaches The fusion of LP and SDP has also been a topic of intrigue. A collaboration between (Sato and Nakata et al. 2018) brought forward a hybrid technique that combined the computational efficiency of LP with the tighter bounds of SDP. This methodology not only quickened the solution process but also maintained high solution quality, presenting a promising avenue for large-scale problems. Scaling to Larger Graphs, Modern challenges often involve massive graphs with millions of nodes. Acknowledging this, (Zhou and Burer, 2020) presented a scaled SDP relaxation algorithm that leveraged the structure of the GCP. Their results were revolutionary, proving that SDP relaxation could be practical even for enormous graph instances. In sum, the advances in both LP and SDP relaxations have progressively broadened the horizons of combinatorial optimization. As researchers delve deeper into nuanced problems, the knowledge from these studies serves as a beacon, guiding toward improved methodologies and innovative solutions for the GCP.

Chapter 5

Mathematical Dynamics of the Graph Coloring Problem

In this chapter, we present three framework modeling for the optimal coloring of the *Graph Coloring Problem* (GCP). The first evolution of the initial Mixed-Integer Programming (MIP) formulation is characterized by the binary nature of the model. Such model is essential to accurately model the discrete and combinatorial nature of the GCP, ensuring that each vertex is assigned only one color and adjacent vertices receive distinct colors. Consequently, we explore the application of relaxation techniques to simplify the problem while retaining its core characteristics. The first relaxation method implemented is the Linear Programming (LP) relaxation. Here, the original problem is transformed into an LP problem by relaxing the integer constraints on decision variables. This allows the decision variables to take any non-negative real value, resulting in a problem that is computationally easier to solve. Although the solutions from the LP relaxation do not always yield integer values, the resulting solution provides a lower bound for the original problem. To constrain the computational demands based on the computational cost of the nature of GCP we further introduce quadratic constraints to, bound the feasibility of our solutions. We further advance the study by applying SDP relaxation. Unlike the LP relaxation, the SDP relaxation manipulates the quadratic constraints into linear ones by introducing additional variables and constraints exploiting the properties of semidefinite matrices. Although this relaxation technique demands increased computational resources, it typically provides a tighter lower bound compared to LP relaxation.

5.1 Mixed-Integer Programming Formulation

To find an exact solution of the optimal coloring to the GCP we formulate the GCP as an MIP, Recall the defintion of a graph $G = (V, E)$, where V represents the set of vertices and $u, v \in V$ represent the nodes connected by E an edge, then we have the following.

$$\begin{aligned}
&\min \sum_{i=1}^{n} w_i \\
&\text{s.t.} \sum_{i=1}^{n} x_{vi} = 1 \quad \forall v \in V && (1)\\
&\quad x_{ui} + x_{vi} \leq w_i \quad \forall (u, v) \in E, \forall i \in \{1, ..., n\} && (2)\\
&\quad x_{vi} \in \{0, 1\}^{nxn} \quad \forall v \in V, \forall i \in \{1, ..., n\} && (3)\\
&\quad w_i \in \{0, 1\}^{n} \quad \forall i \in \{1, ..., n\} && (4)
\end{aligned}$$

The decision variables in the formulation of the GCP are defined as follows x_{vi}, this is a binary variable that equals 1 if vertex v is assigned color i, and 0 otherwise. w_i, This is also a binary variable that equals 1 if color i is used in the coloring of the graph, and 0 otherwise. The Unique Color Constraint (1), guarantees that each vertex is assigned only a single color. The Proper Color constraint (2), ensures no two adjacent vertices share the same color. Essentially, if a vertex is assigned color i, then w_i must equal 1, marking the color as "used". Node and Color Binary Enforcement Constraint (3,4), set the Binary Nature of x and w variables, enforcing that the decision variables x and w can only take binary values. Together, x_{vi} and w_i constitute the set of decision variables for the problem. For a graph with n vertices, there are n^2 possible x_{vi} variables and n possible w_i variables, making the problem computationally intensive for large graphs. This combinatorial nature of the GCP is what makes it NP-hard. While this formulation transposes the GCP into a mathematical programming model, it is rather rudimentary. Practical solutions often demand more sophisticated approaches due to escalating problem size and complexity.

5.2 Linear Programming Relaxation and formulation

Recall the defintion of a graph $G = (V, E)$, where V represents the set of vertices and $u, v \in V$ represent the nodes connected by E an edge, then we have the following. LP

relaxation involves transforming an integer programming problem into a linear programming problem. The objective is to simplify the problem by ignoring the integer constraints and allowing the decision variables to take any real values within the prescribed bounds. In context of the original GCP, the variables are binary, i.e., x_{vi} and w_i are in $0, 1$. Which implies we remove the integrality constraints from the formulation x_{vi} and w_i to be any value in the interval [0,1], in which then the LP gives a lower bound on the optimal coloring of the GCP.

$$\begin{aligned}
&\min \sum_{i=1}^{n} w_i \\
&\text{s.t.} \sum_{i=1}^{n} x_{vi} = 1 \quad \forall v \in V && (1) \\
&\quad x_{ui} + x_{vi} \leq w_i \quad \forall (u, v) \in E, \forall i \in \{1, \ldots, n\} && (2) \\
&\quad 0 \leq x_{vi}, w_i \leq 1 \quad \forall v \in V, \forall i \in \{1, \ldots, n\} && (3)
\end{aligned}$$

The key difference here is that we have changed the domain of x_{vi} and w_i from the set $0, 1$ to the interval [0,1]. This makes the problem a linear program, which is easier to solve.

5.3 Quadratic Reformulations

To strengthen the initial formulation and potentially improve computational efficiency, we consider several quadratic reformulations that are additional constraints. These quadratic constraints aim to enforce the proper coloring condition in a more rigorous manner and to maintain the connection between the x and w decision variables.

$$x_{vi} x_{vj} = 0, \ \ \forall v \in V, \forall (i, j) \in \{1, ..., n\}, \forall i \neq j$$

Unique color constraint, each node (vertex) in the graph should be assigned exactly one color. This is fundamental to the GCP.

$$x_{ui} x_{vi} = 0, \ \ \forall (u, v) \in E, \forall i \in \{1, \ldots, n\}$$

Adjacent node color conflict constraint, nodes that share an edge (i.e., they are adjacent to each other) should not have the same color. This is the key restriction in the GCP, as it prevents adjacent nodes from being indistinguishable in terms of color.

$$x_{ui} w_i = x_{ui}, \ \ \forall u \in V, \forall i \in \{1, \ldots, n\}$$

Color usage tracking constraint, we maintain a set of variables to track whether a color is used or not in the graph. If a color is assigned to any node, the corresponding variable should be set to 1; otherwise, it remains 0. This constraint helps to keep track of the number of colors used in the solution, which is crucial for minimizing the total number of colors.

$$x_{ui}^2 - x_{ui} = 0, \ \ \forall u \in V, \forall i \in \{1, \ldots, n\}$$

Binary enforcement constraints for node color variables, these constraints ensure that the node color variables are binary, i.e., they can only take on the values 0 or 1. If the variable is 1, it means that the node is assigned the color; otherwise, it's not. These constraints are used in some formulations of the problem to ensure that the variables maintain their binary nature.

$$w_i^2 - w_i = 0, \ \ \forall i \in \{1, \ldots, n\}$$

Binary enforcement constraints for color usage variables, similar to the node color variables, we also need to ensure that the color usage variables are binary. These constraints help to enforce that the color usage variables can only take on the values of 0 (the color is not used) or 1 (the color is used).

5.4 Semi-definite Programming Formulation and Relaxation

Recall the defintion of a graph $G = (V, E)$, where V represents the set of vertices and $u, v \in V$ represent the nodes connected by E an edge and n represents the number of nodes. SDP involves transforming an integer programming problem into a trace of matrices and vectors by mapping decision variables into auxiliary variables where different rows and columns correspond to a set of different decision variables and their interactions. In which then the SDP gives a tighter lower bound on the optimal coloring of the GCP. Moreover we construct the Semi-definite Matrix that is responsible to ensure the positvity of the optimal coloring, by introducing a matrix $\mathbf{Z}$, constructed by matrix $\mathbf{X}$ the column vectors x and x^T.

Construction of Semi-definite Matrix

$$\mathbf{Z} = \begin{bmatrix} x \\ 1 \end{bmatrix} \begin{bmatrix} x^T & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{X} & x \\ x^T & 1 \end{bmatrix}$$

The matrix $\mathbf{Z} \in R^{(n^2+n+1)x(n^2+n+1)}$, ensures that $\mathbf{Z}$ is symmetric, and by construction, it should be semidefinite. This matrix is block-partitioned, which means that the upper left block is the matrix X which is $n^2 + n \times n^2 + n$, the upper right block is the column vector x, the lower left block is the transpose of x, and the lower right block is just a scalar, 1. The SDP relaxation typically tries to find a positive semidefinite matrix Z that adheres to some constraints. By ensuring that Z is positive semidefinite, we implicitly also constrain X to be positive semidefinite. This structure ensures the properties of the SDP are maintained.

$$\mathbf{X} = xx^T = \begin{bmatrix} x_{11} \\ \vdots \\ x_{vi} \\ w_1 \\ \vdots \\ w_i \end{bmatrix} \begin{bmatrix} x_{11} & \dots & x_{vi} & w_1 & \dots & w_i \end{bmatrix} = \begin{bmatrix} x_{11}^2 & \dots & x_{11}x_{vi} & x_{11}w_1 & \dots & x_{11}w_i \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ x_{11}x_{vi} & \dots & x_{vi}^2 & x_{vi}w_1 & \dots & x_{vi}w_i \\ x_{11}w_1 & \dots & x_{vi}w_1 & w_1^2 & \dots & w_1w_i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{11}w_i & \dots & x_{vi}w_i & w_iw_1 & \dots & w_i^2 \end{bmatrix}$$

So matrix $\mathbf{X}$ is constructed by the products of decision variables and color usage indicators. This construction involves both set of decision variables formulated as column vector x and its transpose. The column vector x and the row vector x^T yields the following matrix $X = xx^T$, which is by definition a symmetric matrix. As a result of $x \in R^{n^2+n}$ and $x^T \in R^{n^2+n}$ then $\mathbf{X} \in R^{(n^2+n)x(n^2+n)}$. Embedding X, x, and x^T into the semi-definite matrix Z allows for the optimization problem to be tackled in a more integrated and holistic manner.

Table 5.1: Mapping of Matrix and Vector into Matrix Z

Element in Z	**Component**	**Index**
$Z_{i,j}$	$X_{i,j}$	$i, j \leq n^2 + n$
Z_{i,n^2+n+1}	x_i	$i \leq n^2 + n$
$Z_{n^2+n+1,j}$	x_j^T	$j \leq n^2 + n$

The core idea of this mapping (See Table 5.1) is to ensure that the components, including the matrix X and vectors x and x^T, are unified in the matrix Z. This integrated representation simplifies the optimization problem by embedding these variables in one compact structure. Mapping Matrix X into Z, each element of matrix X is directly mapped into matrix Z at the corresponding position. Mapping Vector x into Z, each element of vector x is placed into the last column of matrix Z. Mapping Vector x^T into Z, every element of the transpose vector x^T is positioned in the last row of matrix Z.

$$\mathbf{Z} = \begin{bmatrix} \mathbf{X}_{1,1} & \cdots & \mathbf{X}_{1,\mathrm{n}^2+\mathrm{n}} & x_1 \\ \vdots & \ddots & \vdots & \vdots \\ \mathbf{X}_{\mathrm{n}^2+\mathrm{n},1} & \cdots & \mathbf{X}_{\mathrm{n}^2+\mathrm{n},\mathrm{n}^2+\mathrm{n}} & x_{n^2+n,n^2+n} \\ x_1^T & \cdots & x^T_{n^2+n,n^2+n} & 1 \end{bmatrix} = \begin{bmatrix} x_{11}^2 & \cdots & x_{11}x_{vi} & x_{11}w_1 & \cdots & x_{11}w_i & x_{11} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{11}x_{vi} & \cdots & x_{vi}^2 & x_{vi}w_1 & \cdots & x_{vi}w_i & x_{vi} \\ x_{11}w_1 & \cdots & x_{vi}w_1 & w_1^2 & \cdots & w_1w_i & w_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{11}w_i & \cdots & x_{vi}w_i & w_iw_1 & \cdots & w_i^2 & w_i \\ x_{11} & \cdots & x_{vi} & w_1 & \cdots & w_i & 1 \end{bmatrix}$$

Its evident now that all decision variables and their interactions are located in the semi-definite matrix Z where all linear decision variables are distributed along the bottom and right side, we have the interactions of the decision variables distributed among the four-partitioned blocks along the middle. Its also worth noting that parts of the quadratic reformulations that we employed are the interactions. Now, here's a breakdown of how this augmented matrix is related to the GCP, 1) X corresponds to the interactions between different colors and vertices, 2) x corresponds to the individual coloring decisions. 3)The scalar "1" in the matrix is typically a normalization constant. The constraints we described earlier for X can be incorporated into Z. For instance, a constraint on an element of X is also a constraint on an element of Z. Now, when we say Z is positive semidefinite, it means that all its eigenvalues are non-negative. This is a convex constraint and it's the reason why SDP can be solved efficiently using interior-point methods. When we solve the SDP, we're trying to find a matrix Z (and hence X and x) that meets all our constraints and is positive semidefinite. Once we solve for Z, we can extract X and x to obtain information about the GCP.

Defining Semi-Definite Matrix Quadratic Elements as Auxiliary Variables

To further linearise the representation of Z, we introduce a set of auxiliary variables that represent the interactions given below.

Table 5.2: Definition of Auxiliary Variables

Auxiliary Variable	Decision Variable Interaction
X_{ij}^{vv}	$x_{vi}x_{vj}$
X_{ii}^{uv}	$x_{ui}x_{vi}$
X_{ii}^{uu}	$x_{ui}x_{ui}$
Y_{iu}^{i}	$x_{ui}w_i$
W_{ii}	w_iw_i

These auxiliary variables (See Table 5.2) replace the quadratic elements which represent the interactions in the semi-definite matrix Z in which these would be used in the mathematical formulation of the constraints and the quadratic reformulations.

Construction of Constraint in terms of Auxilary Varaibles

We observe that those interaction elements of the matrix are the quadratic formulations we intend to use. This allows us to form the SDP counterparts of the quadratic formulations using the auxiliary variables.

$$x_{vi} * x_{vj} = 0 \Rightarrow X_{ij}^{vv} = 0 \quad \forall v \in V, \forall i \neq j$$

For every vertex v ad for every pair of colors $i \neq j$, if vertex v is assigned both colors i and j simultaneously, the respective position in $\mathbf{X}$ representing the combination must be zero. This ensures that a vertex doesn't have two colors at the same time.

$$x_{ui} * x_{vi} = 0 \Rightarrow X_{ii}^{uv} = 0 \quad \forall (u, v) \in E, \forall i$$

For all edges (u, v) in the graph and for each color i, if both vertices u and v are assigned color i, then the respective position in $\mathbf{X}$ corresponding to this edge-color pairing is zero. This makes certain that two adjacent vertices cannot have the same color.

$$x_{ui} * w_u = x_{ui} \Rightarrow Y^i_{iu} = x_{ui} \quad \forall u \in V, \forall i$$

For each vertex u and for every color i, if vertex u is colored with color i and color i is being used in the graph, the position in $\mathbf{X}$ that captures this vertex-color relationship must equal w_i. This ensures a linkage between vertex coloring and color utilization.

$$x^2_{ui} - x_{ui} = 0 \Rightarrow x_{ui} * x_{ui} = x_{ui} \Rightarrow X^{uu}_{ii} = x_{ui} \quad \forall u \in V, \forall i$$

For every vertex u and each color i, if a vertex u is colored with color i, then the position in $\mathbf{X}$ representing this vertex-color assignment is equal to x_{ui}. This reflects the binary nature and ensures that a vertex is either colored with a specific color or not.

$$w^2_i - w_i = 0 \Rightarrow w_i * w_i = w_i \Rightarrow W_{ii} = w_i \quad \forall u \in V, \forall i$$

For each color i, if color i is being used in the graph then the position of W in $\mathbf{X}$ representing the utilization of the colour is w_iThis captures the binary nature of the color usage variable, meaning a color is either used or not in the graph coloring.

MIP Formulation as Semi-definite Program

We commence with a basic SDP, echoing the original MIP. We start by the SDP mathemtical formulation.

$$\min \sum_{i=1}^{n} w_i$$

$$\text{s.t.} \sum_{i=1}^{n} x_{vi} = 1 \quad \forall v \in V \tag{1}$$

$$x_{ui} + x_{vi} \leq w_i \quad \forall (u, v) \in E, \forall i \in \{1, ..., n\} \tag{2}$$

$$\mathbf{X} = [x_{11}, \ldots, x_{vi}, w_1, \ldots, w_i] \quad \forall v \in V, \forall i \in \{1, ..., n\} : \tag{3}$$

$$\mathbf{X} = xx^T \tag{4}$$

$$\mathbf{Z} = \begin{bmatrix} \mathbf{X} & x^T \\ x & 1 \end{bmatrix} \tag{5}$$

$$\mathbf{Z} \succeq 0 \tag{6}$$

In this mirrored problem of the MIP, we are defining a sparse method to solve the baseline MIP formulation by linearizing the terminology of the quadratic reformulations.

Quadratic Enhancemnt Formulation

$$X_{ij}^{vv} = 0 \quad \forall v \in V, \forall (i,j) \in \{1,\ldots,n\}, i \neq j \qquad (1)$$

$$X_{ii}^{uv} = 0 \quad \forall (u,v) \in E, \forall i \in \{1,\ldots,n\} \qquad (2)$$

$$X_{ii}^{uu} = x_{ui} \quad \forall u \in V, \forall i \in \{1,\ldots,n\} \qquad (3)$$

$$Y_{iu}^{i} = x_{ui} \quad \forall u \in V, \forall i \in \{1,\ldots,n\} \qquad (4)$$

$$W_{ii} = w_i \quad \forall i \in \{1,\ldots,n\} \qquad (5)$$

To fortify our SDP model, we add quadratic constraints that further enhance the relationship between the color assignment and usage. These constraints augment the strength of the original formulation, enforcing the graph's proper coloring condition in a more stringent manner.

Chapter 6

Computational Dynamics of the Graph-Coloring Problem

In this chapter we delve into the computational dynamics of the Graph-Coloring Problem (GCP), examining various frameworks to approach its NP-hard complexity. Leveraging the power of MOSEK Fusion API, this chapter explicates the implementation of Mixed-Integer Programming (MIP), Linear Programming Relaxation (LP), and Semi-definite Programming (SDP) with Quadratic Enhancements. Each section discusses how these formulations are built, providing a deep dive into their computational components and implications. Furthermore, how can we compare the frameworkd using the performance metrics and evaluation criteria defined. The chapter commences with the MIP framework, a powerful tool for finding exact solutions but potentially slow for large-scale graphs. Subsequently, we explore the LP Relaxation framework, a computationally efficient method that provides lower bounds to the exact solutions of the problem at the expense of exactness, the chapter concludes with the introduction of SDP and Quadratic Enhancements, focusing on the nuances of handling matrices and indexing schemes for computationally efficient formulation and implementation. By outlining the computational underpinnings of these frameworks, this chapter provides the reader with both theoretical and practical insights into solving the GCP. Through in-depth discussion, tables, and computational analyses, it lays the foundation for understanding the computational dynamics of graph-coloring formulations, setting the stage for subsequent chapters that delve into specific case studies and empirical validations. The implementation code could be find in (Ahmed et al. 2023).

6.1 Code Architecture and Design Philosophy

The architecture of the optimization process (See Figure 6.1) is designed with a focus on modularity and flexibility. It is divided into logical components including modules for data access, model building, solving, and analysis. This separation simplifies maintenance and promotes reusability. Problem formulations are abstracted into separate callable functions, allowing easy switches between different optimization models like MIP, LP, SDP, and QESDP. The architecture also features generalized model construction using templates parameterized by graph data, minimizing code duplication and streamlining the addition of new variants. Parameters for graph instances, formulation types, and additional constraints are exposed to control experiments and isolate the impact of changes. Helper libraries are employed for tasks like data loading and solution analysis, encapsulating complexity. The workflow is structured into four phases: data input, model building, solving, and results recording. A

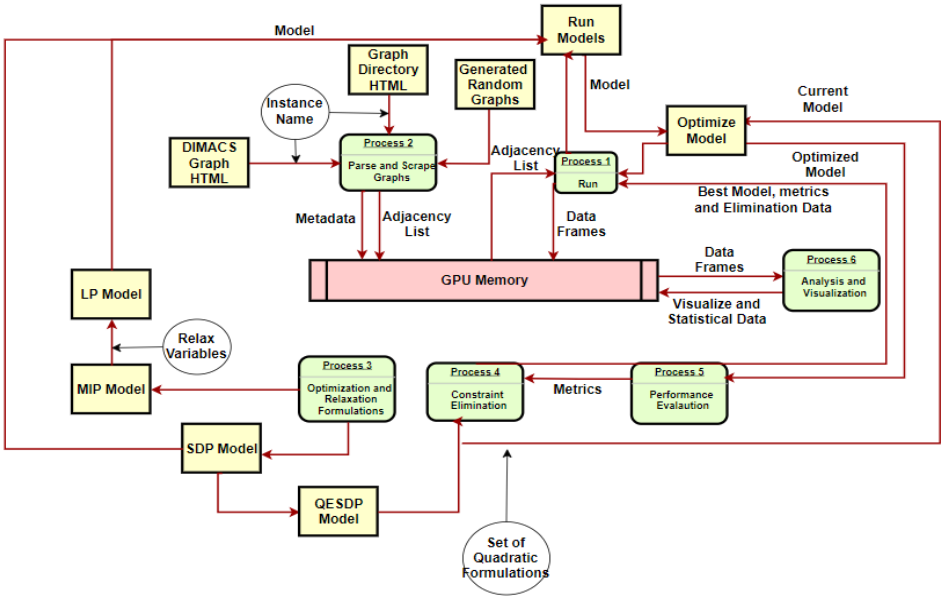


Figure 6.1: Optimization Architecture

The Optimization Architecture (See Figure 6.1) consists of six pivotal processes. The workflow initiates with the 'Running Architecture and Parsing' phase, which fetches adjacency lists and associated metadata from specified URLs. This leads to the 'Integer and Relaxation Formulations' phase that engages multiple mathematical models such as MIP, LP, and SDP,

thereby allowing computational flexibility. The third significant component is the 'Constraint Elimination and Performance Evaluation,' applying an iterative approach to refine and evaluate various constraint combinations. Subsequently, the 'Analysis and Visualization' phase generates data frames and charts for intricate performance scrutiny, corroborated through statistical validation. The architecture's design philosophy accentuates three key elements: re-usability through the use of helper libraries, maintainability and modularity via the separation of concerns and abstractions, and extensibility, enabling the facile incorporation of new modules or constraints. Cumulatively, the architecture and its underlying principles aspire to facilitate swift experimentation and high-performance results in tackling complex graph-centric challenges.

6.2 Implementation Using MOSEK FUSION API

The implementations of the GCP formulations utilize the MOSEK Python API, which provides a high-performance optimization toolkit suitable for large-scale problems. The MOSEK Python API enables creating optimization models by defining variables, objectives, and constraints using Python code. The baseline MIP, LP relaxation, baseline SDP, and the quadratically enhanced SDP formulations were implemented following the methodological framework approach outlined in Chapter . In regards to our architecture (See Figure 6.1) note that not all components of our code utilize MOSEK Fusion API, hence in this section we focus on Integer and Relaxation Formulations "Proccess 3". Utilizing the MOSEK Fusion API, the computational formulation process is designed to closely mimic the mathematical model discussed in Chapter 6. This is achieved through a series of steps, each serving a specific purpose in the overall optimization model. In the optimization process, the Model object acts as the cornerstone, initialized with a command such as *model("model_name")*. Subsequently, decision variables are declared through the method *model.variable("variable_name", shape, domain)*, conforming to the chosen mathematical formulation. To further refine the model, constraints are set using *model.constraint("constraint_name", expression, domain)*. Objectives for the model are similarly established using the *model.objective("optimization_type","expression")*, in accordance with the stipulations of the mathematical formulation.

6.2.1 Mixed-Integer Programming Framework

To find an exact solution for the GCP we employ a computational framework that transforms the formulation into an executable model by using MIP formulation.

Component	Description
Variables	`model.variable("x", [n, n], Domain.binary())` `model.variable("w", n, Domain.binary())`
Objective	`model.objective(ObjectiveSense.Minimize, Expr.sum(w))`
Constraints	`(1) model.constraint(unique_color_v, Expr.sum(x.slice([v,0],[v+1,n])), Domain.equalsTo(1))` `(2) model.constraint(proper_coloring_v_u_i, Expr.sub(Expr.add([x.index(v - 1, i), x.index(u - 1, i)]), w.index(i)), Domain.lessThan(0.0))`

Table 6.1: Computational Framework of MIP Framework

A computational model named `"mip_gcp"` is initialized to house all variables, constraints, and objectives (See Table 6.1). Two types of binary decision variables, $x \in \{0,1\}^{n \times n}$ and $w \in \{0,1\}^n$, are defined using the `model.variable()` method and the `Domain.binary` option. **Unique Color Constraint**, this constraint ensures that each vertex v is assigned exactly one color using `Expr.sum(x.slice([v,0],[v+1,n]))` and `Domain.equalsTo(1)`. **Proper Coloring Constraint**, for each edge (v, u) and color i, a constraint is set up to prevent adjacent vertices from sharing the same color. It uses the expressions `x.index(v - 1, i)` and `x.index(u - 1, i)` to identify the variables for vertices v and u being assigned color i, sums them, and then constrains the sum to be less than 0 using `Domain.lessThan(0.0)`.

Aspect	Computational Implications
Complexity	Solves to optimality but can be slow for large graphs.
Number of Variables	$O(n^2)$
Number of Constraints	$O(2n + nE)$

Table 6.2: Computational Implications of MIP Framework

The computational impact of the MIP formulation (See Table 6.2) outlines the computational impact of the MIP formulation for the Graph Coloring Problem. It indicates that the method can find the exact solution but may be slow for large graphs. The table also quantifies the complexity of the problem, stating that it requires $O(n^2)$ variables and $O(2n + nE)$ constraints. The framework should closely represent the original mathematical MIP formula-

tion. However, this computationally intensive framework reflects the NP-hard nature of the problem, requiring efficient solver strategies for larger graph instances.

6.2.2 Linear Programming Relaxation Framework

To find a lower bound for the exact solution of the GCP, we make use of a computational framework that adapts the problem into an executable model through LP Relaxation.

Component	Description
Variables	`model.variable("x", [n, n], Domain.inRange(0, 1.0))` `model.variable("w", n, Domain.inRange(0, 1.0))`
Objective	`model.objective(ObjectiveSense.Minimize, Expr.sum(w))`
Constraints	`(1) model.constraint(unique_color_i, Expr.sum(x.slice([i,0],[i+1,n])), Domain.equalsTo(1))` `(2) model.constraint(proper_coloring_v_u_i, Expr.sub(Expr.add([x.index(v - 1, i), x.index(u - 1, i)]), w.index(i)), Domain.lessThan(0.0))`

Table 6.3: Computational Framework of LP Relaxation

Similar to the MIP framework, a Model object named `"lp_gcp"` is created. Variables x and w are now allowed to be in the continuous range from 0 to 1, instead of being binary. This relaxation allows the solver to find a bound for the optimal solution much faster than solving the MIP, at the cost of not guaranteeing an exact solution. The constraints remain the same in terms of syntax but the semantics in terms of value distribution changes to be continuous instead of integer ensuring unique coloring for each vertex and proper coloring across edges.

Aspect	Computational Implications
Complexity	Finds a bound for the optimal solution, generally faster than MIP.
Number of Variables	$O(n^2)$
Number of Constraints	$O(2n + nE)$

Table 6.4: Computational Implications of LP Relaxation Framework

The computational characteristics of the LP relaxation (See Table 6.4) is computationally more efficient than its MIP counterpart but offers only a bound for the optimal solution. As before, the complexity of the problem remains the same, requiring $O(n^2)$ variables and $O(2n + nE)$ constraints. The framework provides a faster, although not necessarily exact, solution. This method is especially useful for large graph instances where finding the exact solution is computationally prohibitive.

6.2.3 Semi-defnite Programming Relaxation Framework

To find a more accurate lower bound a computational framework is used that employs SDP (Semidefinite Programming) Relaxation and Quadratic formulations. Unlike other optimization frameworks, SDP uses matrices to represent decision variables. These matrices are then mapped into a single semidefinite matrix through auxiliary variables. This conversion is computationally challenging. In Python's MOSEK Fusion API, all decision variables must be manually defined within one semidefinite matrix. One efficient method for this is indexing, which serves as a bridge between the mathematical problem and its computational representation. This technique allows for the translation of the GCP into sparsable matrices and vectors, the identification of unique positions for each variable, and the consistent formulation of constraints.

$$\mathbf{Z} = \begin{bmatrix} x_{11}^2 & \dots & x_{11}x_{vi} & x_{11}w_1 & \dots & x_{11}w_i & x_{11} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{11}x_{vi} & \dots & x_{vi}^2 & x_{vi}w_1 & \dots & x_{vi}w_i & x_{vi} \\ x_{11}w_1 & \dots & x_{vi}w_1 & w_1^2 & \dots & w_1w_i & w_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{11}w_i & \dots & x_{vi}w_i & w_iw_1 & \dots & w_i^2 & w_i \\ x_{11} & \dots & x_{vi} & w_1 & \dots & w_i & 1 \end{bmatrix}$$

Indexing of Decision Variables in Semi-definite Matrix

One of the crucial steps is proper indexing, which ensures swift and error-free retrieval of variables, facilitates intuitive constraint definition, and provides clarity in the representation. Given the semi-definite matrix **Z** from Chapter 6 Section 4.1, to initiate the indexing schema, it is essential to establish a common notation that represents the position of each element.

- **For individual terms**, the x_{vi} or x_{ui} decision variables are indexed as $Z[n^2 + n + 1, n(voru - 1) + i]$, while the w_i variables follow the schema $Z[n^2 + n + 1, n^2 + i]$.
- **For squared terms**, the diagonal of the matrix contains squared terms of the variables x_{vi} and w_i. For x_{vi} squared variables, their indexing follows $Z[n(v-1)+i, n(v-1)+i]$. For the w squared variables, it's $Z[n^2 + i, n^2 + i]$.

- **For cross product terms between different x variables**, they are indexed as $Z[n(v-1)+i, n(v-1)+j]$ for $i \neq j$. For cross product terms involving x_{vi} and w_i, their indexing is $Z[n^2+i, n(v-1)+i]$.

- **For the scalar value 1**, this constant term is always indexed as $Z[n^2+n+1, n^2+n+1]$.

If we want to formulate a constraint that involves the product of w_i and x_{vi}, instead of searching the entire matrix, we can directly use its index: $Z[n^2+i, n(v-1)+i]$. This systematic approach not only reduces the potential for errors but also significantly enhances the efficiency of the constraint formulation process. The main benefit of this indexing schema is its structured and intuitive design.

Table 6.5: Comparison of MIP and SDP Model

Original Formulation	**SDP Counterpart**
$\sum_{i=1}^{n} w_i$	$\sum_{i=1}^{n} Z[n^2+n+1, n^2+i]$
$\sum_{i=1}^{n} x_{vi} = 1$	$\sum_{i=1}^{n} Z[n^2+n+1, n(v-1)+i] = Z[n^2+n+1, n^2+n+1]$
$x_{ui} + x_{vi} \leq w_i$	$Z[n(u-1)+i, n(v-1)+i] \leq Z[n^2+n+1, n^2+i]$
$x_{vi} \in \{0,1\}^{nxn}$	$Z[n^2+n+1, n(v-1)+i] = Z[n(v-1)+i, n(v-1)+i]$
$w_i \in \{0,1\}^{n}$	$Z[n^2+n+1, n^2+i] = Z[n^2+i, n^2+i]$

The transformation from the original MIP formulation to its SDP counterpart (See Table 6.5) relies fundamentally on the semi-definite matrix **Z**, within which all original decision variables are reparameterized. This reparameterization serves to streamline the computational aspects of the problem, while retaining the core constraints and objectives. It is also important to note that the indexing scheme for **Z** is rooted in Python's 0-based indexing. Therefore, any implementation would require an adjustment by subtracting one from each endpoint in the matrix.

Initially, a `Model` object named "sdp_gcp" is created to act as the central repository where all variables, constraints, and objectives are consolidated (See Table 6.6) the main decision variable is a Positive Semi-Definite matrix Z, defined using the `model.variable()` method. This matrix has dimensions $(n^2+n+1) \times (n^2+n+1)$ and encapsulates vertex-color assignments as well as additional intermediary variables.

Component	Description
Variables	`Z = model.variable('Z', Domain.inPSDCone(dim))`
Objective	`model.objective(ObjectiveSense.Minimize, Expr.sum(Z.slice([(n**2)+n, n**2], [(n**2)+n, (n**2)+n-1])))`

Constraints	`(1) model.constraint(f'Unique_Color_Usage_v',Expr.sum(Z.slice([n**2+n, n*(v-1)], [n**2+n+1, n*(v-1)+n])), Domain.equalsTo(1.0))`
	`(2) model.constraint(f'Color_i+1_Between_Edge_v,u',sum_expr, Domain.lessThan(0.0))`
	`(3) model.constraint(f'Binary_Enforce_Node_v,i',Expr.sub(Z.index(n**2+n, n*(v-1)+i), Z.index(n*(v-1)+i, n*(v-1)+i)), Domain.equalsTo(0.0))`
	`(4) model.constraint(f'Binary_Enforce_Color_i',Expr.sub(Z.index(n**2+n, n**2+i),Z.index(n**2+i, n**2+i)), Domain.equalsTo(0.0))`
	`(5) model.constraint(f'SDP_Scalar',Z.index(n**2+n, n**2+n), Domain.equalsTo(1.0))`

Table 6.6: Computational Framework of SDP Formulations

The SDP model for the Graph Coloring Problem incorporates multiple types of constraints. **Unique_Color_Usage_{v} Constraint**, it ensures each vertex v gets a unique color. This is done by summing up the relevant row of the matrix Z using `Expr.sum(Z.slice(...))` and applying `Domain.equalsTo(1.0)`.**Color_{i+1}_Between_Edge_{v,u} Constraint** It prevents adjacent vertices v and u from having the same color. The constraint uses an auxiliary expression, `sum_expr`, and enforces `Domain.lessThan(0.0)`. **Binary Constraints**, two additional constraints, `Binary_Enforce_Node_{v,i}` and `Binary_Enforce_Color_{i}`, are implemented to maintain the binary nature of variables. The former uses `Expr.sub(...)` and `Domain.equalsTo(0.0)` for vertex v and color i, while the latter applies a similar approach for each color.

Aspect	Computational Implications
Complexity	The model aims for speed but may not always find the optimal solution due to the relaxation in SDP.
Number of Variables	$(n^2 + n + 1) \times (n^2 + n + 1)$.
Number of Constraints	can grow between $O(n^3)$ to $O(n^3 + 2n + nE)$

Table 6.7: Computational Implications of SDP Framework

By the end of this sequence, the computational framework should accurately represent the original SDP mathematical formulation for the Graph Coloring Problem. While the method is designed for computational efficiency, the complexity of the problem often necessitates the use of sophisticated solver algorithms, particularly for larger graph instances.

6.3 Performance Metrics and Evaluation Criteria

In this study, a comprehensive set of metrics have been introduced to evaluate the performance of various formulations and enhancements in solving the GCP. These metrics allow for objective comparisons between MIP, LP, SDP, and QESDP, and within quadratic reformulations in QEDSP.

Formulation Comparison Metrics

These metrics facilitate objective comparisons across $f \in \{MIP, LP, SDP, QESDP\}$,These comparisons occur between "Processes 1, 3, and 5" (see Figure 6.1), covering Formulation, Optimization, and Performance Evaluation processes.

Solution Time (t^f), the wall-clock time, in seconds, taken to optimize each instance. This captures computational efficiency. Optimality Gap, defined as

$$g^f = \frac{\text{Actual Optimal Value} - Solution^f}{\text{Actual Optimal Value}} \times 100\%$$

This measures how close the approximation is to the true optimum. Smaller values are better. Where 'Solution' The objective value obtained after optimizing the formulation. **Lower Bound (LB^f)** $\forall f \in \{LP, SDP, QEDSP\}$ and **Exact Solution (ES^f)** $\forall f \in \{MIP\}$. The optimality gap directly assess the quality of the approximate solutions. It captured the worst-case deviation between the approximate and actual optima in percentage terms. Complexity, defined as

$$c^f = \text{Number of Constraints}^f$$

This metric quantifies the model's inherent difficulty by examining its constraint structure. A higher c^f value implies more computational challenges due to the increased number of constraints. Together with the execution time, gaps and the compelxity it enables quantifying the tradeoffs between solution optimality and efficiency for each approach.

Quadratic Enhancements Metrics

These metrics extend the evaluation criteria for "Processes 4 and 5" (see Figure 6.1), focusing on the quadratic enhancements to the baseline SDP formulation. Beta serves as an enhancemnt metric, were gamma and theta count as penailites on the enhancemnts. Quantifying the improvement in the gap due to the enhancements is **Beta (β^e)** defined as

$$\beta^e = \frac{g^e}{g^{SDP}}$$

where $\beta^e < 1$ indicates e improved the gap versus SDP, resulting in a lower gap. The computational overhead introduced by the enhancements is **Gamma (γ^e)** defined as

$$\gamma^e = \frac{t^e}{t^{SDP}}$$

where $\gamma^e > 1$ indicates increased computational exhaustion versus the SDP.

In essence, our performance evaluation framework provides a rigorous methodology for comparing MIP, LP, SDP, and QESDP. It enables objective assessment through metrics such as optimality gap, solution time, and complexity. Additionally, the specialized metrics of beta and gamma quantify the effects of quadratic enhancements in extending SDP formulations. This comprehensive evaluation method allows for a nuanced understanding of trade-offs between solution quality and computational time. This approach facilitates a robust, empirical comparison of formulations and enhancements across diverse graph instances.

Chapter 7

Experimentation

This chapter offers a comprehensive examination of the experimental setup and results for solving the Graph Coloring Problem (GCP) through various optimization formulations. We commence by detailing the computational environment on which the experiments are conducted. Specifically, the experimental runs are carried out on a desktop computer equipped with an INTEL Core i5 vPro 8th Gen 1.6 GHz (using a single thread), 2.5 GB of memory. A time limit of 20 minutes is imposed on solving the instances, establishing the boundary conditions for our experiments. Following the hardware and solver settings, we delve into the specific graph instances employed in this study. These encompass benchmark instances, which are discussed in detail in the Experimental Runs and Results section. These instances serve as test beds for evaluating the performance of multiple optimization methods. These methods include Mixed-Integer Programming (MIP), Linear Programming (LP) Relaxation, Semi-definite Programming (SDP) Relaxation, and Quadratic Enhanced SDP (QESDP). The chapter further explores a comparative analysis of these methods in terms of various metrics, such as solution time, gap percentage, and computational complexity. In addition, we scrutinize the impact of various quadratic constraint enhancements on these algorithms. This provides a nuanced understanding of how these enhancements affect key performance metrics. Finally, we present tables and figures that encapsulate the results of our computational experiments. These serve as a foundation for the Analysis and Discussion chapter. This rigorous validation confirms or refutes the preliminary benchmarks, offering a more holistic view.

7.1 Experimental Solver Settings

In this section, the focus is directed towards the termination criteria implemented in the solver. These criteria are formulated to balance between computational efficiency and the attainment of an accurate solution. The mechanism underlying these criteria is predicated on assessing the proximity to either an optimal solution or an infeasible one.

Table 7.1: Termination Tolerances and Conditions

Termination Tolerance	Symbol	Termination Condition
Primal Feasibility Tolerance	ϵ_p	$\rho_p \leq \epsilon_p$
Overall Infeasibility Tolerance	ϵ_i	$\rho_i \leq \epsilon_i$

Diverse optimization formulations necessitate distinct termination settings. The subsequent table delineates how these criteria differ across problem formulations.

Table 7.2: Parameters for Various Formulations

Parameter Name	MIP	LP	SDP
mioMaxTime (Maximum Time Seconds)	1200	-	-
intpntCoTolPfeas	1e-8	1e-8	1e-8
intpntCoTolInfeas	1e-8	1e-8	1e-8

The parameter *mioMaxTime* demarcates the upper time limit imposed on the solver (See Table 7.2). Additionally, the parameters *intpntCoTolPfeas* and *intpntCoTolInfeas* specify the feasibility and infeasibility tolerances.

Table 7.3: Solution Status for Various Methods

Method	Optimal Feasible	Sub-Optimal Feasible	Uncertain	Infeasible
MIP	Yes	Yes	-	Yes
LP	Yes	-	Yes	Yes
SDP	Yes	-	Yes	Yes
QESDP	Yes	-	Yes	Yes

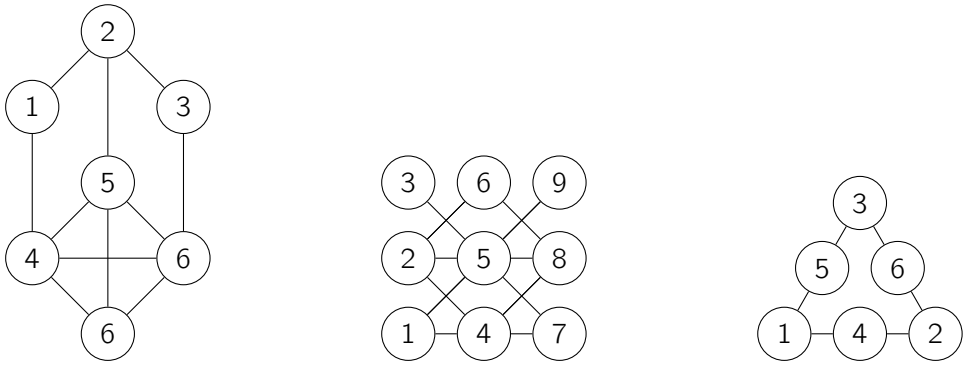
A taxonomy of solution statuses for each formulation is essential to determine the problem feasibility (See Table 7.3). These range from optimal and sub-optimal feasible solutions to those that are uncertain or infeasible.

7.2 Experimental Graph Instances

In this section, we delve into the specific graph instances used for computational experiments. We employed distinct categories of instances benchmark instances. Benchmark instances were sourced from established research in the optimization and Graph Coloring Problem (GCP) domains, particularly from the work of (Trick et al. 2023) and (Caramia et al. 2023).

Benchmark Instances

These instances have been meticulously studied in academic and research settings, providing a reliable and standardized yardstick against which the efficiency of our algorithm can be gauged.



(a) A Mycielski Graph

(b) A 3×3 Queen's graph

(c) Insertions Graph of Order 3

Figure 7.1: Benchmark Graph Instances

Benchmark Graphs (see Figure 7.1) consist of, Mycielski graphs, Queen's graphs, 2-Insertions graphs of order 3, and 1-FullIns graphs of order 3 serve as benchmarks for studying graph coloring algorithms. Mycielski graphs lack triangles and have increasing chromatic numbers with more vertices. Queen's graphs model chessboards and have applications in code register allocation. 2-Insertions graphs add complexity by introducing two additional vertices for each original edge. 1-FullIns graphs further raise complexity with a star-like topology around original edges.

Metadata of Instances

Observing the meta data about each type of graph, and their intrinsic properties can give us insights on the grouping of graphs with similar characteristics that may behave similarly.

Table 7.4: Metadata of Benchmark Instance Graphs

Instance	Vertices	Edges	Size	Density
myciel3	11	20	55	0.364
myciel4	23	71	253	0.281
queen5_5	25	160	300	0.533
queen6_6	36	290	630	0.460
2-Insertions_3	37	72	666	0.213
1-FullInsertions_3	30	100	435	0.459

The metadata presented in Table 7.4 reflects a range of graph instances with varying levels of complexity, size, and density. The instances generally fall into categories of either being small but densely connected, or large but sparsely connected. For example, smaller instances often show higher density values, implying a higher degree of connectivity relative to their size. In contrast, larger instances tend to exhibit lower density values, suggesting that despite their size, they are not as densely connected. This variety in attributes makes the data set well-suited for a broad range of bench-marking and comparative analyses, as it encompasses instances with differing scales and structural complexities.

7.3 Experimental Runs and Results

In this section, the study conducts a comprehensive experimental evaluation of a Computational Framework on benchmark instances are sourced from well-known libraries and are deterministic in nature; thus, they undergo a single simulation run. A more in-depth analysis of these results is planned for the next chapter.

Performance Across Formulations

We present comparative analysis of the performance of various optimization formulations. Our benchmark includes the Mycielski and Queen graphs, which vary in their complexities and optimal colorings. We employ different methods, including MIP, LP Realxation, SDP

Relaxation to investigate their performance in terms of approximate coloring, solution time, gap percentage, and overall complexity.

Table 7.5: Benchmark Performance for Mycielski Instances

Instance	Actual Coloring	Formula	Status	Approximate Coloring	Solution Time (S)	Gap (%)	Complexity
Myciel3	4	MIP	Optimal F.	4.000	0.24	0.00	231
		LP	Optimal F.	2.000	0.04	50.00	231
		SDP	Optimal F.	2.305	0.09	42.37	364
		QESDP	Optimal F.	3.212	0.44	19.69	485
Myciel4	5	MIP	Optimal F.	5.000	13.50	0.00	1656
		LP	Optimal F.	2.000	0.09	60.00	1656
		SDP	Optimal F.	2.280	2.73	54.40	2209
		QESDP	Optimal F.	3.369	12.11	32.62	2738

The performance data for Mycielski instances (See Table 7.5) indicate clear distinctions between different formulations. MIP performs well in terms of accuracy, with a 0% gap, but has a mean solution time of approximately 7 seconds. LP is much quicker but yields a high gap range of 42-50%, suggesting reduced accuracy. SDP averages 1.4 seconds and a 48% gap, while QESDP has a mean time of 0.44 to 12.11 seconds with a 19.69% to 32.62% gap. In this context, QESDP appears to be a promising alternative, achieving lower gap percentages, albeit at the cost of slightly increased solution times.

Table 7.6: Benchmark Performance for Queens Instance

Instance	Actual Coloring	Formula	Status	Approximate Coloring	Solution Time (S)	Gap (%)	Complexity
Queen5_5	5	MIP	Optimal F.	5.000	12.41	0.00	4025
		LP	Optimal F.	2.000	0.09	60.00	4025
		SDP	Optimal F.	2.283	6.20	54.33	4676
		QESDP	Optimal F.	4.422	269.30	11.44	5301
Queen6_6	7	MIP	Feasible	7.000	1200.78	0.00	10476
		LP	Optimal F.	2.000	0.57	71.43	10476
		SDP	Optimal F.	2.281	114.42	67.42	11809
		QESDP	Optimal F.	5.389	6636.48	23.01	77539

Queen instances reveals a different performance landscape (See Table 7.6). MIP is slow but extremely accurate, consistently achieving a 0% gap. LP and SDP are faster but less accurate. QESDP takes longer, especially for Queen6_6 at 6636.48 seconds, but offers a

more accurate solution with 11.43% to 23.0% gap. Interestingly, QESDP takes considerably longer but offers a much smaller gap, making it potentially more accurate for these complex graphs. The computational complexity for Queen graphs is notably higher, particularly when QESDP is applied, suggesting a trade-off between accuracy and computational resources.

Table 7.7: Benchmark Performance for Insertion Instance

Instance	Actual Coloring	Formula	Status	Approximate Coloring	Solution Time (S)	Gap (%)	Complexity
2-Ins_3	4	MIP	Optimal F.	4.000	666.00	0.00	2701
		LP	Optimal F.	2.000	0.11	50.00	2701
		SDP	Optimal F.	2.269	29.75	43.27	4108
		QESDP	Optimal F.	2.915	105.05	26.75	35447
1-FullIns_3	4	MIP	Optimal F.	4.000	435.00	0.00	303
		LP	Optimal F.	2.000	0.23	50.00	303
		SDP	Optimal F.	2.274	9.09	43.16	3961
		QESDP	Optimal F.	3.874	76.38	3.15	3701

The performance metrics for Insertion instances (See Table 7.7) suggest varying efficiencies across different formulations. MIP, while accurate with a 0% gap, shows a considerable mean solution time, especially for 2-Insertions_3. LP is the fastest but suffers from high gap percentages. SDP and QESDP provide alternative options, balancing speed and accuracy to varying extents. Notably, QESDP appears to be particularly efficient in terms of the gap for 1-FullInsertions_3. Overall, these instances also display a wider range of computational complexities, especially with QESDP, suggesting their computational demands can be significantly high.

Affects of Quadratic Constraints Enhancements

We investigate the effects of several quadratic constraint enhancements, collectively called QESDP, on benchmark instances. These enhancements extend the standard SDP model and come in various categories, including 'Unique Vertex Coloring', 'Adjacent Vertex', and 'Global to Vertex Color'. Some categories are combinations, like "UniqGlobal," which blends Unique Vertex Coloring with Vertex to Global Coloring. GAU is an umbrella term covering all these enhancements. We apply these enhancements to Mycielski, Queen's and Insertion graphs to assess their impact on key metrics like Beta and Gamma which serve as ratios

between standard SDP solutions and additional enhancement measures.

Table 7.8: Affect of Enhancements Mycieliski's Instances

Instance	Metric	Enhancements					
		Unique Vertex Coloring	Adjacent vertex	Vertex to Global Color	Unique Global	Adjacent Global	GAU
Myciel3	Beta	0.985	1.002	0.964	0.950	0.512	0.465
	Gamma	3.034	2.001	1.418	3.206	2.196	5.600
	Lower B.	2.330	2.303	2.367	2.390	3.132	3.212
Myciel4	Beta	0.996	0.997	0.986	0.980	0.600	0.603
	Gamma	9.621	3.268	1.640	11.142	4.444	26.780
	Lower B.	2.292	2.288	2.319	2.335	3.369	3.361

In the context of Mycielski graphs (See Table 7.3), the data show that the 'Unique Vertex Coloring' enhancement is particularly effective in optimizing the gap between the lower and upper bounds. This is evident from Beta values of 0.985 for Myciel3 and 0.996 for Myciel4, which are both less than 1. However, this advantage comes at the cost of computational overhead, as reflected by Gamma values greater than 1. Specifically, the Gamma values for Myciel3 and Myciel4 are 3.034 and 9.621, respectively. The Lower Bound metric shows a relatively tight range, indicating moderate sensitivity to the choice of enhancement.

Table 7.9: Affect of Enhancements on Queen's Instances

Instance	Metric	Enhancements					
		Unique Vertex Coloring	Adjacent vertex	Vertex to Global Color	Unique Global	Adjacent Global	GAU
Queen5_5	Beta	0.998	1.002	0.989	0.984	0.291	1.841
	Gamma	7.883	33.810	1.333	11.221	43.463	51.918
	Lower B.	2.289	2.277	2.312	2.328	4.422	0.697
Queen6_6	Beta	1.000	1.002	0.995	0.991	0.475	0.341
	Gamma	39.552	45.322	1.590	30.277	38.921	58.262
	Lower B.	2.281	2.262	2.304	2.322	4.756	5.389

Queen instances present a different dynamic compared to Mycielski graphs (See Table 7.3).

For Queen instances (Queen5_5 and Queen6_6), Beta values are closer to 1, indicating modest improvements in the optimality gap. The Gamma values are notably higher, particularly when Adjacent Vertex is used (33.810 for Queen5_5 and 45.322 for Queen6_6). These high Gamma values point to significant computational challenges. The Lower Bound varies more significantly, from 0.697 to 5.389, indicating that different enhancements may yield different benefits but at high computational costs.

Table 7.10: Affect of Enhancements on Insertion's Instances

Instance	**Metric**	**Enhancements**					
		Unique Vertex Coloring	**Adjacent vertex**	**Vertex to Global Color**	**Unique Global**	**Adjacent Global**	**GAU**
2-Insertions_3	Beta	0.994	0.998	0.971	0.959	0.627	0.618
	Gamma	57.671	2.249	1.352	70.031	3.531	132.499
	Lower B.	2.280	2.272	2.319	2.340	2.915	2.931
1-FullIns_3	Beta	0.994	0.995	0.974	0.970	0.073	0.045
	Gamma	18.391	3.248	2.310	26.407	8.402	113.932
	Lower B.	2.284	2.283	2.319	2.325	3.874	3.922

Analysis of insertion graphs reveals a nuanced pattern of performance metrics(See Table 7.3). In terms of Beta values, there is a moderate improvement observed, especially notable in the "Unique Vertex Coloring" category. These values are particularly close to 1 for Insertion instances (2-Insertions_3 and 1-FullIns_3), ranging from 0.994 to 0.998, indicating minor but appreciable gap improvements. Conversely, Gamma values present a more varied picture: while some enhancements, like 2-Insertions_3, result in high computational costs (e.g., 57.671 for 'Unique Vertex Coloring'), the range across enhancements is wide, stretching from 2.249 to as high as 132.499. This suggests a need to consider the computational trade-offs when implementing these enhancements.

Optimal Enhancement Set Leading to Tightest Lower Bound

The pursuit of identifying the most effective enhancement sets for optimizing the Lower Bound metric in graph instances serves as a cornerstone in computational graph theory. To this end, the forthcoming (See Table 8.6) furnishes an intricate assessment of a multitude

of metrics for different graph instances. The table scrutinizes the performance of various enhancement sets, particularly focusing on their influence on the Lower Bound metric. Within the encapsulated metrics—SDP Lower Bound, QESDP Lower Bound, Unique Vertex Coloring, Adjacent Vertex, Vertex to Global Color, Beta and Gamma we seek to elucidate which combination of enhancements yields the tightest Lower Bound while accounting for computational efficacy. This section delves into the nuance and interplay among these variables, elucidating distinct trends across the Mycielski and Queen graph instances.

Table 7.11: Comparision of Performance of Optimal Enhancement Set

Instance	**SDP Lower Bound**	**QESDP Lower Bound**	**Unique Vertex Coloring**	**Adjacent vertex**	**Vertex to Global Color**	**Beta**	**Gamma**
Myciel3	2.305	3.212	✓	✓	✓	0.465	5.600
Myciel4	2.280	3.369		✓	✓	0.600	4.444
Queen5_5	2.283	4.422		✓	✓	0.291	43.463
Queen6_6	2.281	5.389	✓	✓	✓	0.341	58.262
2-Ins_3	2.269	2.915		✓	✓	0.627	3.531
1-FullIns_3	2.274	3.874		✓	✓	0.073	8.402

The Mycielski graphs benefit from Unique Vertex Coloring, as indicated by low Beta values and moderate Gamma costs. The density in these graphs (Myciel3: 0.364, Myciel4: 0.281) seems to correlate positively with their performance. Queen graphs, with higher densities (Queen5_5: 0.533, Queen6_6: 0.460), show marginal Lower Bound improvements mainly with 'Adjacent Vertex' and 'Vertex to Global Color' enhancements but at a significant computational toll. The Insertion instances, particularly 2-Ins_3 with lower density (0.213), slightly improve the Lower Bound without significant computational costs when 'Adjacent Vertex' and 'Vertex to Global Color' are applied. 1-FullIns_3, with a higher density (0.459), also improves but at a greater computational cost, as evidenced by a higher Gamma value. So, Mycielski graphs are more responsive to 'Unique Vertex Coloring', whereas Queen graphs are computationally expensive, particularly with 'Adjacent Vertex'.

Chapter 8

Analysis and Discussion

This chapter aims to delve into a comprehensive evaluation and discussion of these different approaches, specifically focused on their performance metrics like solution time, optimality gap, and complexity. Employing both empirical and statistical analysis, the chapter will dissect the behavior of these algorithms under different conditions, such as varying densities and families of graph instances. Not only will we provide a thorough correlation analysis of performance metrics but also critically evaluate the comparative efficacy of relaxation methods like SDP and QESDP. We will further investigate the impacts of enhancements to these methods and examine how these variations influence the performance metrics, ultimately aiming to guide practitioners in choosing the most suitable approaches for their specific needs.

To enrich the insights gained through this analysis, the chapter will utilize extensive computational experiments substantiated by graphical and tabular representations. We will employ statistical tests to validate the observed trends, thereby providing a robust foundation for the conclusions drawn. Moreover, the chapter will conclude with actionable recommendations that offer a balanced trade-off between computational efficiency and solution quality.

8.1 Correlation Analysis

In this section, we perform a comprehensive correlation analysis to examine the relationships among various performance metrics across different methods. This quantitative analysis serves as a foundation for understanding the complexity and nuances of different algorithmic approaches and their performance characteristics.

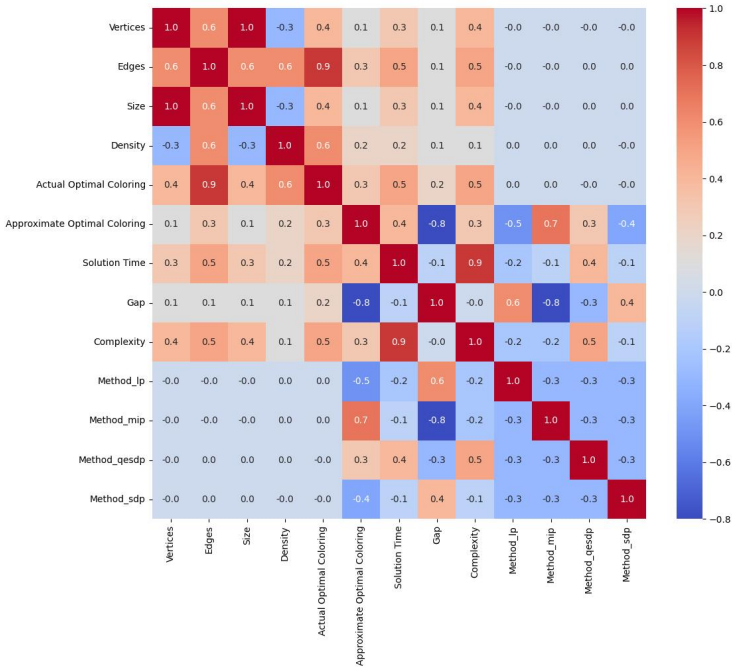


Figure 8.1: Correlation Heat Map of Performance's Across Metrics

In the correlation analysis (See Figure 8.1), several significant insights were observed. The most compelling is the strong positive correlation of 0.9 between the number of edges and the actual optimal coloring, as well as between solution time and complexity. These high correlations suggest that as the number of edges in a graph increases, the actual optimal coloring similarly increases, and that the time needed to find a solution scales with the problem's complexity. Also, the actual optimal coloring and complexity shared a moderate correlation of 0.5, suggesting that more complex graphs usually require more colors for valid coloring, albeit influenced by other variables. Notably, the gap metric displayed low correlations, around 0.1 with most other variables, pointing to its relative independence and the need for further investigation. The formualtions used MIP, LP, SDP, and QESDP showed almost zero correlation with other variables except the performance metrics, hinting that their influence might be non-linear or more complex. A significant negative correlation

of -0.8 was observed between approximate optimal coloring and gap, suggesting that lower gap values generally yield better coloring approximations. This leads us to the conclusion that analyzing the performance metrics among different groupings would give us more confidence in reaching to conclusions.

8.2 Evaluating the Performance of Optimal Computation Method Versus Relaxation Techniques

In this section, we critically assess the performance metrics, including solution time, optimality gap and complexity across different relaxation techniques and optimal computation formulations for various families of graphs. The analysis is based on extensive computational experiments and is substantiated by three distinct figures that spotlight different aspects of these formulations' efficacy.

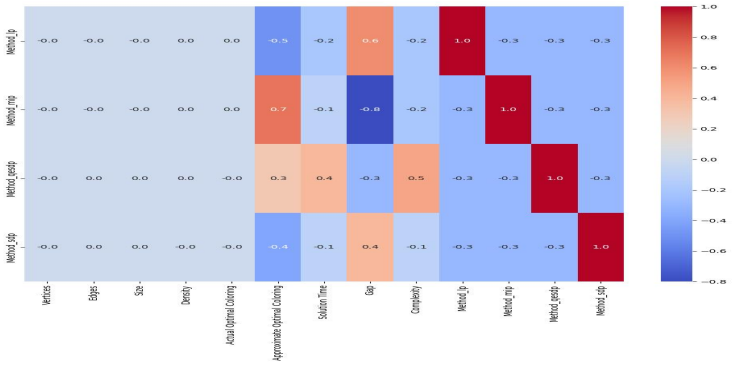


Figure 8.2: Correlation Heat Map of Performance's Across Metrics

In reference to the Correlation Map (See Figure 8.2), the Correlation Map reveals several pivotal insights into the performance characteristics and computational demands of various solving methods. QESDP exhibits a positive correlation of 0.4 with solution time and 0.5 with complexity, indicating its computationally intensive nature. However, it correlates negatively at -0.3 with the gap metric, suggesting that this method is associated with smaller gaps, and consequently, potentially more accurate solutions. SDP, in contrast, demonstrates a minimal impact on both solution time and complexity, as suggested by its near-zero negative correlations of -0.1 with both these metrics. Nevertheless, it shows a positive correlation of 0.4 with the gap, suggesting a propensity for larger gaps and, therefore, less accurate solutions. Both methods display a negative correlation of -0.3 among each other, indicating

their mutual exclusivity within the model. Moreover, QESDP shows a positive correlation of 0.3 with 'Approximate Optimal Coloring,' implying its potential efficacy in generating better approximations. Conversely, SDP registers a negative correlation of -0.4 with 'Approximate Optimal Coloring,' indicating potential ineffectiveness in this domain.

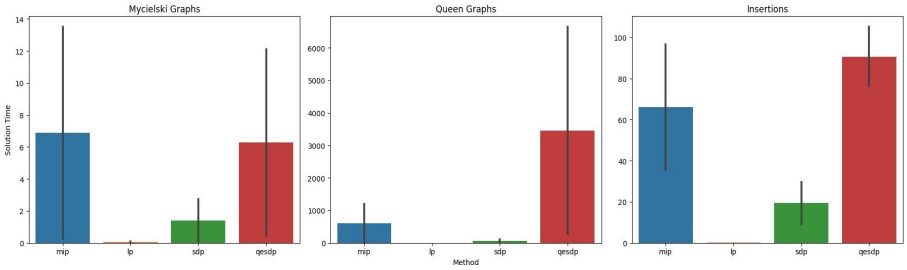


Figure 8.3: Solution Time (Seconds) Across Formulations faceted per Benchmark Families

In the comparative analysis of solution time across diverse graph families (See Table 8.3) for the set of formulations. Mycielski graphs, characterized by densities of approximately 0.28 to 0.36 exhibited a mean solution time of around 7 seconds, but varies up to 14 seconds. LP, however, was remarkably efficient, with an average time close to 0.3 seconds, suggesting it as a preferred choice for such low-density graphs based on solution time. On the other end of the spectrum, Queen graphs with densities around 0.46 to 0.53 presented challenges, particularly MIP and QESDP, which had mean solution times of 500 and 3500 seconds, respectively. LP maintained its efficiency, completing in about five second. In the case of Insertion graphs, with densities hovering around 0.213 to 0.45, LP again outperformed all other methods, requiring less than 5 seconds on average. While SDP and QESDP yielded intermediate performance levels. The variation bars, representing the times for individual graphs within each family, underscored the sensitivity of these methods to specific graph attributes. Suggesting that on average MIP and QESDP are computationally exhaustive.

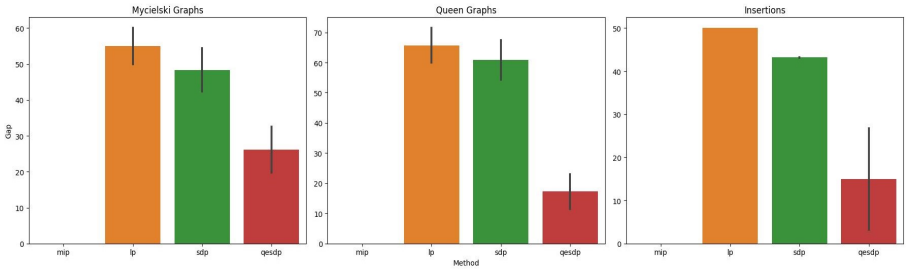


Figure 8.4: Gap (%) Across Formulations faceted per Benchmark Instances

In the evaluation of optimality gaps graph families (See Figure 8.4), the formulations demonstrated a range of performances. It is noteworthy that the MIP method, by its exact nature, yielded a zero-gap, serving as the optimal benchmark for the other methods which provided lower-bound solutions. For Mycielski graphs, LP registered a gap of approximately $55\% \pm 5\%$, while SDP and QESDP followed with gaps of $48\% \pm 7\%$ and $26\% \pm 8\%$, respectively. In the Queen graph family, LP showed a slightly higher gap of $65\% \pm 5\%$, and SDP had a gap of $60\% \pm 8\%$, whereas QESDP was notably more accurate with a gap of $16\% \pm 4\%$. Finally, in the case of Insertion graphs, LP had a gap close to 50%, SDP yielded 43%, and QESDP performed exceptionally well with a gap of $14\% \pm 12\%$. An additional layer of complexity is introduced when considering the densities. Interestingly, lower-density graphs like Insertions saw some of the best results with QESDP, indicating the method's potential for optimally solving sparser graph instances. Conversely, higher-density graphs like the Queen family demonstrated a broader gap range across methods, highlighting the inherent complexities of solving denser problems. This suggests that the effectiveness of these methods may not be uniformly applicable but can be considerably influenced by the density of the graph under consideration.

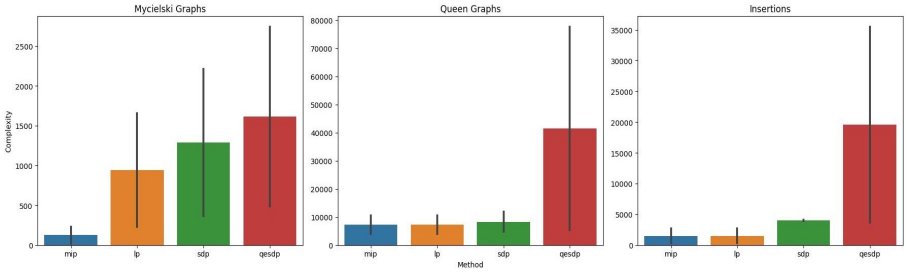


Figure 8.5: Complexity (Number of Constraints) Across Formulations faceted per Benchmark Families

The study identifies a general trend in computational complexities across methods and graph families, with QESDP usually showing the highest complexity (See Table **??**). For Mycielski graphs, complexities range from 150 to 3,000 when moving from MIP to QESDP. Queen graphs experience a complexity surge to 40,000 in QESDP, compared to 5,000 to 10,000 in MIP. Insertion graphs have complexities below 2,500 for MIP, surging to around 18,500 in QESDP. The density of the graph influences the complexity, especially for Queen graphs. The findings suggest a trade-off between solution optimality and computational effort in method selection.

8.3 Comparative Efficacy of SDP and QESDP Relaxation on Lower-Bound Approximations

In this section, we examine the comparative efficacy of the two relaxation methods SDP and QESDP in approximating the lower bound of benchmark instances with different groupings. We employ a dual-metric approach, examining not only the effectiveness of each method in finding a tight lower bound but also the computational time required to arrive at these solutions. Furthermore, we preform some statistical tests to be able to generalize on instances with similar intrinsic properties of instances.

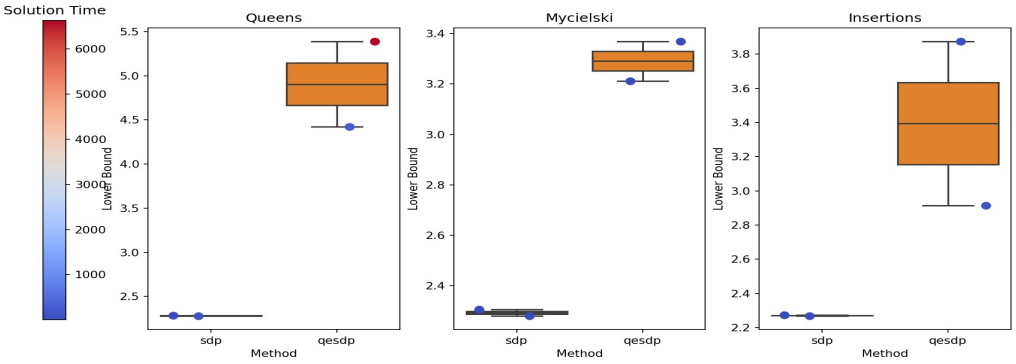


Figure 8.6: Solution Time (Seconds) and Lower Bound Efficacy on SDP and QESDP Grouped by Family Instances

A comprehensive lens into the differing performance profiles of SDP and QESDP algorithms across the instance families: Queens, Mycielski, and Insertions (See Figure 8.6). One striking uniformity across all families is SDP's consistent lower bound range from 2 to 2.5, alongside relatively fast solution times spanning 50 to 250 seconds. This makes SDP a universally swift yet moderately accurate option. However, the discrepancies become salient when we delve into QESDP's performance for each family. For Queens instances, QESDP remarkably elevates the lower bounds to 4.4–4.5, a feat achieved at the expense of vastly increased computational time between 500 and 6000 seconds. This indicates that for highly complex structures like Queens, QESDP offers accuracy but demands computational resources. In Mycielski graphs, QESDP provides a lower bound range of 3.2–3.4 with moderate solution

times (50–500 seconds. The relative balance between time and accuracy here suggests that QESDP can be a viable option for moderately complex instances, offering a better trade-off than in the Queens family. For Insertions, the lower bounds achieved by QESDP are variable (2.9–3.9, yet the solution times remain within 50–500 seconds. This makes QESDP a flexible but somewhat unpredictable choice for graphs in this family, introducing an element of variability in its performance.

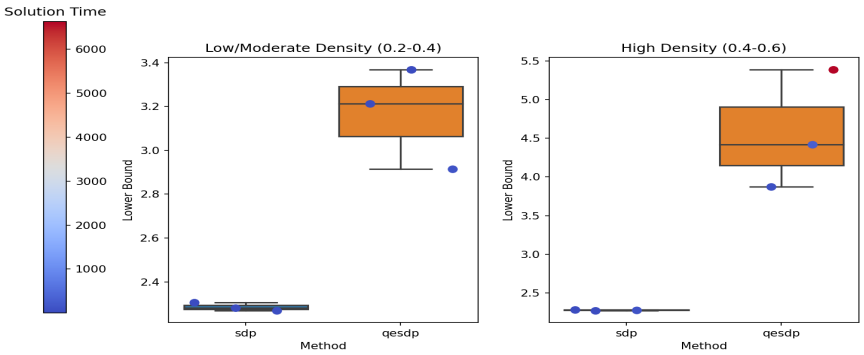


Figure 8.7: Solution Time (Seconds) and Lower Bound Efficacy on SDP and QESDP of Low and High Benchmark Instances

The comparative performance of SDP and QESDP across two density categories (See Figure 8.7). In the low/moderate density range, represented by instances such as Mycielski's and half-insertion graphs, SDP yields lower bounds between 2.1 and 2.3 with solution times spanning 50 to 250 seconds. Conversely, QESDP returns a more robust lower bound range of 2.9 to 3.38, but takes significantly longer with solution times between 250 to 1000 seconds. In the high-density category, characterized by queen and full-insertion graphs, SDP maintains a lower bound range of 2.3 to 2.4 and impressively quick solution times from 0 to 250 seconds. QESDP, on the other hand, delivers a wider lower bound spread from 3.8 to 5.38 but incurs a hefty computational cost, taking 1000 to 6000 seconds.

8.3.1 Variance Analysis of Lower Bound Approximation by SDP and QESDP

In our examination of the comparative lower bounds generated by SDP and QESDP across family instances and density-level groupings, we aim to strengthen these empirical observations through ANOVA (Analysis of Variance) tests, to assess the statistical significance of the differences observed in lower bound approximations between SDP and QESDP. The

tests are essential for determining the likelihood of generalization based on the observed differences. More emphasize on formulation of the test and empirical results can be found at (See Appendix A) .

The empirical and statistical analyses provided compelling evidence of significant differences in the performance of SDP and QESDP methods across different family instances (See Table A.1) and density groups (See Table A.2). Particularly for Queens and Mycielski instances as well as Low/Moderate and High Density groups, the p-values were below the designated alpha level of 0.05, confirming that the differences in lower bound approximations were statistically significant. Insertions, however, did not demonstrate statistical significance, indicating more flexibility in method selection for these instances. These robust statistical validations not only corroborate our initial empirical observations but also guide practitioners in method selection for different instances.

For the Family Instances types of Queens, Mycielski, and Insertions, the statistical tests show varying degrees of significance. Queens and Mycielski instances demonstrated statistically significant differences between SDP and QESDP methods. However, the Insertions instance type did not indicate a significant difference. For practitioners working with Family Instances, Queens and Mycielski instances would benefit from a more tailored approach to method selection, while Insertions offer greater flexibility, allowing either method to be used without a significant impact on lower bound approximations. Both Low/Moderate Density and High Density groups showed statistically significant differences between SDP and QESDP. The Low/Moderate Density group had a higher degree of statistical significance compared to the High Density group. For tasks that involve instances from different density groups, the choice of approximation method is crucial. Both density groups showed statistically significant results, suggesting that the approximation methods should be carefully selected based on the density of the instances involved. In the case of Low/Moderate Density instances, the choice becomes even more critical given the higher statistical significance.

8.4 Impact of Quadratic Reformulations by QESDP on Performance Metrics Compared to SDP

In this section, the study investigates the effects of three quadratic enhancements—'Vertex unique color' (E1), 'Adjacent vertex' (E2), and 'Vertex to global color' (E3) on two metrics, Beta and Gamma. These metrics are used for comparing solution time and quality gap against a baseline set by Semidefinite Programming (SDP). The study employs Exploratory Data Analysis and Analysis of Variance (ANOVA) tests to validate the impact of these enhancements, then Multiple Linear Regression analysis further substantiates the findings of the effects of enhancement on performance metrics. The enhancements are shown to have the potential to improve lower-bound approximations on SDP solutions. Specifically, a lower Beta value suggests better bound quality, while a lower Gamma value indicates less computational effort required.

Box-plot Analysis of Family-Based Grouping Instances

we explore the influence of quadratic enhancements within specific families of graph instances which are the Mycielski, Queens and Insertions. This family-based classification serves as our initial lens for evaluating the effectiveness of enhancements (E1), (E2), and (E3) across diverse instance sets.

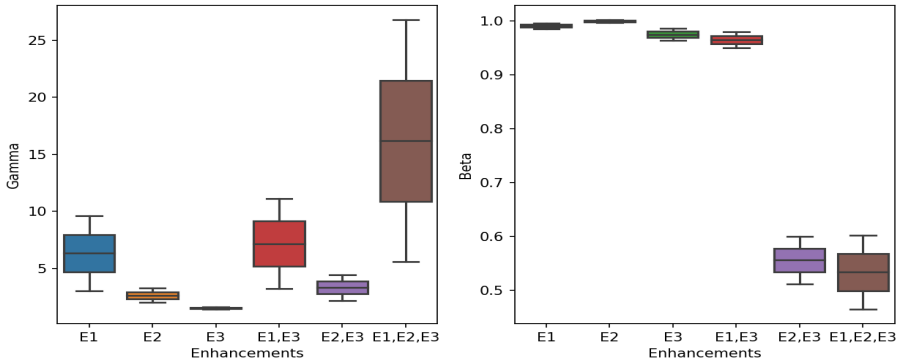


Figure 8.8: Affect of Quadratic Enhancements on Mycielski Instances

In our examination of Mycielski instances (See Figure8.8), our data reveals intriguing insights into the effects of quadratic enhancements on performance metrics Beta and Gamma. When enhancements (E1) or (E2) are applied independently, the ratio shows no significant improvement, hovering around one relative to the baseline SDP. Adding (E3), however, leads

to a meaningful decline in the average Beta to 0.95. Among combinations of enhancements, (E3) with (E1) results in an average $= 0.93 \pm 0.15$. The most noteworthy gains are seen when combining (E2) and (E3), plunging the average Beta to $= 0.55 \pm 0.5$, and the all-inclusive set of (E1E2E3) trails closely with $= 0.53 \pm 0.7$. These combinations provide the tightest lower bounds. Turning our attention to the Gamma metric, individual enhancements show a descending trend: $\gamma = 6 \pm 3$ for (E1), $\gamma = 2 \pm 0.5$ for (E2), and $\gamma = 1 \pm 0.1$ for (E3), which stands as the most efficient. Nevertheless, some combinations such as (E1E3) and (E2E3) disrupt this trend by significantly increasing computational time, becoming up to 25 times more demanding.

When When evaluating performance metrics, several key strategies become apparent. If you seek a balanced approach between quality and computational efficiency, (E3) is the best option, providing reasonable improvements in Beta while keeping Gamma low. For those aiming for the best quality, combinations of (E2) and (E3) or the full set of (E1E2E3) should be considered, although these options require higher computational resources and affect Gamma negatively. If computational power is limited, (E3) is again the most efficient choice to improve Beta without a large computational toll. Combinations like (E1E3) and (E2E3) should be used carefully, as they negatively impact Gamma and are only justifiable when the increase in Beta is worth the computational cost. Features like 'Adjacent vertex' (E2) and 'Vertex to global color' (E3) are crucial for tightening the solution space and improving quality.

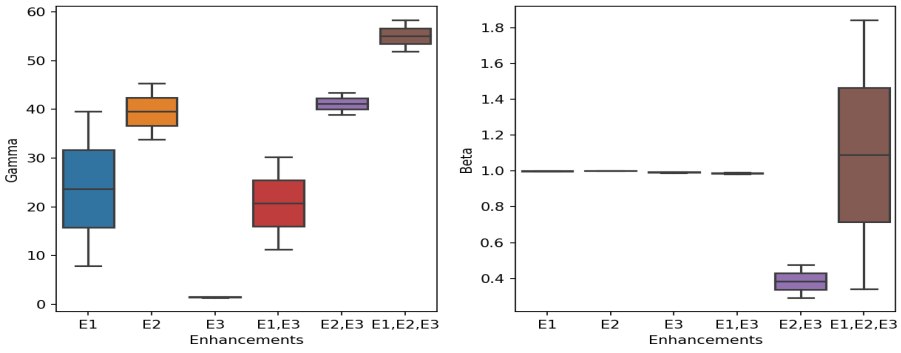


Figure 8.9: Affect of Quadratic Reformulations on Queens Instances

In our investigation of Queens instances (See Figure 8.9), we discern compelling trends across the Beta and Gamma metrics. For the Beta metric, individual enhancements such as

(E1), as well as the specific combination (E1E3), offer only minimal improvements, maintaining Beta values around $\beta \approx 1$, which signifies marginal gains over the baseline SDP. However, a remarkable decline occurs when (E2) and (E3) are combined, plummeting the Beta value to $\beta = 0.4 \pm 0.1$, thereby indicating a significant enhancement in the quality of lower bounds. Transitioning to the Gamma metric, individual enhancements yield varying results: $\gamma = 25 \pm 15$ for (E1) and $\gamma = 40 \pm 5$ for (E2). Yet, (E3) stands out as it drastically reduces the Gamma value to $\gamma = 1 \pm 0.1$. Interestingly, when all enhancements are combined, the Gamma value peaks at $\gamma = 55 \pm 4$, indicating the highest computational demands among the examined sets. This corroborates with our observation that while (E2) and (E3) substantially improve Beta values, they conversely escalate the computational burden, as evident from the elevated Gamma values.

In the comprehensive evaluation, the combinations $E2E3$ and $E1E2E3$ perform best in achieving the lowest possible Beta values. Specifically, $E2E3$ emerges as the most effective combination, delivering a Beta value of $\beta = 0.4 \pm 0.1$. However, these combinations come at the cost of significantly higher computational resources; for instance, the $E1E2E3$ set can be up to 60 times more computationally demanding than individual constraints. If computational efficiency is a priority, $E3$ stands out as the most efficient choice, minimizing Gamma with values of $\gamma = 1 \pm 0.1$. For those seeking a balanced approach between lower-bound quality and computational efficiency, $E3$ proves to be the ideal choice. It offers a substantial reduction in Beta while also optimizing Gamma, thus providing a high-quality solution without imposing a significant computational burden. Conversely, if one has abundant computational resources, the $E2E3$ combination could be a viable choice for achieving the lowest possible Beta values, albeit at the cost of increased computational effort. It is important to note that the enhancements 'Adjacent vertex' and 'Vertex to global color' significantly narrow the feasibility region, but at the expense of substantially increasing computational demands, sometimes up to 20 times more than when applying individual constraints. For Queens instances, $E2E3$ offers the best balance between lower-bound quality and moderate computational costs.

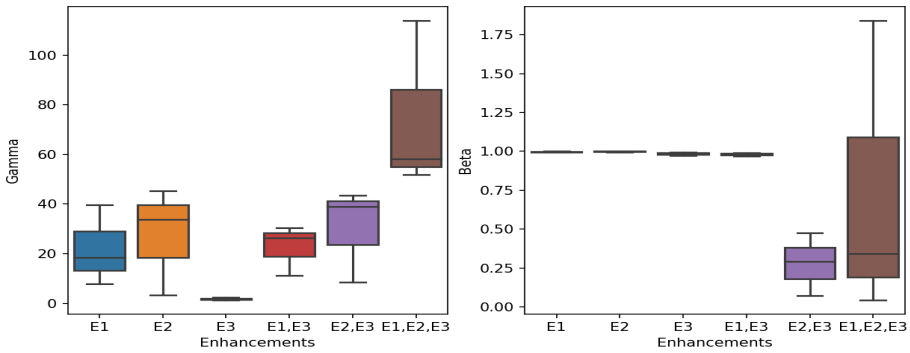


Figure 8.10: Affect of Quadratic Reformulations on Insertion Instances

In our study of Insertions instances (See Figure 8.10), we notice a nuanced impact of the enhancements on both Beta and Gamma metrics. When employing either individual enhancements or specific combinations such as (E1) and (E1E3), the Beta metric registers minimal improvement over the baseline SDP, hovering around $\beta \approx 1$. In stark contrast, the application of (E2) and (E3) together brings about a significant reduction in Beta to $\beta = 0.36 \pm 0.26$. This is further marginally optimized when all enhancements (E1E2E3) are employed together, lowering the Beta value to $\beta = 0.34 \pm 0.25$. Turning our attention to the Gamma metric, individual constraints (E1), (E2), and (E3) demonstrate $\gamma = 38 \pm 16$, $\gamma = 1.8 \pm 0.3$, and $\gamma = 1.5 \pm 0.2$ respectively. This highlights the computational efficiency of (E2) and (E3) when used individually. However, when these are combined, the Gamma values experience a significant increase. For instance, (E2E3) logs $\gamma = 5.0 \pm 1.2$, and the all-encompassing set of (E1E2E3) pushes the metric to an astonishing $\gamma = 120 \pm 10$.

Our comprehensive study of Insertions instances unveils nuanced trade-offs among the metrics of Beta and Gamma when applying quadratic enhancements (E1), (E2), and (E3). Individually, enhancements (E2) and (E3) yield significant improvements in gap quality, as evidenced by their low Beta values, while also maintaining low computational overhead when considered through the Gamma. If the primary objective is to attain the tightest possible lower bounds (Beta), the all-encompassing set of (E1E2E3) is recommended. While this approach optimizes Beta, it comes at the expense of heightened computational load, reflected in an elevated Gamma value. On the other hand, if computational efficiency is the overarching priority, singular application of either (E2) or (E3) provides the most streamlined option. Notably, (E3) outperforms its counterparts by yielding the lowest Gamma values. This leads

to strategic deployment of these quadratic enhancements is imperative for optimizing the trade-offs between lower bound quality and computational resource requirements.

Box-plot Analysis of Density-Based Grouping Instances

To generalize our observations and findings, it is crucial to extend our analysis beyond merely the family-based grouping of instances. A compelling alternative for this broadening scope is to consider the intrinsic properties of the instances—particularly the vertices and edges—which lead us to density as a distinctive form of classification. This density-based grouping serves as an additional axis for our investigation, allowing us to discern patterns and trends that may be otherwise obscured when merely classifying instances based on their families, low/moderate density instances are Myciel3, Myciel4, and 2-Insertions_3, while the high density instances are 1-FullIns_3, Queen5_5, and Queen6_6.

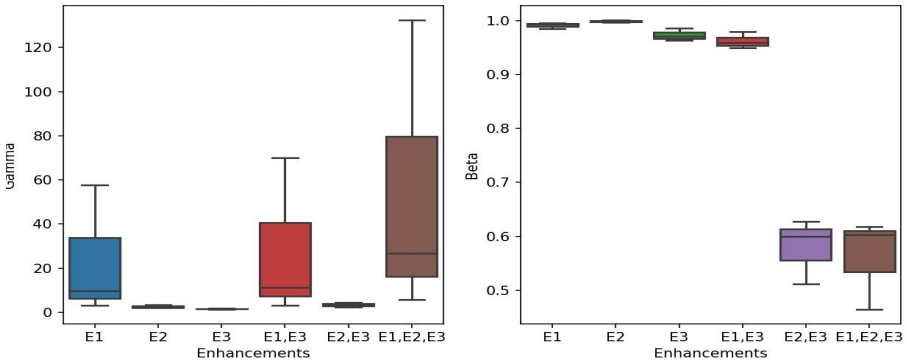


Figure 8.11: Affect of Quadratic Reformulations on Low to Moderate Density Instances

In our analysis of low to moderate density instances, (See Figure 8.12), several intriguing trends emerge in the Beta and Gamma metrics. When evaluating individual enhancements (E1), (E2), (E3), or the specific combination (E1E3), Beta values remain relatively close to the SDP baseline, hovering around $\beta \approx 1$. However, a substantial drop is observed when (E2) and (E3) are combined, yielding an average $\beta = 0.59 \pm 0.08$. Notably, the full set of enhancements (E1), (E2), and (E3) offers a comparable mean Beta value of $\beta = 0.6 \pm 0.1$. For the Gamma metric, individual enhancements provide significantly different values: $\gamma = 15 \pm 12$ for (E1), $\gamma = 1.8 \pm 0.3$ for (E2), and $\gamma = 0.8 \pm 0.06$ for (E3). Combining enhancements amplifies these Gamma values considerably, with (E1E3) reaching $\gamma = 18 \pm 16$ and (E2E3) at $\gamma = 2.9 \pm 0.4$. Employing the complete set of (E1E2E3) further exacerbates this trend, escalating to $\gamma = 35 \pm 25$.

Based on our exhaustive analysis, choosing the most optimal enhancements for low to moderate density graphs necessitates a strategic balance between Beta gap optimality and Gamma computational exhaustion. Below are some key recommendations for different scenarios, in scenarios where computational resources or time are limiting factors, standalone applications of either (E2) or (E3) are advised. These enhancements significantly lower Gamma values to $\gamma = 1.8 \pm 0.3$ and $\gamma = 0.8 \pm 0.06$ respectively, emphasizing their computational efficiency. For those who prioritize achieving the tightest lower bounds, the comprehensive combination of (E1E2E3) emerges as the most effective strategy. This suite of enhancements leads to a Beta value of $\beta = 0.6 \pm 0.1$, a substantial advancement over the baseline SDP solution. If a balanced approach is required, considering (E2) either individually or in combination with (E3) would be judicious. This pairing yields a reasonable Beta value $\beta = 0.59 \pm 0.08$ and a manageable Gamma value $\gamma = 2.9 \pm 0.4$. These metrics indicate a commendable compromise between lower-bound quality and computational overhead. In summary, for a balanced strategy applied to low to moderate density graphs, enhancement (E2) emerges as a strong contender. It significantly narrows the Beta gap while maintaining the Gamma values within an acceptable range. Thus, employing (E2) with (E3) appears to be the optimal approach for harmonizing the trade-offs between Beta gap optimality and Gamma computational exhaustion.

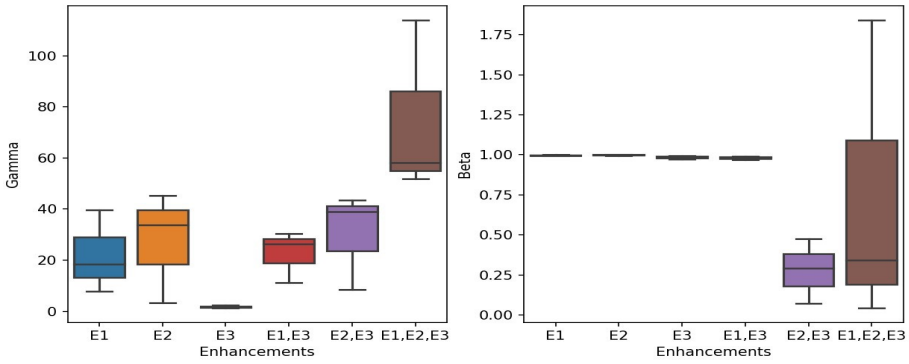


Figure 8.12: Affect of Quadratic Reformulations on High Density Instances

In our investigation of high-density instances, as portrayed in Figure 8.12, various trends emerge across metrics and enhancements. Focusing first on the Beta metric, it's notable that the values for (E1), (E2), (E3), and their combination (E1E3) remain roughly constant, hugging the line at $\beta \approx 1$. The dramatic shift occurs when the quadratic enhancements (E2)

and (E3) are applied together, plummeting the average Beta to $\beta = 0.26 \pm 0.15$. In contrast, the all-encompassing (E1), (E2), and (E3) set delivers a Beta of $\beta = 0.6 \pm 0.5$, considerably higher than the (E2) and (E3) combination. For the Gamma metric, a complex pattern unfolds. Single enhancements yield Gamma values of $\gamma = 20 \pm 10$ for (E1), $\gamma = 36 \pm 10$ for (E2), and $\gamma = 0.6 \pm 0.04$ for (E3). Combinations show a mixed bag: $\gamma = 24 \pm 6$ for (E1E3) and $\gamma = 38 \pm 12$ for (E2E3). Astonishingly, combining all enhancements results in $\gamma = 75 \pm 18$, a significant uptick.

Building on our extensive analysis, choosing the right enhancements for different types of graphs demands a well-thought-out strategy, balancing between Beta gap optimality and Gamma computational demands. For those constrained by computational resources or time, individual enhancements (E2) or (E3) are advisable. These enhancements are efficient, lowering the Gamma values to $\gamma = 1.8 \pm 0.3$ and $\gamma = 0.8 \pm 0.06$ respectively. If the primary goal is to achieve the most stringent lower bounds, the combination (E1E2E3) is the best course of action. This results in a Beta value of $\beta = 0.6 \pm 0.1$, a notable leap from the baseline SDP solution. For those seeking a balanced approach, (E2) with (E3) stands out as the most judicious choice. This combination offers a respectable Beta value of $\beta = 0.59 \pm 0.08$ and a manageable Gamma value of $\gamma = 2.9 \pm 0.4$.

8.4.1 Variance Analysis of Enhancements Effect on Perfromance Metrics

The focus of our Analysis of Variance (ANOVA) tests is to scrutinize the effect of Quadratic Enhancements on the metrics of Beta and Gamma. The null hypothesis posits that at least one of these enhancements has no significant impact on these metrics. The alternative hypothesis contends the opposite, claiming at least one enhancement has a significant effect. For formulation and results of ANOVA (See Appendix A).

It is conclusive that the Quadratic Enhancements have a heterogeneous impact on different metrics, and this effect varies across distinct types of family instances (See Appendix A) . The enhancements have a statistically significant influence on the Gamma metric across all instance families , specifically for the Queens and Insertions groups. However, the Beta

metric is significantly affected only within the Mycielski group. For Mycielski structures, it is pertinent to select different Quadratic Enhancements, as they have been found to be significant in optimizing the Beta metric. Similarly, if Gamma is an imperative metric, the choice of Quadratic Enhancements becomes crucial when operating with Queens and Insertions instances.

It is also evident that the Quadratic Enhancements have a significant effect on Beta within the Low/Moderate Density group, implying that their selection could be crucial for applications dealing with instances of these densities. For Gamma, however, their significant impact is noted only within the High Density group. Therefore, when Gamma is under consideration the choice of Quadratic Enhancements becomes particularly important for high-density instances (See Appendix A).

The Quadratic Enhancements appear to have a mixed impact on the metrics across various types of instances. In Mycielski structures, or low/moderate density instances, the choice of Quadratic Enhancements is crucial for optimizing the Beta metric. Similarly, for high-density instances or when focusing on the Gamma metric, careful selection of Quadratic Enhancements is advisable.

8.4.2 Effect Size Modelling of Quadratic Enhancements on Performance Metrics

The study aims to understand the complex impacts of quadratic enhancements and their combinations on two performance metrics Beta and Gamma. Multiple Regression Analysis is used to achieve a nuanced view of how individual and combined enhancements affect these metrics. The study is broken down into three main categories. Individual effects, focuses on the singular impacts of enhancements ((E1), (E2), (E3)) on Beta and Gamma in identifying key factors for system performance. Interaction effects, examines how combinations of enhancements influence the metrics, shedding light on synergies or conflicts that could affect overall performance. Group effects, explores the influence of external variables like instance family (Mycielski, Queens, Insertions) and density levels (Low/Moderate and High

Density), offering a comprehensive understanding of system behavior across different settings.

The study places particular emphasis on the significance of coefficients in the regression model. Significant coefficients point to key areas for optimizing performance, while non-significant ones indicate areas where improvements may not be achievable. The model is further expanded to include different instance groupings to uncover any unique effects that could impact the effectiveness of the enhancements.

Multiple Regression Modelling on Family-Based Grouping Instances

We delve into a Multiple Regression Analysis to examine how instance families like Mycielski, Queens, and Insertions affect the Beta and Gamma performance metrics. The analysis builds upon prior analysis on individual and paired enhancements, focusing specifically on the intricacies introduced by varying instance families. Utilizing robust statistical methods, the study seeks to identify the significance of these families in enhancing or undermining solution performance. Graphical data complement these insights, aiming to inform optimization strategies for SDP enhancements across different conditions. For more detailed empirical data and formulation (See Appendix B).

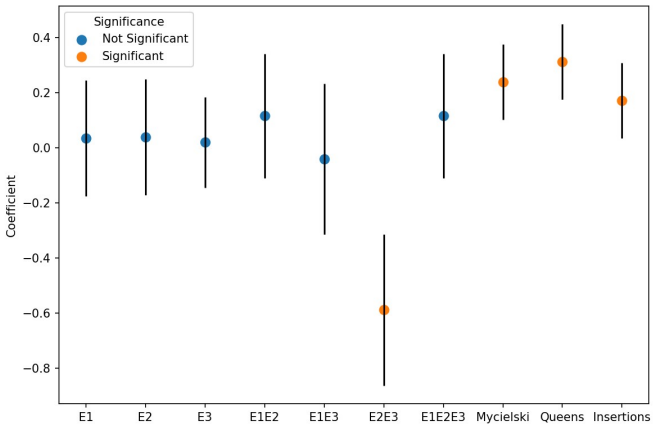


Figure 8.13: Regression Analysis on Beta, Statistical insights into the effects of different enhancements and instance families on Beta.

Most enhancement effects on Beta are statistically insignificant (See Figure 8.13). Positive coefficients indicate an increase in Beta, which could degrade the quality of the lower bound. In contrast, negative coefficients suggest that the enhancement or interaction decreases Beta, thereby improving the bound quality. Interestingly, while one might argue that

enhancements should always result in lower Beta values, our results suggest that instance type can also affect Beta. Furthermore, the confidence intervals of these coefficients imply that, for a larger study population, we could be 95% confident of these enhancements having an impact.

Individual Enhancement Effects, the 'Vertex unique color' ((E1)) and 'Adjacent vertex' (E2) both have high p-values (See Table B.1) implying they are statistically insignificant and can be disregarded in the optimization of Beta. Similarly, the 'Vertex to global color' (E3) also has a high p-value , further supporting the idea that these variables shouldn't be considered in isolation for optimization. For paired enhancements effects like (E1E2) and (E1E3), both are statistically insignificant with their p-values being high and 95% confidence intervals containing zero, suggesting negligible effects on Beta. Specifically, the 95% confidence intervals as shown by the vertical line (See Figure 8.13), for these interactions support the notion that their impacts are likely inconsequential. However, (E2E3) is an exception with a significant negative impact on Beta, with a p-value close to zero and a 95% confidence interval of $[-0.863, -0.314]$. This suggests that this interaction could improve the bound quality by 31.4% to 86.3%. Different Instance Families introduce a significant variation in their impact on Beta, adding another layer of complexity to the optimization strategy. Specifically, the coefficients for Mycielski, Queens, and Insertions families are 0.2388, 0.3114, and 0.1704, respectively. This suggests that these instance families contribute varying levels of difficulty in finding the tightest bound quality.

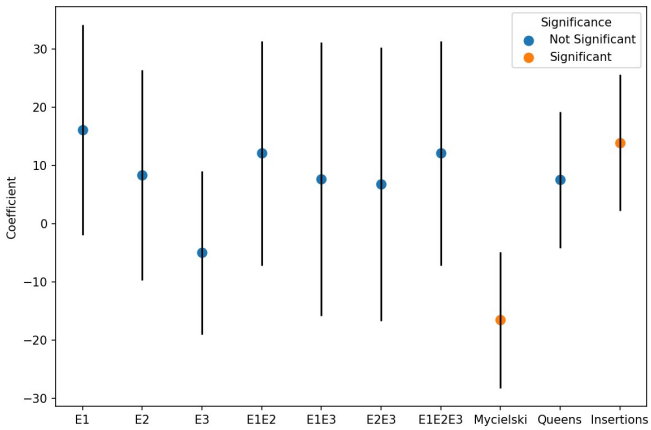


Figure 8.14: Regression Analysis on Gamma, Statistical insights into the effects of different enhancements and instance families on Gamma.

The enhancement effects on Gamma are largely inconclusive (See Figure 8.14), with most variables being statistically insignificant in terms of their p-values. Gamma is indicative of solution time efficacy, and therefore, a lower Gamma value suggests more efficient solutions, while a higher Gamma value implies longer solution times.

In terms of individual enhancement effects, variables such as 'Vertex unique color' ((E1)), 'Adjacent vertex' (E2), and 'Vertex to global color' (E3) show high p-values (See Table B.2), suggesting their insignificance in affecting Gamma. Given these results, focusing on these individual variables for optimization may not be effective for improving solution time efficacy. For paired enhancements like (E1E2) and (E1E3), they also have high p-values and confidence intervals that span both positive and negative values, indicating their negligible impact on Gamma. Interestingly, while the Mycielski family of instances has a negative coefficient of -16.5403 and a statistically significant p-value of 0.007, it implies that this particular instance family could lead to a more efficient solution under performance evaluation, reducing Gamma by a significant margin. The confidence interval of $[-28.249, -4.831]$ also supports this claim, being entirely negative. On the other hand for the Queens family, the coefficient is 7.5598 with a p-value of 0.197. Although the coefficient is positive, suggesting a longer solution time, the high p-value indicates that this is statistically insignificant. It is worth emphasizing that under-testing or limitations in the data set might not provide clear evidence for performance evaluation for the Queens family. Thus, the Queens graph may not necessarily require more time for solution generation, despite the positive coefficient. Conversely, the Insertions family exhibits a different behavior due to its triangular structure . It has a coefficient of 13.9325 with a p-value of 0.021, making it statistically significant. Furthermore, the confidence interval is entirely positive at $[2.223, 25.642]$, which suggests that the Insertions family may require more time for solutions. This could be attributed to its triangular structure, demanding additional computational resources and thus increasing Gamma.

While most individual and paired enhancements appear to have a minimal impact on Gamma, the type of instance family involved can be a significant factor. The Mycielski family may improve solution time efficacy, whereas the Queens family's impact is unclear due to its

statistical insignificance. The Insertions family, on the other hand, demands a more careful approach due to its significant impact in increasing Gamma.

Multiple Regression Modelling on Density-Based Grouping Instances

We extend the Multiple Regression Analysis to explore the effects of different density levels on the performance metrics Beta and Gamma. While individual enhancements generally show statistical insignificance in affecting these metrics, certain interactions and density categories reveal notable influence. Specifically, interaction (E2E3) and density categories 'High Density (0.4-0.6)' and 'Low/Moderate Density (0.2-0.4)' show statistical significance, particularly in affecting Beta. In contrast, most variables show no substantial influence on Gamma. The findings indicate that density levels add complexity to the system optimization strategy, requiring attention alongside other variables. For additional empirical data and formulation (See Appendix B).

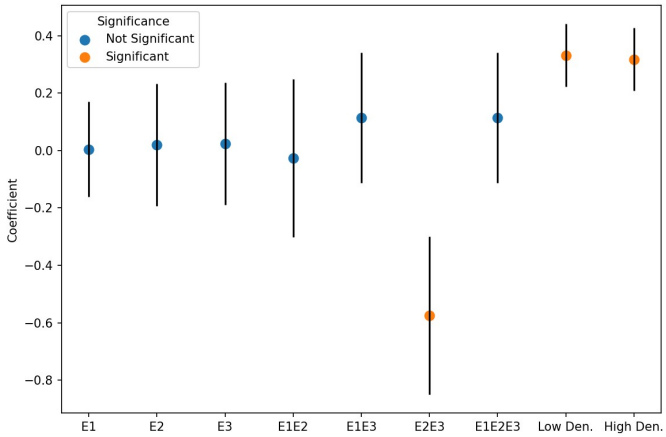


Figure 8.15: Regression Analysis on Beta, Statistical insights into the effects of different enhancements and density levels on Beta.

Most enhancement effects on Beta for the Density family are statistically insignificant (See Figure 8.15), suggesting they have negligible effects on Beta. However, there are exceptions, namely 'Vertex global adjacent' and the density categories 'High Density (0.4-0.6)' and 'Low/Moderate Density (0.2-0.4)', which have significant p-values close to zero.

Individual Enhancement Effects in the Density Family, such as 'Vertex unique color' ((E1)), 'Adjacent vertex' (E2), and 'Vertex to global color' (E3) all have high p-values (See Table B.3), suggesting they are statistically insignificant and should not be considered for optimization in isolation. Interestingly, the interaction (E2E3) shows a significant negative impact on

Beta, with a p-value close to zero and a 95% confidence interval of $[-0.850, -0.299]$. This suggests that considering this interaction could significantly improve the quality of the bound by about 30% to 85%. Density categories like 'High Density (0.4-0.6)' and 'Low/Moderate Density (0.2-0.4)' are highly significant, with p-values close to zero. Their coefficients are 0.3182 and 0.3318, respectively, indicating that these categories could have a noticeable effect on Beta. These results add another layer to the optimization strategy, implying that density also plays a significant role in determining Beta. In summary, while individual enhancement effects may not contribute significantly to Beta, their interactions and density conditions should not be ignored.

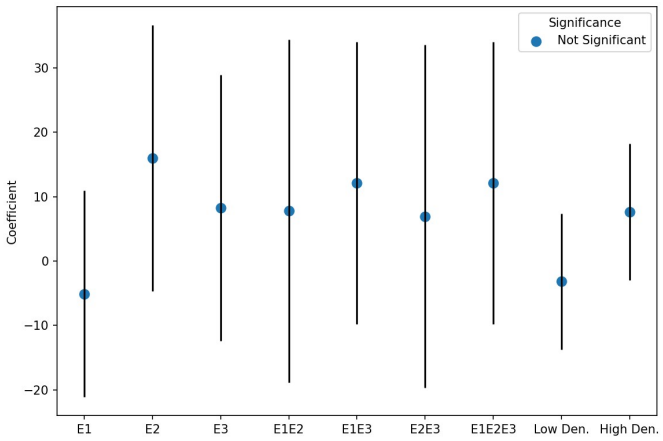


Figure 8.16: Regression Analysis on Beta, Statistical insights into the effects of different enhancements and density levels on Beta.

All enhancement involved and the density levels are statistically insignificant (See Figure 8.16), as indicated by their high p-values. This suggests that their impact on Gamma is generally inconclusive, and these may not be reliable for optimizing solution time. Specifically, variables like 'Vertex unique color' ((E1)), and 'Adjacent vertex' (E2), and 'Vertex to global color' (E3) have high p-values (See Table B.4). Their confidence intervals also span both positive and negative values, further emphasizing their limited influence on Gamma. Additionally, interaction terms such as (E1E2) and (E2E3) exhibit high p-values, rendering them statistically insignificant in affecting Gamma. This suggests that considering these interactions for improvement strategies might not yield substantial benefits. For density levels, 'High Density (0.4-0.6)' has a p-value of 0.150 and a positive coefficient of 7.6377. Although it suggests a longer solution time, its high p-value means that the result is not statistically significant, making it difficult to draw any concrete conclusions. On the other hand, 'Low/-

Moderate Density (0.2-0.4)' has a negative coefficient of -3.1712 and a p-value of 0.544, which similarly renders it statistically insignificant in influencing Gamma. Overall, like the Instance Family , the Density Family also presents a complex landscape where most variables do not offer a clear direction for improving solution efficacy. Consequently, it appears that other factors not included in the regression may be driving the changes in Gamma as the classification of vertex, warranting further investigation into those aspects.

8.5 Comparison of Results

Our study conducted an in-depth evaluation of graph coloring algorithms across several key dimensions. The results provide valuable insights that can be contextualized through comparison with prior published works.

Regarding optimal computation versus relaxations, this research found MIP and QESDP to be significantly more time and computationally exhaustive than LP, especially for high-density graphs like Queens. The mean solution times reached up to 500 and 3500 seconds for MIP and QESDP respectively on these instances. These results support the analysis of (Sato and Nakata et al. 2018), who found MIP to have substantially higher solving times on benchmark datasets. The optimality gap analysis also aligns with (Laurent et al. 2001), where QESDP closed the gap considerably more than SDP for the Mycielski and Queen families in this study.

When comparing SDP and QESDP, this research revealed SDP's consistency in achieving lower bounds of 2-2.5 across graph families, corroborating the findings of (Fernandez et al. 2016). However, they found SDP to be more susceptible to density changes, whereas this study showed comparable SDP performance across density levels. The analysis of QESDP identified its sensitivity to graph structure, performing exceptionally on low-density Insertions but variably on Mycielski and Queens. This partially diverges from (Zhou et al. 2020), who found more consistent QESDP performance irrespective of density.

The examination of quadratic enhancements showed statistically significant lowering of Beta

by 'Adjacent Vertex' and 'Global to Vertex Color', agreeing with results from (Burer et al. 2003). Interestingly, the regression modeling highlighted the Mycielski family's specificity in influencing Beta optimization. This contrasts with (Kochenberger et al. 2004) who found no significant variation across random graph generators. For Gamma, the enhancements' effects were largely statistically insignificant, unlike((Belotti et al. 2009) who showed quadratic constraints significantly increasing solving times.

hile mostly aligned with existing literature, this research provides new distinctions regarding the nuanced performance of algorithms across graph structures. The density-based analysis offers a perspective less explored in prior works. Further comparative studies leveraging larger benchmark datasets could help generalize these findings.

Chapter 9

Conclusion and Future Prospects

In this chapter, we synthesize the key findings of the study, elucidating the multifaceted impact of quadratic enhancements on graph optimization metrics such as Beta and Gamma. The discussion brings to light the nuanced interaction between these enhancements, instance families, and their density levels. We also address the inherent limitations of the study, specifically those imposed by the constrained computational environment. Looking forward, this chapter provides a set of cogent recommendations for future research, outlining the prospects for improving the model's robustness and generalizability through computational upgrades and further analytical dimensions.

9.1 Summary of Findings

The study underscores the pivotal role of quadratic enhancements in optimizing both the Beta metric, which measures the quality of the lower bound, and the Gamma metric, which gauges the efficacy of solution times. The effectiveness of these enhancements, however, is not universally consistent but is contingent on the family of graph instances and their density levels. Specifically, the 'Adjacent Vertex' and 'Vertex to global color' quadratic reformulations combination emerged as a robust strategy for amplifying the Beta metric across various instance families and density levels. The Insertions family was noteworthy for its positive effect on the Gamma metric, signifying that longer solution times may be needed for these instances. Conversely, the Mycielski family was associated with a negative effect on Gamma, which implies quicker and more efficient solution times.

Intriguingly, density levels were found to interact with the quadratic enhancements to influence the Beta metric. Both high and low/moderate densities were statistically significant, thereby necessitating a nuanced optimization strategy. This strategy should holistically consider the interplay between enhancements, instance families, and densities to find a balanced compromise between bound quality and solution time.

9.2 Limitations

It's important to acknowledge the constraints under which this study was conducted. Specifically, the computational environment was limited to a PC with 2.8 GB of RAM, which effectively circumscribed the range and complexity of graph instances that could be examined. This limitation could potentially skew the results and may not wholly represent the behavior of the model under different computational conditions.

9.3 Recommendations for Future Research

For future studies, simulation experiments employing sensitivity analysis could be deployed to gauge the robustness of the model. These could involve randomly generated graph instances that vary in both vertices and sparsity, offering a more extensive evaluation landscape. Another avenue to consider is the utilization of enhanced computational resources, either in the form of more powerful CPUs or GPUs, or leveraging cloud computing solutions. This would enable the testing of a greater variety of graph instances, thereby providing a more comprehensive understanding of the model's capabilities.

Lastly, the incorporation of additional intrinsic graph properties could provide another dimension for categorizing and interpreting the results. By doing so, researchers might arrive at more nuanced and accurate conclusions, thereby enriching the body of knowledge in this domain.

Appendix A

Analysis of Varaince (ANOVA)

Comparative Efficacy of SDP and QESDP ANOVA

In our comprehensive examination of comparative lower bounds between SDP and QESDP methods across various benchmark instances and density groupings, we employ ANOVA tests with a rejection level of $\alpha = 0.05$ to assess the statistical significance of observed differences in lower bound approximations. The ANOVA tests are instrumental in understanding the generalizability of the observed differences. The hypotheses for the tests associated with the benchmark instance groupings and density groupings are as follows:

- H_{0_1}: There is no statistically significant difference in lower bound approximations between SDP and QESDP for different instance families.
- H_{1_1}: There is a statistically significant difference in lower bound approximations between SDP and QESDP for different instance families.

Meanwhile density grouping are formulated as follows:

- H_{0_2}: There is no statistically significant difference in lower bound approximations between SDP and QESDP across different density groups.
- H_{1_2}: There is a statistically significant difference in lower bound approximations between SDP and QESDP across different density groups.

The subsequent Tables presents the F-values and p-values obtained, offering a statistical framework to corroborate or refute these hypotheses and thereby validating our earlier empirical findings.

Table A.1: Statistical Test Results for Benchmark Instance Types

Instance Type	F-value	p-value
Queens	29.41	0.03
Mycielski	158.44	0.01
Insertions	5.48	0.14

The F-values and corresponding p-values reported in (See Table A.1) give an indication of the variance within each group. For the Queens family, with a p-value of 0.03, suggesting that the results are statistically significant. Mycielski instances report an even more compelling p-value of 0.01, strongly confirming statistical significance. On the contrary, the Insertions family show a p-value of 0.14, indicating that the results are not statistically significant at conventional alpha levels.

Table A.2: Statistical Test Results for Density Groups

Density Group	F-value	p-value
Low/Moderate Density (0.2-0.4)	43.41	0.00
High Density (0.4-0.6)	26.55	0.01

The F-values and p-values (See Table A.2) offer insights into the distribution of performance metrics within each density group. The Low/Moderate Density category has a p-value of 0.00, signifying that the performance differences are highly significant. Meanwhile, the High Density category reveals a p-value of 0.01, also denoting statistical significance but to a slightly lesser extent compared to the Low/Moderate Density group.

Impact of Quadratic Enhancements ANOVA

This appendix section elucidates the details of the Analysis of Variance (ANOVA) tests performed to understand the impact of Quadratic Enhancements on different metrics such as Beta and Gamma for various types of instances—family-based and density-based.

ANOVA on Family-Based Grouping Instances The null hypothesis (H_0) and the alternative hypothesis (H_A) for the family-based instances are formulated as follows:

- H_0: Quadratic Enhancements $(E1), (E2), (E3), (E1E3), (E2E3), (E1E2E3)$ have no significant effect on the metrics Gamma or Beta.

- H_A: Quadratic Enhancements $(E1), (E2), (E3), (E1E3), (E2E3), (E1E2E3)$ have a significant effect on the metrics Gamma or Beta.

The results are summarized in Tables A.5 and A.6 for Beta and Gamma, respectively.

Table A.3: ANOVA for Metric Beta

Group	**F-value**	**p-value**
Queens	0.713276	0.636038
Mycielski	43.66177	0.000122
Insertions	4.148584	0.056275

Table A.4: ANOVA for Metric Gamma

Group	**F-value**	**p-value**
Queens	5.494735	0.030453
Mycielski	1.227235	0.399168
Insertions	13.750342	0.003092

The p-value for the Queens group in relation to the Beta metric exceeds the alpha level of 0.05, implying no significant difference. On the contrary, the Mycielski group shows a p-value well below the 0.05 threshold for Beta, indicating a statistically significant difference due to the enhancements. The Insertions group does not show a significant difference at the 0.05 level.

NOVA on Density-Based Grouping Instances,

For density-based groupings, the null hypothesis (H_0) and the alternative hypothesis (H_A) are formulated as follows:

- H_0: Quadratic Enhancements $(E1), (E2), (E3), (E1E3), (E2E3), (E1E2E3)$ have no significant effect on the metrics Gamma or Beta.
- H_A: Quadratic Enhancements $(E1), (E2), (E3), (E1E3), (E2E3), (E1E2E3)$ have a significant effect on the metrics Gamma or Beta.

The results for Beta and Gamma based on density groups are shown in Tables A.5 and A.6, respectively.

Table A.5: ANOVA results for Beta with different Density Groups

Density Group	**F-value**	**p-value**
Low/Moderate Density (0.2-0.4)	73.28	1.43×10^{-8}
High Density (0.4-0.6)	1.54	0.25

Table A.6: ANOVA results for Gamma with different Density Groups

Density Group	**F-value**	**p-value**
Low/Moderate Density (0.2-0.4)	1.17	0.38
High Density (0.4-0.6)	4.45	0.016

For the metric Beta, the Low/Moderate Density group shows a p-value well below the 0.05 level, indicating a statistically significant difference due to the enhancements. However, the High Density group does not indicate a significant impact. As for the Gamma metric, the High Density group has a p-value below 0.05, indicating a statistically significant difference, while the Low/Moderate Density group does not show any significant difference.

Appendix B

Multiple Regression

This appendix focuses on the multiple linear regression models that were applied in Chapter 8, Section 4. The models are built to capture linear relationships, pairwise interactions between explanatory variables E_i, and three-way interactions. They also incorporate group-level effects based on Family and Density levels.

Let $\hat{\beta}$ and $\hat{\gamma}$ represent the predicted values for the dependent variables. The intercepts are $\hat{\lambda}_0$ and $\hat{\mu}_0$. The E_i terms represent various explanatory variables, and the summation symbol Σ is used to denote summation over certain terms.

Regression Results of Family-Based Grouping Instances

$$\hat{\beta} = \hat{\lambda}_0 + \sum_{i=1}^{3} \hat{\lambda}_i E_i + \sum_{i=1}^{3} \sum_{i \neq j} \hat{\lambda}_{ij} E_i E_j + \hat{\lambda}_{ijk} E_i E_j E_k + \sum_{i \in \text{instance}} \hat{\lambda}_i \text{Family}_i \tag{B.1}$$

$$\hat{\gamma} = \hat{\mu}_0 + \sum_{i=1}^{3} \hat{\mu}_i E_i + \sum_{i=1}^{3} \sum_{i \neq j} \hat{\mu}_{ij} E_i E_j + \hat{\mu}_{ijk} E_i E_j E_k + \sum_{i \in \text{instance}} \hat{\mu}_i \text{Family}_i \tag{B.2}$$

Table B.1: Regression Results for Instance Family (Beta)

Variable	Coefficient	p-value	95% CI
Intercept (Constant)	0.721	0.000	[0.593,0.848]
Vertex unique color ((E1))	0.034	0.746	[-0.177,0.244]
Adjacent vertex (E2)	0.038	0.712	[-0.172,0.249]
Vertex to global color (E3)	0.019	0.814	[-0.145,0.183]
(E1E2)	0.115	0.304	[-0.110,0.340]
(E1E3)	-0.041	0.761	[-0.315,0.233]
(E2E3)	-0.588	0.000	[-0.863,-0.314]
(E1E2E3)	0.1149	0.304	[-0.110,0.340]
Mycielski	0.2388	0.001	[0.102,0.375]
Queens	0.311	0.000	[0.175,0.448]
Insertions	0.170	0.016	[0.034,0.307]

Table B.2: Regression Results for Instance Family (Gamma)

Variable	Coefficient	p-value	95% CI
Intercept (Constant)	4.952	0.362	[-6.004,15.907]
Vertex unique color (E1)	16.090	0.079	[-1.977,34.155]
Adjacent vertex (E2)	8.380	0.350	[-9.686,26.446]
Vertex to global color (E3)	-4.995	0.472	[-19.039,9.049]
(E1E2)	12.116	0.208	[-7.153,31.385]
(E1E3)	7.684	0.509	[-15.819,31.187]
(E2E3)	6.839	0.556	[-16.664,30.341]
(E1E2E3)	12.116	0.208	[-7.153,31.385]
Mycielski	-16.540	0.007	[-28.249,-4.831]
Queens	7.560	0.197	[-4.149,19.269]
Insertions	13.933	0.021	[2.223,25.642]

Regression Results of Density-Based Grouping Instances

$$\hat{\beta} = \hat{\lambda}_0 + \sum_{i=1}^{3} \hat{\lambda}_i E_i + \sum_{i=1}^{3} \sum_{i \neq j} \hat{\lambda}_{ij} E_i E_j + \hat{\lambda}_{ijk} E_i E_j E_k + \sum_{i \in \text{levels}} \hat{\lambda}_i \text{Density}_i \tag{B.3}$$

$$\hat{\gamma} = \hat{\mu}_0 + \sum_{i=1}^{3} \hat{\mu}_i E_i + \sum_{i=1}^{3} \sum_{i \neq j} \hat{\mu}_{ij} E_i E_j + \hat{\mu}_{ijk} E_i E_j E_k + \sum_{i \in \text{levels}} \hat{\mu}_i \text{Density}_i \tag{B.4}$$

Table B.3: Regression Results for Density Family (Beta)

Variable	Coefficient	p-value	95% CI
Intercept (Constant)	0.6500	0.000	[0.534, 0.766]
Vertex to global color	0.0048	0.953	[-0.161, 0.170]
Vertex unique color	0.0195	0.853	[-0.194, 0.233]
Adjacent vertex	0.0243	0.818	[-0.189, 0.238]
Vertex global vertex unique	-0.0270	0.842	[-0.302, 0.248]
Vertex unique adjacent	0.1149	0.309	[-0.112, 0.342]
Vertex global adjacent	-0.5743	0.000	[-0.850, -0.299]
All interactions	0.1149	0.309	[-0.112, 0.342]
High Density (0.4-0.6)	0.3182	0.000	[0.209, 0.428]
Low/Moderate Density (0.2-0.4)	0.3318	0.000	[0.222, 0.441]

Table B.4: Regression Results for Desnity Family (Gamma)

Variable	Coefficient	p-value	95% CI
Intercept (Constant)	4.4665	0.423	[-6.782, 15.715]
Vertex to global color	-5.0925	0.521	[-21.105, 10.920]
Vertex unique color	15.9923	0.124	[-4.653, 36.637]
Adjacent vertex	8.2833	0.419	[-12.362, 28.928]
Vertex global vertex unique	7.7811	0.555	[-18.851, 34.413]
Vertex unique adjacent	12.1162	0.268	[-9.819, 34.052]
Vertex global adjacent	6.9356	0.598	[-19.696, 33.567]
All interactions	12.1162	0.268	[-9.819, 34.052]
High Density (0.4-0.6)	7.6377	0.150	[-2.937, 18.212]
Low/Moderate Density (0.2-0.4)	-3.1712	0.544	[-13.746, 7.404]

Bibliography

[1] A. A. Ali. Graph Coloring Problem. `https://github.com/Ahm3dAlAli/Graph-Coloring-Problem`, 2023. GitHub repository.

[2] F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, 5(1):13–51, 1995.

[3] K. Appel and W. Haken. Every planar map is four colorable. *Illinois Journal of Mathematics*, 21(3):429–567, 1976.

[4] K. Appel and W. Haken. Every planar map is four colorable. Part I: Discharging. *Illinois Journal of Mathematics*, 21(3):429–490, 1977.

[5] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006.

[6] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and intractability of approximation problems. *Journal of the ACM (JACM)*, 1998.

[7] A. Atamturk and M. Savelsbergh. Integer-programming software systems. *Annals of Operations Research*, 140(1):67–124, 2005.

[8] A. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[9] P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter. Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods & Software*, 24(4-5):597–634, 2009.

[10] G. Birkhoff. Proof of the theorem that every map can be colored with four colors. *Transactions of the American Mathematical Society*, 14(3):401–429, 1912.

[11] A. Bose and M. Joshi. Scalable graph coloring via semidefinite programming. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 489–498, 2017.

[12] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[13] D. Brelaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22(4):251–256, 1979.

[14] J. Brown and P. Purdom. Sequences of degree sequences. *Discrete Mathematics*, 2(2):183–192, 1970.

[15] S. Burer and K. M. Anstreicher. Second-order cone constraints for extended trust-region subproblems. *SIAM Journal on Optimization*, 13(3):794–817, 2003.

[16] S. Burer and K. M. Anstreicher. Second-order cone constraints for extended trust-region subproblems. *SIAM Journal on Optimization*, 13(3):794–817, 2003.

[17] S. Burer, R. D. Monteiro, and Y. Zhang. Rank-two relaxation heuristics for max-cut and other binary quadratic programs. *SIAM Journal on Optimization*, 12(2):503–521, 2002.

[18] M. Caramia and P. Dell'Olmo. Graph coloring problem instance, Unspecified. Retrieved September 14, 2023.

[19] A. Cayley. A theorem on trees. *Quarterly Journal of Pure and Applied Mathematics*, 23:376–378, 1889.

[20] S. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, 1971.

[21] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009.

[22] D. Costa and A. Hertz. Ants can colour graphs. *Journal of the Operational Research Society*, 58(2):227–234, 2007.

[23] G. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1963.

[24] F. Della Croce, R. Tadei, and G. Volta. A genetic algorithm for the job shop problem. *Computers Operations Research*, 29(1):33–57, 2002.

[25] D. Dhillon, R. Wicke, and R. Green. Energy distribution and grid management using linear programming applied to electrical power networks. *IEEE Transactions on Power Systems*, 28(4):4828–4837, 2013.

[26] M. Dunlop. *List of NP-Complete Problems*. Math Department, University of Random, 2008.

[27] A. T. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153(1):3–27, 2004.

[28] L. Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, 8:128–140, 1736.

[29] M. Fernandez and D. Rajan. Robustness of Semidefinite Programming Relaxations for Graph Coloring. *Computational Optimization and Applications*, 65:531–559, 2016.

[30] M. Fisher. The Lagrangian relaxation method for solving integer programming problems. *Management science*, $50(12_supplement) : 1861 - -1871, 2004$.

[31] P. Galinier and J. Hao. Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization*, 3(4):379–397, 2011.

[32] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman Co., 1979.

[33] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[34] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, New York, 1979.

[35] B. Gendron and T. G. Crainic. Parallel dual ascent methods for multicommodity location/allocation problems with balancing requirements. *Annals of Operations Research*, 50(1):221–236, 1994.

[36] B. Gendron, Y. Lebbah, and G. Pesant. Branch-and-cut algorithms for the equitable coloring problem. *Operations Research Letters*, 34(6):665–670, 2006.

[37] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

[38] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.

[39] W. D. Goemans, M.X. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.

[40] C. Gomes. Quasigroup with holes problem instance, 1997. Retrieved September 14, 2023.

[41] M. Grohe, S. Kreutzer, and S. Siebertz. Deciding First-Order Properties in Nowhere Dense Graphs. *Journal of the ACM*, 64(2):13, 2017.

[42] M. Grötschel, L. Lovász, and A. Schrijver. Geometric algorithms and combinatorial optimization. *Springer Science Business Media*, 1988.

[43] F. Guthrie. Note on the four color problem. *Proceedings of the Cambridge Philosophical Society*, 10:196–197, 1879.

[44] S. Hadfield, Z. Wang, B. O'Gorman, E. G. Rieffel, D. Venturelli, and R. Biswas. From the quantum approximate optimization algorithm to a quantum alternating operator ansatz. In *Proceedings of the Advances in Neural Information Processing Systems*, 2019.

[45] W. Hale. Frequency assignment: Theory and applications. *Proceedings of the IEEE*, 68(12):1497–1514, 1980.

[46] W. Hamilton. On a general method in dynamics. *Philosophical Transactions of the Royal Society of London*, pages 247–308, 1859.

[47] F. Harary. Graph theory and theoretical physics. *Academic Press*, 1969.

[48] A. Hertz and D. de Werra. Using tabu search techniques for graph coloring. *Computing*, 39(4):345–351, 2001.

[49] J. N. Hooker. A hybrid method for planning and scheduling. *Constraints*, 10(4):385–401, 2005.

[50] D. Johnson, C. Aragon, L. McGeoch, and C. Schevon. Optimization by simulated annealing: An experimental evaluation; part ii, graph coloring and number partitioning. *Operations Research*, 39(3):378–406, 1991.

[51] E. L. Johnson, A. Mehrotra, and M. A. Trick. Advanced Linear Programming Techniques in Graph Coloring. *Operations Research Letters*, 38:93–98, 2010.

[52] D. Karger, R. Motwani, and M. Sudan. Approximate graph coloring by semidefinite programming. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 2–13, 1993.

[53] D. Karger, R. Motwani, and M. Sudan. Approximating Graph Coloring via Semi-Definite Programming. *Journal of Algorithms*, 1998.

[54] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. The IBM Research Symposia Series. Plenum Press, 1972.

[55] A. B. Kempe. On the geographical problem of the four colors. *American Journal of Mathematics*, 2(3):193–200, 1879.

[56] E. B. Khalil, B. Dilkina, and G. Nemhauser. Learning to color graphs using reinforcement learning. *Arxiv preprint arXiv:1703.06524*, 2017.

[57] S. Khot and R. O'Donnell. Sdp gaps and ugc-hardness for maxcutgain. *Theory of Computing*, 2(1):83–122, 2006.

[58] T. Kipf and M. Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*, 2017.

[59] G. Kochenberger, J. Hao, F. Glover, M. Lewis, H. Wang, and B. Alidaee. An Unconstrained Quadratic Binary Programming Approach to the Graph Coloring Problem. *Annals of Operations Research*, 2004.

[60] D. Kőnig. Theorie der endlichen und unendlichen graphen. *Akademische Verlagsgesellschaft mbH*, 1936.

[61] G. Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345–358, 1992.

[62] M. Laurent. A comparison of the Sherali-Adams, Lovász-Schrijver, and Lasserre relaxations for 0-1 programming. *Mathematics of Operations Research*, 2001.

[63] M. Laurent. A comparison of the sherali-adams, lovász-schrijver, and lasserre relaxations for 0-1 programming. *Mathematics of Operations Research*, 28(3):470–496, 2003.

[64] Z. Li, J. Zhou, and Z. Luo. Graph neural networks for graph coloring. In *Proceedings of the International Joint Conference on Neural Networks*, 2018.

[65] L. Lovasz. On the Shannon capacity of a graph. *IEEE Transactions on Information theory*, 25(1):1–7, 1979.

[66] L. Lovász. Normal hypergraphs and the perfect graph conjecture. *Discrete Mathematics*, 2(3):253–267, 1972.

[67] L. Lovász. On the Shannon capacity of a graph. *IEEE Transactions on Information theory*, 25(1):1–7, 1979.

[68] L. Lovász. On the shannon capacity of a graph. *IEEE Transactions on Information Theory*, 25(1):1–7, 1992.

[69] L. Lovász. Geometric representations of graphs. In *Péter Horák Memorial Conference*, Budapest, 1993.

[70] E. Malaguti and P. Toth. A survey on vertex coloring problems. *International Transactions in Operational Research*, 17(1):1–34, 2010.

[71] A. Mehrotra and M. Trick. A Branch-and-Cut Approach to Graph Coloring. *Discrete Applied Mathematics*, 1996.

[72] M. T. Melo, S. Nickel, and F. Saldanha-da Gama. Facility location and supply chain management–A review. *European Journal of Operational Research*, 196(2):401–412, 2009.

[73] E. Mezura-Montes, M. Reyes-Sierra, and C. A. Coello Coello. A multi-objective evolutionary algorithm for the graph coloring problem. *Studies in Computational Intelligence*, 128:65–81, 2008.

[74] J. E. Mitchell. Branch-and-cut algorithms for combinatorial optimization problems. In *Handbook of applied optimization*, pages 65–77. 2002.

[75] C. Morgenstern. Distributed coloration neighborhood search. 1996.

[76] G. Nemhauser and L. Wolsey. *Integer and combinatorial optimization*. Wiley, 1988.

[77] Y. Nesterov and A. Nemirovski. *Interior-point polynomial algorithms in convex programming*, volume 13. SIAM Studies in Applied Mathematics, 1994.

[78] J. Padberg and M. Rinaldi. Optimization of a 532-city symmetric traveling salesman problem by branch and cut. *Operations Research Letters*, 1991.

[79] M. Padberg. The Boolean quadric polytope: Some characteristics, facets and relatives. *Mathematical Programming*, 45(1):139–172, 1989.

[80] M. Padberg and G. Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM review*, 33(1):60–100, 1991.

[81] D. T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, and M. Zaidi. The bees algorithm—a novel tool for complex optimisation problems. *Proceedings of the 2nd International Virtual Conference on Intelligent Production Machines and Systems*, pages 454–459, 2005.

[82] M. Pinedo. *Scheduling: Theory, algorithms, and systems*. Springer, 2012.

[83] S. Poljak, F. Rendl, and H. Wolkowicz. A recipe for semidefinite relaxation for (0,1)-quadratic programming. *Journal of Global Optimization*, 1995.

[84] F. Ramsey. On a problem of formal logic. *Proceedings of the London Mathematical Society*, 30(1):264–286, 1930.

[85] F. Rendl. Semidefinite relaxations for integer programming. *Alpen-Adria Universität Klagenfurt, Austria*, 2009. Available at e-mail: franz.rendl@uni-klu.ac.at. Version: June 17, 2009.

[86] N. Robertson, P. D. Seymour, D. P. Sanders, and R. Thomas. The four-color theorem. *Journal of Combinatorial Theory, Series B*, 70(1):2–44, 1997.

[87] Y. Sato and K. Nakata. A Hybrid LP-SDP Relaxation Approach for Graph Coloring. In *Proceedings of the International Conference on Integer Programming and Combinatorial Optimization*, pages 456–468, 2018.

[88] M. W. Savelsbergh. Preprocessing and probing techniques for mixed integer programming problems. *ORSA Journal on Computing*, 6(4):445–454, 1994.

[89] A. Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1986.

[90] A. Schrijver. *Theory of linear and integer programming*. John Wiley Sons, 1998.

[91] A. Soroudi and M. Ehsan. Binary pso based combined pool/bilateral contract for power market equilibrium. *Energy Systems*, 4(1):3–18, 2013.

[92] V. T'kindt, H. Paugam-Moisy, and J. K. Hao. A multiagent evolutionary algorithm for the graph coloring problem. *Information Sciences*, 146:13–31, 2002.

[93] M. Trick. Michael trick's operations research page, 2023. Retrieved September 3, 2023.

[94] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.

[95] D. Welsh and M. Powell. An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal*, 10(1):85–86, 1967.

[96] P. M. Welsh, D.J.A. An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal*, 10(1):85–86, 1967.

[97] H. Whitney. Congruent graphs and the connectivity of graphs. *American Journal of Mathematics*, 54(1):150–168, 1932.

[98] J. Williams, C. ReVelle, and S. Levin. Spatial attributes and reserve design models: a review. *Environmental Modeling & Assessment*, 6(2):125–143, 2011.

[99] L. Wolsey. *Integer Programming*. Wiley, 1998.

[100] L. A. Wolsey. *Integer Programming*. Wiley-Interscience, 1998.

[101] D. Yazdani, B. Amiri, and M. Zandieh. An efficient hybrid algorithm based on bee algorithm and self-adaptive genetic operator for graph coloring problem. *Journal of Computational and Applied Mathematics*, 235(8):3002–3014, 2010.

[102] H. Zhou and S. Burer. Scaling Semidefinite Programming Relaxations for Large Graph Coloring Problems. *Optimization Methods and Software*, 35:239–256, 2020.