UNIVERSIDADE FEDERAL FLUMINENSE

MARIA LUÍZA LÓPEZ DA CRUZ

Near-Bipartiteness on graphs having small dominating sets

NITERÓI

2023

MARIA LUÍZA LÓPEZ DA CRUZ

Near-Bipartiteness on graphs having small dominating sets

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Ciência da Computação

Orientador:
Uéverton dos Santos Souza

Coorientador:
Raquel de Souza Francisco Bravo

NITERÓI

2023

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

C955n Cruz, Maria Luíza López da
Near-Bipartiteness on graphs having small dominating sets / Maria Luíza López da Cruz. - 2023.
56 f.: il.

Orientador: Uéverton dos Santos Souza.
Coorientador: Raquel de Souza Francisco Bravo.
Dissertação (mestrado)-Universidade Federal Fluminense, Instituto de Computação, Niterói, 2023.

1. Teoria dos grafos. 2. Algoritmo em grafos. 3. Complexidade computacional. 4. Otimização combinatória (Computação). 5. Produção intelectual. I. Souza, Uéverton dos Santos, orientador. II. Bravo, Raquel de Souza Francisco, coorientadora. III. Universidade Federal Fluminense. Instituto de Computação.IV. Título.

CDD - XXX

Bibliotecário responsável: Debora do Nascimento - CRB7/6368

MARIA LUÍZA LÓPEZ DA CRUZ

Near-Bipartiteness on graphs having small dominating sets

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Ciência da Computação

Aprovada em agosto de 2023.

BANCA EXAMINADORA

Prof. Uéverton dos Santos Souza - Orientador, UFF

Profª. Raquel de Souza Francisco Bravo - Coorientador, UFF

Prof. Fábio Protti, UFF

Prof. Rodolfo Alves de Oliveira, UFF

Profª. Erika Morais Martins Coelho, UFG

Niterói

2023

"Dê-me, Senhor, agudeza para entender, capacidade para reter, método e faculdade para aprender, sutileza para interpretar, graça e abundância para falar. Dê-me, Senhor, acerto ao começar, direção ao progredir e perfeição ao concluir"

(Santo Tomás de Aquino).

Agradecimentos

Primeiramente, agradeço a Deus pelo dom da vida, por todas as coisas boas que Ele me deu e me dá e pela oportunidade de trilhar essa jornada acadêmica. Agradeço também pela oportunidade de conhecer tantas pessoas maravilhosas, colegas e amigos com os quais tive o prazer de conviver durante estes anos na Universidade Federal Fluminense e que farão sempre parte da minha vida.

Agradeço também à minha família, por todo amor, carinho, atenção e apoio que me dão. Sem eles, eu não seria nada. Seu carinho e incentivo foram a força motriz que me impulsionou nos momentos de desafio e incerteza. Eles são minha base, meu porto seguro, meu tesouro mais precioso.

Desejo registrar também a minha profunda gratidão aos professores que, com dedicação e sabedoria, compartilharam conhecimentos e experiências durante minha jornada acadêmica. Em especial, não posso deixar de agradecer ao meu orientador, Professor Uéverton Souza, e à minha coorientadora, Professora Raquel Bravo, pela orientação precisa, pelos conselhos valiosos e pela paciência em me guiar na elaboração deste trabalho. Vocês foram verdadeiros mestres, e sou grata por terem acreditado em meu potencial e contribuído de forma significativa para o meu crescimento acadêmico.

Estendo também meus agradecimentos à CAPES e à FAPERJ pela concessão das bolsas que tornaram possível a realização deste trabalho. Seu apoio foi essencial para minha jornada acadêmica e para a concretização deste estudo. Muito obrigada pelo investimento em minha formação e no avanço da pesquisa no país.

Agradeço também aos funcinários desta instituição, cujo trabalho contribuiu direta ou indiretamente para o desenvolvimento deste estudo. Sua prontidão e apoio foram fundamentais em cada etapa deste percurso.

Por fim, agradeço a todos os colegas de curso, pessoas maravilhosas que conheci na UFF. Cada interação, debate e troca de ideias enriqueceram minha formação acadêmica e pessoal, deixando uma marca indelével em minha vida. Sou muito privilegiada por tê-los ao meu lado.

Resumo

No problema Near-Bipartiteness nos é fornecido um grafo simples $G = (V, E)$ e perguntado se $V(G)$ pode ser particionado em dois conjuntos $\mathcal{S}$ e $\mathcal{F}$ de tal forma que $\mathcal{S}$ seja um conjunto estável e $\mathcal{F}$ induza uma floresta. Alternativamente, Near-Bipartiteness pode ser visto como o problema de determinar se G admite um conjunto de vértices de retroalimentação independente (independent feedback vertex set) $\mathcal{S}$ ou uma cobertura por vértices acíclica (acyclic vertex cover) $\mathcal{F}$. Uma vez que tal problema é NP-completo até mesmo para grafos com diâmetro três, estudamos primeiramente a propriedade de ser quase-bipartido em grafos que possuem uma aresta dominante, uma subclasse natural de grafos com diâmetro três. No que diz respeito a grafos que possuem uma aresta dominante, apresentamos um algoritmo de tempo polinomial para Near-Bipartiteness e provamos que Connected Near-Bipartiteness, a variante em que a floresta deve ser conexa, é NP-completo. Além disso, mostramos que Independent Feedback Vertex Set, o problema de encontrar uma quase-bipartição $(\mathcal{S}, \mathcal{F})$ minimizando $|\mathcal{S}|$, e Acyclic Vertex Cover, o problema de encontrar uma quase bipartição $(\mathcal{S}, \mathcal{F})$ minimizando $|\mathcal{F}|$, são ambos NP-difíceis quando restritos a essa classe de grafos. Estendendo nossa abordagem de tempo polinomial para lidar com Near-Bipartiteness em grafos que possuem conjuntos dominantes de tamanhos limitados, obtemos um algoritmo de tempo $O(n^2 \cdot m)$ para resolver Near-Bipartiteness em grafos livres de P_5, melhorando o atual estado da arte de tempo $O(n^{16})$.

Palavras-chave: Problema da quase-bipartição, conjunto de vértices de retroalimentação independente, cobertura por vértices acíclica, conjunto independente, aresta dominante.

Abstract

In the Near-Bipartiteness problem, we are given a simple graph $G = (V, E)$ and asked whether $V(G)$ can be partitioned into two sets $\mathcal{S}$ and $\mathcal{F}$ such that $\mathcal{S}$ is a stable set and $\mathcal{F}$ induces a forest. Alternatively, Near-Bipartiteness can be seen as the problem of determining whether G admits an independent feedback vertex set $\mathcal{S}$ or an acyclic vertex cover $\mathcal{F}$. Since such a problem is NP-complete even for graphs with diameter three, we first study the property of being near-bipartite on graphs having a dominating edge, a natural subclass of diameter-three graphs. Concerning graphs having a dominating edge, we present a polynomial-time algorithm for Near-Bipartiteness and prove that Connected Near-Bipartiteness, the variant where the forest must be connected, is NP-complete. In addition, we show that Independent Feedback Vertex Set, the problem of finding a near-bipartition $(\mathcal{S},\mathcal{F})$ minimizing $|\mathcal{S}|$, and Acyclic Vertex Cover, the problem of finding a near-bipartition $(\mathcal{S},\mathcal{F})$ minimizing $|\mathcal{F}|$, are both NP-hard when restricted to such a class of graphs. Extending our polynomial-time approach to deal with Near-Bipartiteness on graphs having bounded dominating sets, we obtain a $O(n^2 \cdot m)$-time algorithm to solved Near-Bipartiteness on P_5-free graphs, improving the current $O(n^{16})$-time state of the art.

Keywords: Near-bipartite; independent feedback vertex set; acyclic vertex cover; stable set; dominating edge.

List of Figures

Lista de Abreviaturas e Siglas

CNF conjunctive normal form

FPT Fixed-Parameter Tractable

IFVS Independent Feedback Vertex Set

TSP Traveling Salesman Problem

Contents

1 Introduction

With the constant advance of technology, graphs are increasingly becoming a part of everyday tasks for a significant portion of the population. They directly or indirectly assist in accessing linked websites on the internet or finding the shortest route, for example. Another well-known and extensively studied application involves partitioning the set of vertices in a graph into subsets. This partitioning technique is widely used in clustering and detecting problems in social networks, as well as in scheduling problems and image processing. Hence, the diversity of problems within the field of Graph Theory highlights the importance of this subject both in theory and practice.

1.1 Motivation

In 1972, Richard Karp presented the NP-completeness proof of 21 fundamental problems for Computer Science (KARP, 1972). Feedback Vertex Set, Independent Set and Vertex Cover are three of these classical problems. Feedback Vertex Set consists of finding a minimum set of vertices such that its removal eliminates all cycles of the input graph, Independent Set consists of determining a maximum set of pairwise nonadjacent vertices (also known as a stable set), and Vertex Cover is the problem of determining a minimum set of vertices intersecting all edges (called vertex cover) of the input graph. Note that if $\mathcal{S}$ is a stable set of $G = (V,E)$ then $\mathcal{F} = V(G) \setminus \mathcal{S}$ is a vertex cover of G.

An independent feedback vertex set (IFVS) of a graph G is a set of vertices that is independent/stable and also a feedback vertex set of G. Defined by Yang A. and Yuan J. in (YANG; YUAN, 2006), a graph $G = (V,E)$ has a *near-bipartition* $(\mathcal{S},\mathcal{F})$ if there exist $\mathcal{S} \subseteq V$ and $\mathcal{F} = V \setminus \mathcal{S}$ such that $\mathcal{S}$ is a stable set, and $\mathcal{F}$ induces a forest. Furthermore, $\mathcal{S}$ and $\mathcal{F}$ can be empty sets. A graph that admits a near-bipartition is a *near-bipartite graph*. Note that the class of near-bipartite graphs is exactly the class of graphs having independent feedback vertex sets. Also, a graph G has an independent feedback vertex

set $\mathcal{S}$ if and only if it has an acyclic vertex cover $\mathcal{F}$, i.e., a vertex cover $\mathcal{F}$ such that $G[\mathcal{F}]$ is acyclic (a vertex cover inducing a forest).

The problem of recognizing near-bipartite graphs, so-called Near-Bipartiteness, is NP-complete even when restricted to graphs with maximum degree four (YANG; YUAN, 2006), graphs with diameter three (BONAMY et al., 2018), line graphs (BONAMY et al., 2019), and planar graphs (BONAMY et al., 2017; DROSS; MONTASSIER; PINLOU, 2017). On the other hand, Brandstädt et al. (BRANDSTÄDT; BRITO, et al., 2013) proved that Near-Bipartiteness is polynomial-time solvable on cographs. Yang and Yuan (YANG; YUAN, 2006) showed that Near-Bipartiteness is polynomial-time solvable for graphs of diameter at most two and that every connected graph of maximum degree at most three is near-bipartite except for the complete graph on four vertices (K_4). Besides, Bonamy et al. (BONAMY et al., 2019) proved that Near-Bipartiteness on P_5-free graphs can be solved in $O(n^{16})$ time. FPT algorithms parameterized by k for finding an independent feedback vertex set of size at most k can be found in (AGRAWAL et al., 2017; LI; PILIPCZUK, 2020; MISRA; PHILIP, et al., 2012)

A *coloring* for a graph G is an assignment of colors (labels) to all vertices of G. A *proper coloring* for G is an assignment of color $c(u)$, for each vertex $u \in V$, such that $c(u) \neq c(v)$ if $uv \in E(G)$. A graph G is *k-colorable* if there exists a proper coloring for G with at most k colors. The *chromatic number* of G, $\chi(G)$, is the smallest number k for G being k-colorable. A clear necessary condition for a graph to be near-bipartite is the following.

Proposition 1. *If a graph G is near-bipartite then G is 3-colorable.*

By Proposition 1, it holds that K_4 is a natural forbidden subgraph for near-bipartite graphs. A graph G is called *perfect* if for every induced subgraph H of G holds that its chromatic number equals the size of its largest clique, $\chi(H) = \omega(H)$. In particular, $\omega(G) = \chi(G)$.

Given the relationship between Near-Bipartiteness and 3-Coloring it becomes interesting to question the behavior of Near-Bipartiteness on subclasses of perfect graphs. Note that a perfect graph is 3-colorable if and only if it is K_4-free.

However, the complexity of 3-Coloring and Near-Bipartiteness are not necessarily the same, depending on the graph class being explored. Grötschel, Lovász and Schrijver (GRÖTSCHEL; LOVÁSZ; SCHRIJVER, 1984) proved that Coloring is solved in polynomial time for perfect graphs, while Brandstädt et al. (BRANDSTÄDT; BRITO,

et al., 2013) proved that Near-Bipartiteness is NP-complete in the same graph class.

Near-Bipartiteness can also be seen as a variant of 2-Coloring. For an input graph G, the question is whether its vertex set can be colored with two colors (not necessarily properly coloring) such that one color class is K_2-free (a stable set), and the other is cycle-free (i.e., induces a forest). Other 2-Coloring variants have already received attention in the literature. In (ACHLIOPTAS, 1997), Achlioptas studied the problem of determining if there exists a bipartition of $V(G)$ where each part (color class) is H-free for some fixed graph H. He showed that for any graph H on more than two vertices, the problem is NP-complete. Another variant was considered by Schaefer (SCHAEFER, 1978), who asked whether a given graph G admits a 2-coloring of the vertices such that each vertex has exactly one neighbor with the same color as itself. Schaefer proved that such a problem is NP-complete even for planar cubic graphs. The problem studied by Schaefer (SCHAEFER, 1978) is a particular case of a defective coloring called (2,1)-coloring. A (k,d)-coloring of a graph G is a k-coloring of $V(G)$ such that each vertex has at most d neighbors with the same color. Some studies on (2,1)-coloring include (BORODIN; KOSTOCHKA; YANCEY, 2013; COWEN; GODDARD; JESURUM, 1997; LIMA, Carlos Vinicius et al., 2021). In addition, the problem of finding a bipartition where each part induces a subgraph of minimum degree at least k (for a given integer k) was studied in (BANG-JENSEN; BESSY, 2019). Also, the problem of partitioning the edge set of a graph into a stable set of edges (matching) and a forest has been studied in (LIMA, Carlos V.G.C. et al., 2017; PROTTI; SOUZA, 2018).

1.2 Goals

Motivated by the studies of 2-coloring variants and the natural relevance of feedback vertex sets that are independent/stable as well as vertex covers that are acyclic, we focus on the Near-Bipartiteness problem and its variants.

Thus, the main objective of this work is to analyze the Near-Bipartiteness problem in specific cases of graphs having a dominating edge, as well as in P_5-free graphs. Furthermore, we aim to extend the analysis to the problem of partitioning the set of vertices in graphs that have a dominating edge into two sets $\mathcal{S}$ and $\mathcal{T}$, where $\mathcal{S}$ is an independent set and $\mathcal{T}$ is a tree. Therefore, the objectives of this study are:

1. Analyze the Near-Bipartiteness problem in graphs with dominating edge and P_5-free graphs.

2. Extend the analysis to the problem of partitioning the set of vertices in graphs with dominating edge into two sets $\mathcal{S}$ and $\mathcal{T}$, where $\mathcal{S}$ is an independent set and $\mathcal{T}$ is a tree (Connected Near-Bipartiteness).

3. Study the problem of finding a near-bipartition ($\mathcal{S}$,$\mathcal{F}$) minimizing $|\mathcal{S}|$ (Independent Feedback Vertex Set (IFVS)) in graphs with dominating edge.

4. Investigate the problem of finding a near-bipartition ($\mathcal{S}$,$\mathcal{F}$) minimizing $|\mathcal{F}|$ (Acyclic Vertex Cover) in graphs with dominating edge.

By achieving these objectives, we aim to advance the knowledge and understanding of the Near-Bipartiteness problem and its variants, contributing to the fields of graph theory and computational complexity.

1.3 Main Contribution

The study aims to demonstrate that the Near-Bipartiteness problem can be solved in $O(n^4)$ time for a graph without induced P_5, improving upon the existing result in the literature of $O(n^{16})$ soluability (BONAMY et al., 2019). In this way, these findings contribute to a better understanding of the problem and provide more efficient solutions in practical scenarios.

Regarding the Near-Bipartiteness problem on P_5-free graphs, it is mentioned in (BACSÓ; TUZA, 1990) that every connected graph without P_5 as induced subgraphs contains either a dominating clique or a dominating P_3. Therefore, for a connected graph without as induced P_5, the goal is to determine if it admits the Near-Bipartiteness, and this can be achieved by following these steps proposed in this dissertation:

1. Check the existence of a set inducing K_4 in $O(n^4)$ time (which certifies no answer).

2. Obtain a dominating P_3 or K_3 (due to (BACSÓ; TUZA, 1990)).

3. Execute the algorithms described in Theorem 5 (for graphs having dominating K_3) or Theorem 6 (for graphs having dominating P_3), which will be presented in Chapter 5 of this work, solving the problem in $O(m.n^2) \cong O(n^4)$ time.

1.4 Dissertation Organization

The dissertation will be organized into seven chapters as follows:

- Chapter 1: Introduction. In this chapter, we provide a contextualization of the Near-Bipartiteness problem and its relevance. Additionally, we highlight the objectives and main contributions of this work to the literature.

- Chapter 2: Theoretical foundations. Here, we present fundamental concepts and notions of graphs that will be used throughout this dissertation. We also discuss satisfiability problems and graph problems that are employed in the mathematical proofs developed. Finally, we present an introduction to the concepts of algorithm complexity and problem classification.

- Chapter 3: The Near-Bipartiteness problem. In this chapter, we delve deeper into the Near-Bipartiteness problem, the main subject of this study. We define the concept of near-bipartite graphs and Near-Bipartiteness, and present some existing results in the literature related to this problem.

- Chapter 4: Graphs having a dominating edge an P_5-free graphs. This chapter presents each of the special graph classes chosen for the study of Near-Bipartiteness. In addition, we will also present some auxiliary lemmas that were used during the development of the mathematical proofs, which will contribute to their understanding.

- Chapter 5: Results obtained in graphs having a dominating edge. Here, we present the results obtained from studying Near-Bipartiteness in graphs with a dominating edge, as well as the partitioning of the vertex set of these graphs into an independent set and a tree (Connected Near-Bipartiteness). Additionally, the problem of finding a near-bipartition $(\mathcal{S},\mathcal{F})$ minimizing $|\mathcal{S}|$ (Independent Feedback Vertex Set (IFVS)) and the problem of finding a near-bipartition $(\mathcal{S},\mathcal{F})$ minimizing $|\mathcal{F}|$ (known as Acyclic Vertex Cover) have also been studied for this class of graphs.

- Chapter 6: Results obtained in P_5-Free graphs. This chapter presents the results obtained from studying Near-Bipartiteness in P_5-free graphs. Additionally, we provide a table of time complexities for each of the cases analyzed in this work.

- Chapter 7: Conclusion. In this final chapter, we recapitulate the main findings and contributions of the study, as well as discuss possible future research directions.

2 Basic Concepts

In this chapter, we will present definitions and basic concepts in the area of Graph Theory that will be used throughout this work.

2.1 Definitions

Definition 1 (Loop). *A loop is an edge whose vertex is related to itself, that is, given a graph G, an edge $e \in E(G)$ is said to be a loop when $e = (v,w)$ and $v = w$, $v, w \in V(G)$.*

Definition 2 (Multiple edges). *Double edges, also called multiple edges, are edges that have the same endpoints, that is, given a graph G, the edges e_1, $e_2 \in E(G)$ are called multiple edges if $e_1 = (v_1, v_2)$ and $e_2 = (v_3,v_4)$ where v_1, v_2, v_3 and $v_4 \in V(G)$ with $v_1 = v_3$ and $v_2 = v_4$.*

Definition 3 (Graph). *A graph is an ordered pair $G = (V,E)$, where V is a finite and nonempty set of vertices, denoted by $V(G)$ and E is a set of unordered pairs of distinct vertices, called edges, denoted by $E(G)$.*

Other point to be mentioned is that a simple graph does not contain loops (see definition 1), nor multiple edges (see definition 2). In this work, we will denote by graphs what we define as simple graphs.

Definition 4 (Directed graph). *A directed graph, also known as a digraph D, is an ordered pair (V, E), denoted as $D = (V, E)$, where V is a finite non-empty set of elements called vertices, and E is a set of ordered pairs of distinct vertices in V, called directed edges or arcs. For each distinct pair of vertices (v, w) in D, there is a unique direction in the edge (v, w) from v to w, and we say that (v, w) diverges from v and converges to w.*

The graphs referenced in this work are referred to as *undirected graphs*, as they have undirected edges. Therefore, in an *undirected graph* G, if (u, v) is an edge belonging to $E(G)$, then both u is adjacent to v and v is adjacent to u.

Definition 5 (Cardinality of the set of vertices). *The cardinality of the set of vertices, that is, the number of vertices of a graph will be denoted by n or $|V(G)|$, or simply, $|V|$.*

Likewise, the cardinality of the set of edges of a graph will be called by m or $|E(G)|$, or simply, $|E|$.

Definition 6 (Trivial graph)**.** *A graph G is said to be trivial if $|V(G)| = 1$, that is, if G has only one vertex.*

Definition 7 (Adjacent vertices)**.** *A vertex u is adjacent to another vertex v in G if $(u,v) \in E(G)$. In this case, we say that u and v are neighbours in G, and that the edge $e = (u,v)$ is incident to u and v, or which has endpoints u and v. The notation uv is also used to express that u is adjacent to v. Otherwise, we say that u and v are non-adjacent.*

Definition 8 (Neighborhood)**.** *Let u be a vertex of the graph G. We denote by $N(u)$ the set of vertices adjacent to u in G, i.e., $N(v) = \{u \in V(G) : vu \in E(G)\}$. Such a set is called the neighborhood of u. On the other hand, $N[u]$ denotes the set $N(u) \cup \{u\}$, which is named closed neighborhood of u.*

Alternatively, we can also define the neighborhood in a graph concerning a subset of vertices or an edge as follows:

Neighborhood of a vertex concerning a subset of vertices: The neighborhood of a vertex v concerning a subset of vertices S of the graph G, denoted as $N_S(v)$, is the set of vertices in S that are adjacent to v. Formally, $N_S(v) = \{u \in S \mid (v,u) \in E(G)\}$.

Neighborhood of an edge concerning a subset of vertices: The neighborhood of an edge e concerning a subset of vertices S, denoted as $N_S(e)$, is the set of vertices in S that are adjacent to at least one of the endpoints of e. Formally, $N_S(e) = \{v \in S \mid v$ is adjacent to $e\}$.

Definition 9 (Universal vertex)**.** *A vertex u is said to be universal when $N(u) = V(G) - \{u\}$, that is when the neighborhood of the vertex u is all the vertices of the graph except it.*

Definition 10 (Degree of a vertex)**.** *The degree of a vertex $v \in V(G)$, denoted by $d(v)$, is the number of edges incident to vertex v.*

Definition 11 (Subgraph)**.** *A graph H is a subgraph of a graph G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. That is, the set of vertices of H is equal to or contained in the set of vertices of the graph G. Likewise, the set of edges of H is equal to or contained in the set of edges of the graph G.*

On the other hand, the *induced subgraph* is a graph that preserves the structure of the original graph. In other words, let G be a graph and Y be a subset of the vertices of G, $Y \subseteq V(G)$, $Y \neq \emptyset$. The subgraph H of G induced by Y, denoted by $G[Y]$, is the subgraph H of G such that $V(H) = V(G[Y]) = Y$ and $E(H)$ is the set of all edges of G that have both endpoints in Y, $E(H) = E(G[Y]) = \{(x,y) \in E(G) \mid x \in Y \ e \ y \in Y\}$. For this work, when we refer to subgraphs we will implicitly assume that they are induced subgraphs.

Definition 12 (Complete graph). *A graph G is complete if, for any two distinct vertices of G, these are adjacent. We denote by K_n the complete graph with n vertices.*

Definition 13 (Isomorphism). *Given two graphs G_1 and G_2, we say that G_1 and G_2 are isomorphic, denoted as $G_1 \approx G_2$, when there exists a bijective function (an injective and surjective function):*

$$f : V(G_1) \to V(G_2)$$

Such that,

$$(v_1, v_2) \in E(G_1) \leftrightarrow (f(v_1), f(v_2)) \in E(G_2)$$

Thus, for $G_1 \approx G_2$ to hold, if (v_1, v_2) represents an edge in G_1, with v_1 and v_2 being their corresponding vertices, then $(f(v_1), f(v_2))$ is an edge in G_2, and their vertices $f(v_1)$, $f(v_2)$, respectively, are the images of v_1 and v_2. In this way, it can be said that the function $f : V(G_1) \to V(G_2)$ preserves their adjacencies.

Furthermore, some noteworthy observations are:

- *Since f is injective and surjective, we have $|V(G_1)| = |V(G_2)|$;*
- *As for each pair $(v, w) \in |E(G_1)|$, there is a corresponding pair $(f(v), f(w)) \in |E(G_2)|$. Therefore, $|E(G_1)| = |E(G_2)|$.*

Definition 14 (Dominating edge). *A dominating edge $e = (e_1, e_2)$ of a graph G is an edge that belongs to $E(G)$, and for each remaining vertex w of $V(G) \setminus \{e_1, e_2\}$ ($w \neq e_1$ and $w \neq e_2$) w is adjacent to e_1 or to e_2.*

Definition 15 (Walk). *A walk P in a graph G is an alternating finite sequence of vertices and edges of the form $(v_1, e_1, v_2, e_2, \ldots, v_n)$, where the vertices v_1 and v_n are called, respectively, origin and end of the walk P and the other vertices are called internal.*

It is worth noting that in a *walk* you can have repeated edges and vertices. However, when there is a case where all the edges of a walk are distinct, it is called *trail*. Another particular case is when all vertices belonging to a walk are distinct. In this case it is called *path* (see definition 16). In particular, a *path* is a *trail*.

Definition 16 (Path). *A path in a graph G is a walk $P = v_1 v_2 \ldots v_k$, where the v_i's are vertices (two by two distinct).*

Definition 17 (Chord). *A chord in P is an edge connecting two non-consecutive vertices of a path P.*

Definition 18 (Induced path). *An induced path is a path without chords. P_k denotes a path induced by k vertices. We say that a graph is free of P_k when it does not contain a P_k as an induced subgraph.*

Definition 19 (Cycle). *A path $v_1, \ldots, v_k, v_{k+1}$ is called a cycle when $v_1, \ldots, v_k$ is a path with $k \geq 3$ and $v_1 = v_{k+1}$.*

Definition 20 (Cyclic graph). *A graph G is said to be cyclic when G contains a cycle as an induced subgraph. Otherwise, we say that G is acyclic. We denote a cyclic graph with n vertices by C_n, $n \geq 3$. A cycle is called an even cycle if n is even, $C_4, C_6, \ldots C_n$, otherwise, C_n is odd if n is odd, $C_3, C_5, C_7, \ldots C_k$.*

Definition 21 (Maximal/Minimal). *A set S is maximal with respect to a given property P if S satisfies P, and every set S' that properly contains S does not satisfy P. On the other hand, the set S is minimal with respect to a given property P if S satisfies P and every set S' that is properly contained in S does not satisfy P.*

Definition 22 (Maximum/Minimum). *A set S is considered maximum with respect to a given property P when it is both maximal concerning P and has the highest cardinality of vertices. Conversely, S is regarded as a minimum set with respect to property P when it is both minimal concerning P and has the lowest cardinality of vertices.*

Definition 23 (Clique). *A set of vertices C of a graph G is a clique, if the subgraph C of G, G[C], is a complete graph. We denote by $\omega(G)$ the size of the maximum clique, that is:*

$$\omega(G) = max\ \{\ |V'|\ /\ V' \subseteq V\ and\ V'\ is\ a\ clique\ of\ G\ \}$$

Definition 24 (Independent set). *A graph G is null or edgeless when it has no edge, that is, $E(G) = \emptyset$. A set of vertices I of a graph G is an independent set if the subgraph I of G, G[I], is a graph without edges. We define by $\alpha(G)$ the size of the maximum independent set, that is:*

$$\alpha(G) = max\ \{\ |V'|\ /\ V' \subseteq V\ and\ V'\ is\ an\ independent\ set\ of\ G\ \}$$

A point to emphasize is that every *edgeless graph* is an *independent set*. However, not every *independent set* is a *edgeless graph*.

Definition 25 (Coloring). *A coloring of a graph G is a partition of V(G) where each class of the partition is an independent set. A k-coloring is a partition of V(G) into k classes. The chromatic number of G, denoted by $\chi(G)$, is the smallest k for which there is a k-coloring of G. In this case, we say that the graph G is k-chromatic or k-colorable.*

Definition 26 (Connected/ Disconnected graph, Connected component)**.** *A graph G is connected if for every pair of distinct vertices v and w of $V(G)$ there is a path from v to w. Otherwise, G is said to be disconnected. A connected component of G is a maximally connected subgraph of G.*

Definition 27 (Tree)**.** *A tree is an acyclic and connected graph. A tree T is said to be rooted when some vertex $v \in V(T)$ is chosen and thus classified as the root of the tree. In this way, the graphical representation becomes the root at the top and all the other vertices of the tree below it, as if the tree were "hanging" by the root, thus having a clear hierarchy between the vertices. Let v, w be two vertices of a rooted tree T with root r. If v belongs to the path from r to w in T, then we call v an ancestor of w and w a descendant of v. In particular, if $(v,w) \in E(T)$ then v is the parent of w, denoted by parent(w), w being the child of v, denoted by child(v). Two vertices that have the same parent are called siblings. The root r of a tree has no parent, while every vertex $v \neq r$ has a single parent. A leaf is a vertex that has no children. The level of a vertex v, level(v), of a tree is called the distance from the root r to v, that is, the number of edges that separate the vertex v from the root and the height of the tree T is the highest of the levels.*

Definition 28 (Complement of a graph)**.** *The complement of a graph G is a graph $\overline{G}$ wiyh the same set of vertices as G such that there will be an edge between two vertices (v, e) in $\overline{G}$, if and only if there is no edge in between (v, e) in G.*

Definition 29 (Bipartite graph)**.** *A graph G is bipartite when the set of vertices of G can be partitioned into two subsets, V_1 and V_2, such that every edge of G joins a vertex of V_1 to another of V_2, that is, it is possible to partition $V(G)$ into two independent sets. A graph is said to be complete bipartite if it is bipartite and has edges connecting each of the vertices of V_1 to each of the vertices of V_2. That is, for each pair of vertices v_1, v_2, being $v_1 \in V_1$ and $v_2 \in V_2$, there is an edge connecting them. We denote by $K_{n,m}$ the complete bipartite graph, where $|V_1| = n$ e $|V_2| = m$. A graph is k-partied when its set of vertices can be partitioned into k independent sets. A graph is split if its set of vertices can be partitioned into an independent set and a clique.*

Definition 30 (Perfect graph)**.** *A graph G is perfect if and only if for every induced subgraph H of G, the size of the largest clique of H is equal to the chromatic number of H, that is, $\chi(H) = \omega(H)$.*

For this work, we used the notations and standard concepts of Graph Theory. It is worth mentioning that any notation that was not defined/mentioned in this section can be found in (BONDY; MURTY, et al., 1976).

2.2 Complexity of algorithms

In recent years, the Computing area has witnessed significant advances, particularly in the domain of Algorithms and Complexity Theory, where the focus lies on creating and

investigating computational algorithms. This field aims to delve into problems that can be effectively tackled with computers, seeking to generate the most optimal solutions for such problems. Additionally, it involves classifying these problems into distinct categories based on their level of complexity or difficulty, including those that may be intractable. As a result, as the field continues to evolve, the study of algorithms and complexity theory remains instrumental in the pursuit of computational efficiency and problem classification.

2.2.1 Time Complexity

Within the scope of Graph Theory, the study of algorithmic problems and their time complexity plays a fundamental role. Formally, an algorithmic problem consists of a set D of all possible inputs to the problem, called *set of instances*, and a question Q to be answered.

The resolution of an algorithmic problem depends on the development of an algorithm capable of taking an instance of the problem as input and producing an output that effectively answers the question posed by the problem. This algorithm serves as a computational tool to navigate through the complexities of the problem and provide a desired solution.

The term "complexity" generally refers to the required resources for an algorithm to solve a problem from a computational perspective. Specifically, time complexity, or simply complexity, characterizes the time requirements of an algorithm. It can be defined as a function that operates on the size of problem instances, expressing the time needed for the algorithm to solve the problem for various input sizes. In essence, it quantifies the maximum amount of time the algorithm can require to solve a problem of a particular size.

Understanding the time complexity of algorithms in the context of Graph Theory is crucial for assessing their efficiency and predicting their performance in different instances of the problem. By analyzing the growth patterns and behavior of time complexity functions, researchers and practitioners can make informed decisions about algorithm selection, optimization strategies, and ultimately obtain more effective problem-solving results. References such as the works by (CORMEN et al., 2009) or (DASGUPTA; PAPADIMITRIOU; VAZIRANI, 2006) delve deeper into the topic of time complexity analysis and its significance in Graph Theory and beyond.

2.2.2 Classification of Problems

In relation to the categorization of problems, there are three distinct types that can be identified. A problem is classified as an *optimization problem* when the objective is to find the best feasible solution among a given set of possibilities. On the other hand, an *evaluation problem* is characterized by the need to calculate the cost associated with an ideal feasible solution. Lastly, a problem is said to be a *decision problem* when the question to be observed/analyzed requires an answer in the form of a definitive “YES” or “NO” answer. Below is an example of a decision problem that we denote as p, to illustrate this concept.

Let π represent the following problem: *"Given a graph G, determine whether G is a near-bipartite graph."*

Considering the π problem, its set of instances consists of all possible graphs. Consequently, we can represent the π problem as follows:

GENERIC INSTANCE OF π: A graph G.
QUESTION: Is G a near-bipartite graph?

It is evident that the aforementioned problem π is actually a decision problem, more specifically, a recognition problem. To solve the problem π it is necessary to develop an algorithm capable of recognizing near-bipartite graphs.

2.2.3 Complexity classes

As for the classification of algorithms, we refer to an algorithm as polynomial if its time complexity, which measures the number of steps the algorithm takes, can be expressed as a polynomial function of the input size. In other words, polynomial algorithms are those that consume a reasonable amount of time, and their computation times don't grow excessively fast as the problem size increases. The class of decision problems for which polynomial algorithms exist is known as P, and such problems are referred to as *polynomial problems*.

A decision problem is considered polynomial non-deterministic when every instance that produces a "YES" answer has a concise certificate. In essence, the algorithm can generate a proof that the "YES" answer can be verified within a reasonable amount of time for the given instance size. This class of decision problems is denoted as NP. On the other hand, the Co-NP class consists of problems that have a succinct certificate for the instances that produce a response "NO".

Regarding the relation between the P and NP classes, it is established that P is a subset of NP, meaning that any decision problem solvable by a polynomial-time deterministic algorithm can also be solved by a polynomial-time non-deterministic algorithm. In fact, if a problem Π belongs to P and α represents any deterministic polynomial-time algorithm for Π, we can construct a non-deterministic polynomial-time algorithm for Π simply by utilizing α as a recognition algorithm to justify the "YES" answer for Π. Therefore, if Π belongs to P, it also belongs to NP. However, the question of whether P is a subset of NP ($P \subseteq NP$) or whether P is equal to NP ($P = NP$) remains an unresolved problem.

Another important class of problems is NP-complete, which comprises NP problems that have the property that if any one of them can be solved in polynomial time, then every other NP-complete problem can also be solved in polynomial time. In other words, we can define NP-Complete *problems* as a class that represents the set of all problems X in NP for which it is possible to reduce any other NP problem Y to X in polynomial time.

Let π_1 (D_1, Q_1) and π_2 (D_2, Q_2) be two decision problems. A transformation or polynomial reduction from problem π_1 to π_2 is a function f, $f : D_1 \to D_2$, that satisfies two conditions: The first one is f can be computed in polynomial time, and the second

one is for every instance $I \in \pi_1$, I produces a "YES" answer for π_1 if and only if $f(I)$ produces a "YES" answer for π_2.

To prove that a certain problem π is NP-complete, it suffices to demonstrate that $\pi \in NP$ and for every problem $\pi' \in NP$ there is a polynomial transformation of π' into π.

Similarly, a decision problem π belongs to the class Co-NP-complete (and is referred to as Co-NP-complete) when $\pi \in$ Co-NP and there exists a (Co-)NP-complete problem π' such that:

(1) if π' is NP-complete, there is a function f computable in polynomial time where for every instance I' of π', I' produces a "YES" answer for π' if and only if $I = f(I')$ produces a "NO" answer for π; and

(2) if π' is Co-NP-complete, there is a function f computable in polynomial time where for every instance I' of π', if I' produces a "NO" answer for π' if and only if $I = f(I')$ produces a "NO" answer for π.

Finally, we have the class of NP-hard problems, which represents the computational complexity of decision problems that are at least as hard as problems in the NP class (non-deterministic polynomial time). These problems are considered among the most challenging to solve efficiently. The field of Graph Theory, with its diverse applications in various domains, presents a vast number of NP-hard problems to be analyzed, like the well-known Traveling Salesman Problem (TSP), the Graph Coloring Problem, and the Hamiltonian Cycle Problem. These problems have been much studied due to their computational intractability and wide implications such as optimization, scheduling and other practical domains. Consequently, the literature presents several proposals for approximation algorithms and heuristic techniques to solve these NP-hard graph problems, trying to find a balance between solution quality and computational efficiency. Graph theory provides a rich foundation for exploring the complexities of NP-hard problems and developing innovative approaches to tackle their complexity (GAREY; JOHNSON, 1979; CORMEN et al., 2009).

As a reference for this section, we recommend (GAREY; JOHNSON, 1979) and (SZWARCFITER, 1988).

2.3 Boolean formulas and satisfiability problems

Satisfiability problems involve determining the existence of satisfactory assignments to logical formulas by assigning truth values to their variables. In other words, their goal is to determine whether a given logical formula in conjunctive normal form (CNF) can be satisfied.

A boolean expression in CNF is a standardized way of representing a logical formula using conjunctions ($\wedge$) and disjunctions ($\vee$). In this work, we adopt the notation ($\cdot$) to represent conjunctions and ($+$) to represent disjunctions.

In CNF, the boolean expression is written as a conjunction of clauses, where each clause is a disjunction of literals (boolean variables or their negations). Each clause is

enclosed in parentheses. As for the literals, x_i and $\overline{x}_i$, they represent, respectively, the positive and negative literals of variable x_i. Below, Figure 1 is an example of a boolean expression in CNF:

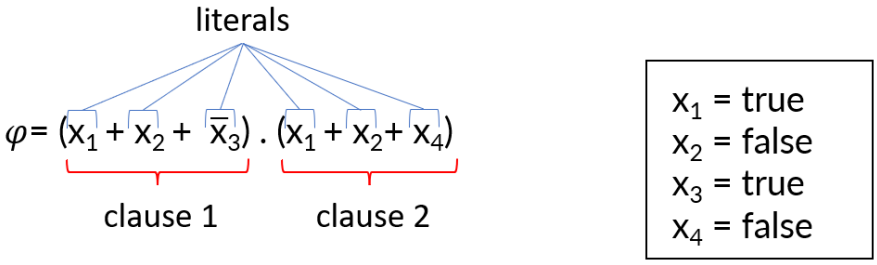

Figure 1: Example of a boolean expression in 3-CNF.

In the expression above, we have two clauses: $(x_1 + x_2 + \overline{x}_3)$ and $(x_1 + x_2 + x_4)$. Additionally, it can be verified that this boolean formula is in 3-CNF. That is, it is a boolean expression composed of two clauses, where each of them has exactly three literals connected by logical OR operators represented by $(+)$. The clauses are connected by AND operators represented by $(\cdot)$.

As can be observed, there exists a possible assignment of truth values to the variables $x_1 \ldots x_4$ that makes the expression φ *true*. Therefore, we say that φ is *satisfied*.

Satisfiability problems are much studied in the area of computer science and mathematics due to their wide applications. In this section, we will discuss some of them that were used during the development of this work.

2.3.1 2SAT

The 2SAT PROBLEM or 2-SATISFIABILITY is a well-known polynomial problem (P) in graph theory and computational complexity (ASPVALL; PLASS; TARJAN, 1982). It belongs to the class of Boolean satisfiability problems. In 2SAT, the logical formula consists of clauses, which are disjunctions of literals connected by logical OR operators, with each clause containing exactly two literals (2-CNF). The task is to find an assignment of truth values to the variables that satisfies all the clauses.

2SAT

Instance: A Boolean formula composed of 2-CNF clauses where the literals can be Boolean variables or their negations.

Goal: Find (if any) an assignment of *true* or *false* values to the variables that satisfies all the clauses in the 2-CNF formula.

The 2SAT problem has applications in various areas, including formal verification, artificial intelligence, and optimization. It has been extensively studied in both graph theory and computational complexity. Efficient algorithms, such as the famous Tarjan's

algorithm, have been developed to solve 2SAT in linear time, making it a fundamental problem in the field.

2.3.2 1-in-3SAT

The 1-IN-3-SAT problem is a variant of the well-known satisfiability problem, which plays a crucial role in graph theory and theoretical computer science. The 1-IN-3-SAT problem is known to be NP-complete (GAREY; JOHNSON, 1979).

In this problem, we are given a Boolean formula in conjunctive normal form (CNF) where each clause contains exactly three literals. The goal is to determine if there exists an assignment of truth values to the variables such that exactly one literal in each clause evaluates to true.

1-IN-3SAT

Instance: A Boolean formula composed of 3-CNF clauses where the literals can be Boolean variables or their negations.
Goal: Find (if any) an assignment of *true* or *false* values to the variables that satisfies at least one literal in each 3-CNF clause.

2.3.3 Positive-Min-Ones-2SAT

The POSITIVE-MIN-ONES-2SAT is a relevant NP-complete problem in the field of graph theory and Boolean satisfiability (MISRA; NARAYANASWAMY, et al., 2013). In this problem, the objective is to determine whether a given Boolean formula in conjunctive normal form (CNF) has a truth value assignment, minimizing the number of variables that have been assigned *true*.

In a CNF formula, a clause is a disjunction of literals, where each literal represents a Boolean variable. In the POSITIVE-MIN-ONES-2SAT problem, each clause must consist of exactly two literals. The goal is to find a truth assignment that makes the entire formula true, with the additional constraint of minimizing the number of variables set to *true*.

POSITIVE-MIN-ONES-2SAT

Instance: A set of 2-CNF clauses where the literals are Boolean variables.
Goal: Find (if any) an assignment of *true* or *false* values to the variables that minimizes the number of variables set to true in the clauses.

Furthermore, this problem can be naturally related to the *Minimum Vertex Cover* problem in graph theory. In a graphical interpretation, Boolean variables can be represented as vertices in a graph, and 2-CNF clauses can be mapped to edges in the graph.

Each clause C_i corresponds to an edge in the graph, connecting two vertices representing the variables v_i and w_i. The task of minimizing the number of *true* variables in

the clauses is equivalent to selecting the smallest possible number of vertices in such a way that each edge of the graph is incident to at least one selected vertex.

2.4 Graph Problems

Throughout this section, we will explore some graph problems that were fundamental for the development of this work. These problems have been widely studied in the field of graph theory.

2.4.1 Vertex Cover

The VERTEX COVER problem is a fundamental problem in graph theory that has garnered significant attention in both theoretical and practical domains. In simple terms, the problem aims to find the smallest set of vertices in a graph such that every edge in the graph is incident to at least one vertex in the set. This set of vertices is known as a vertex cover.

Formally defined, a vertex cover in a graph G is a subset V' of the vertex set V, such that for every edge (u, v) in G, at least one of the vertices u or v is in V'. The goal is to identify the smallest possible vertex cover in the given graph.

VERTEX COVER

Instance: A simple undirected graph $G = (V, E)$.
Goal: Find (if any) the smallest set of vertices such that every edge of the graph has at least one of its endpoints belonging to this set.

The VERTEX COVER problem is one of the classic NP-complete problems (KARP, 1972) studied in the field of graph theory.

2.4.2 Independent Set

The INDEPENDENT SET problem, as its name suggests, aims to find the largest independent set of vertices in a graph. An independent set is a subset of vertices in which no two vertices are adjacents. The goal is to identify the largest possible subset of vertices that satisfies this condition.

Formally, an independent set in a graph G is a subset of vertices V' such that for every pair of vertices u and v in V', there is no edge between u and v in G. The objective of the INDEPENDENT SET problem is to identify the largest independent set in the given graph.

INDEPENDENT SET

Instance: A simple undirected graph $G = (V, E)$.
Goal: Find (if any) the largest set in which no pair of vertices are connected by an edge.

The INDEPENDENT SET problem is widely recognized and extensively studied in the field of graph theory. It is known to be NP-complete (KARP, 1972), indicating its computational complexity.

2.4.3 Feedback Vertex Set

The FEEDBACK VERTEX SET problem is a fundamental problem in graph theory that focus on finding the smallest set of vertices in a graph whose removal eliminates all cycles. In other words, the goal is to identify a vertex set that, when removed from the graph, transforms it into an acyclic graph.

Formally defined, a feedback vertex set in a graph G is a subset of vertices V' such that the removal of these vertices from G results in a graph without any cycles. The objective of the FEEDBACK VERTEX SET problem is to determine the smallest possible feedback vertex set in the given graph.

INDEPENDENT SET

Instance: A simple undirected graph $G = (V, E)$.
Goal: Find (if any) the smallest set of vertices such that, after removing these vertices and their incident edges, the resulting is an acyclic graph.

The FEEDBACK VERTEX SET is a well-known NP-complete problem (KARP, 1972), indicating its computational complexity in finding an optimal solution. Researchers have proposed various approximation algorithms and heuristics to efficiently find near-optimal solutions for this problem.

3 Near-bipartition problem in P_5-free graphs and graphs with dominating edge

3.1 Near-bipartition problem

In this chapter, we will address the problem of NEAR-BIPARTITENESS and other relevant problems considered for the development of this dissertation. This problem is currently a subject of extensive study and presents itself as a complex challenge involving the analysis of graphs and the identification of near-bipartite structures. By delving into this problem in detail, we aim to understand its characteristics and properties for graphs having a dominating edge and for P_5-free graphs, which will be discussed in Chapter 3.2. Additionally, we will discuss other related problems that are fundamental to the construction and improvement of the work carried out. By analyzing these issues, our intention is to provide valuable insights for the advancement of the field and contribute to more efficient and effective solutions in this specific domain.

Since a near-bipartition $(\mathcal{S},\mathcal{F})$ of a graph G is a partition of $V(G)$ into an independent set $\mathcal{S}$ and an induced forest $\mathcal{F}$, we consider the following problems.

The problem of INDEPENDENT FEEDBACK VERTEX SET (IFVS) is a well-studied problem in graph theory that combines the concepts of *feedback vertex set* and *independent set*. A set $\mathcal{S}$ of vertices in a graph G is a *feedback vertex set* of G if removing the vertices in $\mathcal{S}$ results in an acyclic graph, i.e., the graph $G - S$ is a forest.

The INDEPENDENT FEEDBACK VERTEX SET problem considered in this work is formulated as follows: given an undirected graph G with its set of vertices and edges, the goal is to find a minimum-sized independent feedback vertex set of G. In other words, we aim to determine the existence of a near-bipartition $(\mathcal{S},\mathcal{F})$ of G that minimizes the size of $\mathcal{S}$. Below, in Figure 2, is an example of graph G that admits an independent feedback vertex set.

It is important to mention, not every graph admits an independent feedback vertex set (consider complete graphs on at least four vertices). Graphs that do admit an independent feedback vertex set are said to be *near-bipartite*, and we can ask about recognizing these graphs.

INDEPENDENT FEEDBACK VERTEX SET

Instance: A simple undirected graph $G = (V, E)$.
Goal: Find (if any) a minimum independent feedback vertex set of G, i.e., a near-bipartition $(\mathcal{S},\mathcal{F})$ of G that minimizes the size of $\mathcal{S}$.

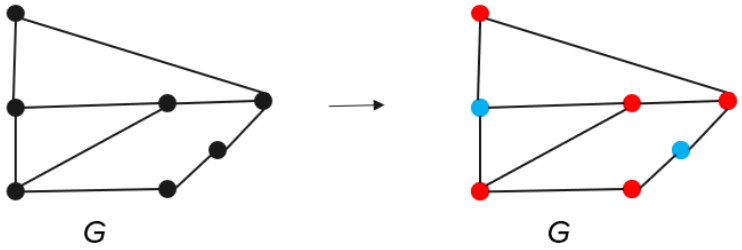

Figure 2: Example of INDEPENDENT FEEDBACK VERTEX SET problem.

Another problem analyzed was the NEAR-BIPARTITENESS problem, which holds significant importance in Graph Theory. In this problem, given an undirected graph, the objective is to determine if it is possible to partition the set of vertices into two parts, $\mathcal{S}$ and $\mathcal{F}$, where $\mathcal{S}$ forms an independent set and $\mathcal{F}$ forms an acyclic set (i.e., induces a forest). In Figure 3 we have an example of a graph G which is near-bipartite.

The NEAR-BIPARTITENESS problem is known to be NP-complete, even for graphs with a maximum degree of 4 (YANG; YUAN, 2006) or diameter 3 (BONAMY et al., 2017), as mentioned in Chapter 1.

Studying NEAR-BIPARTITENESS is relevant from both a theoretical and practical perspective. Theoretically, the problem provides insights into the complexity of graph partitioning problems and their relationships with other classical problems. Furthermore, understanding the structure of graphs that admit a near-bipartition can lead to the development of efficient algorithms for solving related problems.

NEAR-BIPARTITENESS

Instance: A simple undirected graph $G = (V,E)$.
Question: Does G have a near-bipartition $(\mathcal{S},\mathcal{F})$?

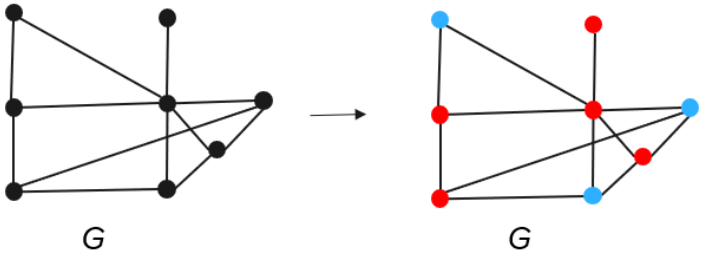

Figure 3: Example of NEAR-BIPARTITENESS problem.

Another problem addressed in this work was the Acyclic Vertex Cover. This problem involves finding a minimum-sized set of vertices in an undirected graph that covers all the edges of the graph while ensuring that the chosen set of vertices does not induce any cycles in the resulting graph. In other words, the goal is to find a set of vertices that covers all the edges of the graph in an acyclic manner (See an example in Figure 4).

Furthermore, the Acyclic Vertex Cover problem is related to other challenges in graph theory, such as the Independent Feedback Vertex Set problem and the Vertex Cover problem which further enhances its impact and relevance in graph research.

Acyclic Vertex Cover

Instance: A simple undirected graph $G = (V, E)$.
Goal: Find (if any) a minimum acyclic vertex cover of G, i.e., a near-bipartition $(\mathcal{S},\mathcal{F})$ of G that minimizes the size of $\mathcal{F}$.

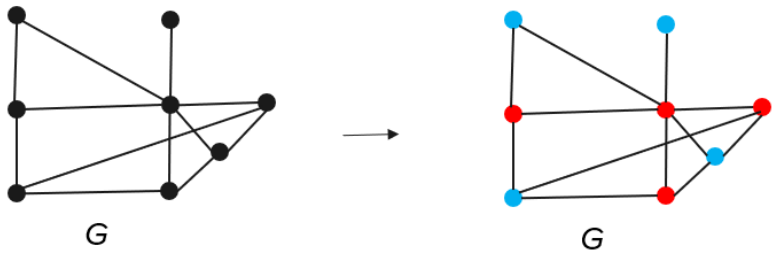

Figure 4: Example of Acyclic Vertex Cover problem.

Also, we consider the problem of determining whether a graph G can have its set of vertices partitioned into an independent set and a *tree*, called Connected Near-Bipartiteness See an example in Figure 5), which was shown to be NP-complete even on bipartite graphs of maximum degree four by Brandstädt, Le, and Szymczak (BRANDSTÄDT; LE; SZYMCZAK, 1998).

Connected Near-Bipartiteness

Instance: A simple undirected graph $G = (V,E)$.
Question: Does G have a near-bipartition $(\mathcal{S},\mathcal{F})$ such that $G[\mathcal{F}]$ is connected?

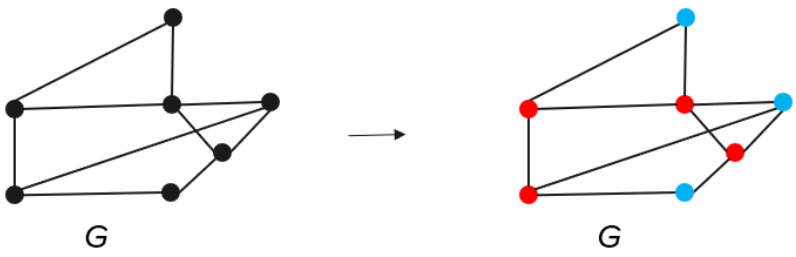

Figure 5: Example of Connected Near-Bipartiteness problem.

3.2 P_5-free graphs and graphs with dominating edge

In Graph Theory, the study of graph classes is of great importance to understand and characterize specific properties of these structures. In this chapter, we will discuss two particular classes of graphs: graphs having a dominating edge and P_5-free graphs. We will explore their definitions and properties.

3.2.1 Graphs having a dominating edge

.

Graphs with dominating edges are objects of study in the field of Graph Theory, with a wide range of research and results dedicated to this class of graphs. Graphs with dominating edges have a special property related to dominance. Given a graph G, it is said to be a *graph having a dominating* edge if it has an edge $e = (e_1, e_2) \in E(G)$ such that for every vertex $v \in V(G)$, $v \neq e_1$ and $v \neq e_2$, either v is adjacent to e_1 or v is adjacent to e_2.

The class of graphs having a dominating edge is a natural subclass of graphs with diameter three, a class for which NEAR-BIPARTITENESS remains NP-complete (BONAMY et al., 2018). Concerning this special class of graphs, we present a polynomial-time algorithm for NEAR-BIPARTITENESS and prove that CONNECTED NEAR-BIPARTITENESS, the variant where the forest must be connected, is NP-complete. In addition, we show that INDEPENDENT FEEDBACK VERTEX SET, the problem of finding a near-bipartition $(\mathcal{S},\mathcal{F})$ minimizing $|\mathcal{S}|$, and ACYCLIC VERTEX COVER, the problem of finding a near-bipartition $(\mathcal{S},\mathcal{F})$ minimizing $|\mathcal{F}|$, are both NP-hard when restricted to graphs having a dominating edge.

3.2.2 P_5-free graphs

P_5-free graphs are a special class of graphs that do not contain any subgraph isomorphic to a path of five vertices (P_5), see Definition 16. This class of graphs has been extensively studied due to its interesting properties.

The study of P_5-free graphs dates back to the late 1960s when researchers began exploring the structural properties of graphs that did not contain fixed-size paths. This category of graphs has since been extensively investigated in the field of Graph Theory.

For instance, Hoàng et al. (HOANG et al., 2010) demonstrated that the k-Coloring problem can be solved in polynomial time for P_5-free graphs, where k is any integer. Additionally, Golovach and Heggernes (GOLOVACH; HEGGERNES, 2009) established that Choosability becomes tractable for P_5-free graphs when the parameters are fixed, specifically considering the size of the lists of admissible colors.

Furthermore, Lokshantov et al. (LOKSHTANOV; VATSHELLE; VILLANGER, 2014) made a significant contribution by addressing a longstanding unresolved problem. They provided a polynomial-time algorithm for Independent Set on P_5-free graphs.

Finally, we also highlight that Bonamy et al. (BONAMY et al., 2019) proved that

Near-Bipartiteness on P_5-free graphs can be solved in $O(n^{16})$ time. Regarding this problem in the analyzed cases for P_5-free graphs in this work, we demonstrate that the problem of Near-Bipartiteness can be solved in $O(n^4)$ time for a graph without P_5, improving upon the existing literature result of $O(n^{16})$ mentioned above.

Thus, in Chapter 4 and Chapter 5, we will present each of the analyzed cases for these special graph classes. The discovered findings contribute to a better understanding of the problems and provide more efficient solutions in practical scenarios.

3.2.3 Auxiliary Lemmas

In this section, we will present two auxiliary lemmas that were used during the mathematical proofs. These lemmas pertain to certain recurring aspects in the proofs of specific cases analyzed. The creation of this subsection aimed to reduce repetitions of sections and/or ideas throughout the proofs, avoiding excessively lengthy and tiresome analyses.

Lemma 1. *A graph G admits an independent set A such that $G[V - A]$ is an edgeless graph if and only if G is bipartite.*

Proof. Assume that G is bipartite, meaning that its vertex set V can be partitioned into two disjoint subsets V_1 and V_2 such that all edges in G connect vertices from V_1 to V_2. Let A be one of these subsets, say V_1, and let B be the other subset, V_2. Then, A is an independent set (independent set) because there are no edges between vertices within A or within B.

Now, consider the graph $G[V-A]$, which is obtained by removing all vertices in A and their incident edges from G. Since A is an independent set, there are no edges between A and B, so removing A does not affect the edges between B and the remaining vertices in $G[V - A]$. Therefore, $G[V - A]$ is still an edgeless graph.

Conversely, suppose that G admits an independent set A such that $G[V - A]$ is an edgeless graph. We want to show that G is bipartite.

Let's create two sets, V_1 and V_2, where V_1 contains the vertices in A, and V_2 contains the vertices in $V - A$. Since A is an independent set, there are no edges within A or within V_2. Also, since $G[V - A]$ is an edgeless graph, there are no edges between V_1 and V_2, as removing A does not create any new edges between these sets.

Therefore, we have successfully partitioned the vertex set V into two disjoint subsets, V_1 and V_2, such that all edges in G connect vertices from V_1 to V_2. This defines G as a bipartite graph. □

Lemma 2. *Given a graph G with a dominating edge (u,v), if G admits a partition $(\mathcal{S},\mathcal{T})$ with $(u,v) \in E(T)$ then $N(u,v)$ induces a bipartite graph.*

Proof. Let G be a graph with a dominating edge (u, v). We observe that this dominating edge cannot belong to any independent set in G. Consequently, vertices u and v cannot

be part of the same independent set. Suppose that G admits a partition $(\mathcal{S},\mathcal{T})$ where (u, v) belongs to $E(\mathcal{T})$, implying that $\{u, v\} \in \mathcal{T}$.

Given that graph G admits a partition $(\mathcal{S},\mathcal{T})$ where (u, v) belongs to $E(\mathcal{T})$, we conclude that the graph $G - \{u,v\}$ must induce a bipartite graph. In other words, the set of vertices of G with the dominating edge removed must be partitionable into two independent sets A_1 and A_2, where all vertices to be placed in A_1 belong to partition $\mathcal{T}$, and the vertices belonging to A_2 are part of partition $\mathcal{S}$.

Thus, we have shown that $N(u, v)$ (the vertices adjacent to u and v) induces a bipartite graph. Therefore, $N(u, v)$ can be partitioned into two disjoint subsets A_1 and A_2.

□

4 Results obtained in graphs having a dominating edge

In this chapter, we will present our findings regarding the study of partitioning a graph with a dominating edge into a stable set and a *tree* (Connected Near-Bipartiteness), as well as the problem of partitioning it into a stable set and a forest (Near-Bipartiteness).

Next, we show that Connected Near-Bipartiteness is NP-complete on graphs having a dominating edge, while Near-Bipartiteness becomes solvable in polynomial time within the same class.

Theorem 1. Connected Near-Bipartiteness *is NP-complete even when restricted to graphs having a dominating edge.*

Proof. The proof is based on a reduction from 1-in-3SAT, a well-known NP-complete problem (GAREY; JOHNSON, 1979). In such a problem we are given a formula φ in conjunctive normal form where each clause contains exactly three literals and asked whether there exists a satisfying assignment so that exactly one literal in each clause is set to true.

Given an instance φ of 1-in-3SAT, we construct a graph G such that φ has a truth assignment such that each clause has exactly one literal set to true if and only if G is partitionable into a stable set and a tree.

From φ we construct G as follows:

1. first consider $G = (\{u, v\}, \{uv\})$;
2. add a chordless cycle C of size 4 in G induced by $\{k_1, k_2, k_3, k_4\}$, and add edges from u for all vertices in C;
3. add a chordless cycle $C' = l_1, m, l_2, n_1, n_2$;
4. add the edges ul_1, ul_2, vm, vn_1 and vn_2;

   At this point, notice that every $(\mathcal{S},\mathcal{T})$-partition of G has $v \in \mathcal{S}$ and $u \in \mathcal{T}$.
5. for each variable x_i of φ create vertices v_{x_i} and $v_{\overline{x}_i}$ and add edges $v_{x_i}v_{\overline{x}_i}$, uv_{x_i} and $uv_{\overline{x}_i}$;
6. for each clause C_j of φ create a vertex c_j in G and add the edge vc_j;

7. Finally, add an edge $c_j v_{x_i}$ if the clause C_j contains the literal x_i, and add an edge $c_j v_{\overline{x}_i}$ if the clause C_j contains the literal $\overline{x}_i$. (see Figure 6 below)

It is worth noting that there is no possibility of cycle formation in the induced subgraphs that remain in the graph G constructed from the formula φ of 1-IN-3SAT. Firstly, chordless cycles are added, such as the cycle C of size 4 induced by $\{k_1, k_2, k_3, k_4\}$, which is cycle-free internally. Then, a second chordless cycle C' is added. Furthermore, vertices u and v are connected to these cycles in a way that keeps all cycles separate, and there are no additional vertices that form cycles between them.

Moreover, during the rest of the construction, vertices and edges are added in a manner that maintains this property of chordless cycles. Therefore, the structure of the graph G is such that there is no opportunity for cycle formation in the induced subgraphs that remain after partitioning, which is crucial for the success of the reduction.

The Figure 6 shows a example of a graph, denoted as G, constructed from a 3-CNF formula $\varphi = (x_1 + x_2 + x_4) \cdot (\overline{x}_2 + x_3 + x_4).(x_2 + x_3 + \overline{x}_4)$ according to Theorem 1. Furthermore, it presents an $(\mathcal{S},\mathcal{T})$-partition of G derived from a 1-in-3 truth assignment of φ. In this representation, the white vertices form the stable set, while the black vertices form the tree.

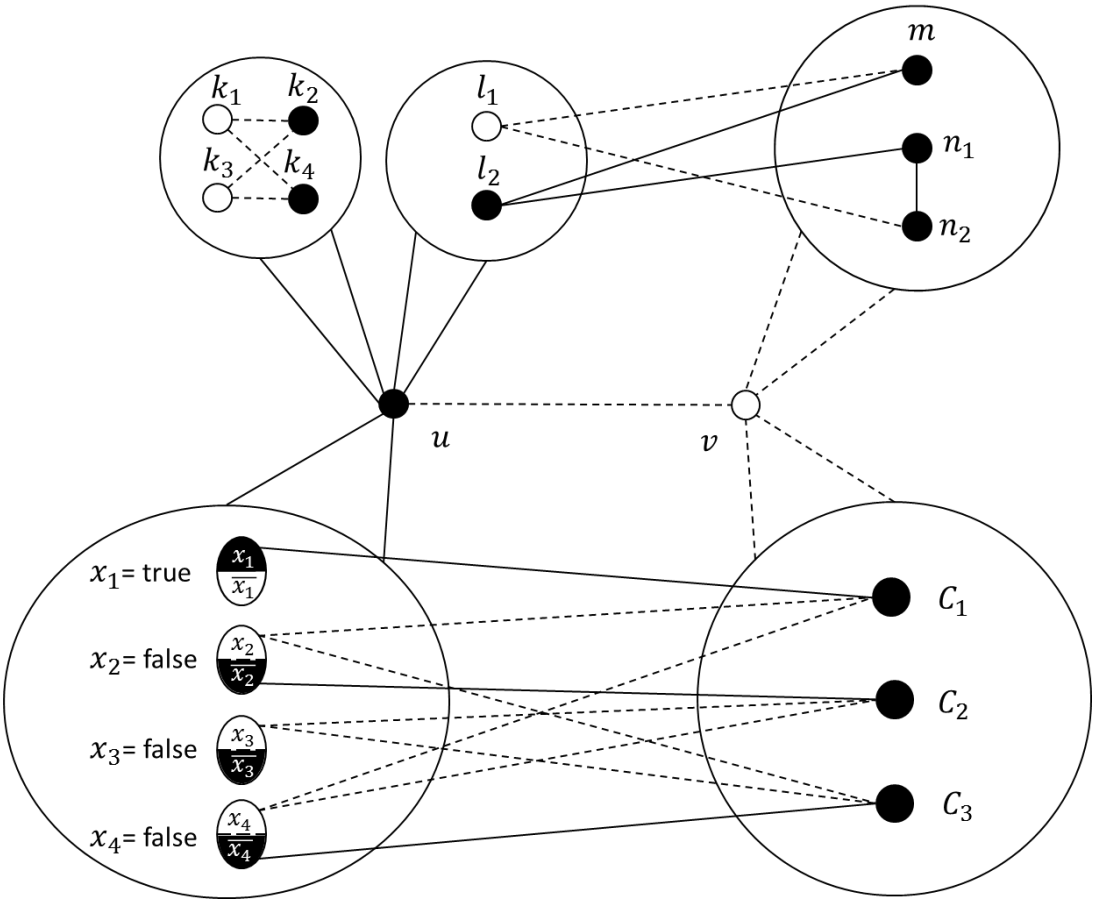

Figure 6: Example of graph G and an near-bipartition representation from the formula $\varphi = (x_1 + x_2 + x_4) \cdot (\overline{x}_2 + x_3 + x_4) \cdot (x_2 + x_3 + \overline{x}_4)$,

.

If φ is a 3-CNF formula having a truth assignment A such that each clause has exactly one literal set as true, then we can construct an $(\mathcal{S},\mathcal{T})$-partition of G by setting $\mathcal{S} = \{k_1, k_3, l_1, v\} \cup \{v_{x_i} : x_i = false \in A\} \cup \{v_{\overline{x}_i} : x_i = true \in A\}$ (clearly $\mathcal{S}$ is a stable set). Since A defines a 1-in-3 truth assignment then each vertex c_j has exactly one neighbor in $G[V \setminus \mathcal{S}]$ then $\mathcal{T} = V \setminus \mathcal{S}$ induces a tree.

Conversely, if G admits an $(\mathcal{S},\mathcal{T})$-partition then, by construction, it holds that $v \in \mathcal{S}$ and $u \in \mathcal{T}$. This implies that every vertex c_j belongs to $\mathcal{T}$, and that for each pair v_{x_i}, $v_{\overline{x}_i}$ exactly one of these vertices belongs to $\mathcal{T}$. Also, since $\mathcal{T}$ is connected each c_j has at least one neighbor in $\mathcal{T}$, thus as $\mathcal{T}$ is acyclic each vertex c_j has exactly one neighbor in $\mathcal{T}$ (each c_j must be a leaf in $\mathcal{T}$). Therefore, we can construct a 1-in-3 truth assignment by setting $x_i = true$ iff $v_{x_i} \in \mathcal{T}$. □

Contrasting with Theorem 1, we show that when we remove the connectivity constraint, i.e., we look for a forest instead of a tree, the problem becomes polynomial-time solvable.

Theorem 2. *Given a graph G and a dominating edge of G, one can determine in $O(n^2)$ time whether G is a near-bipartite graph.*

Proof. Let $u, v \in V(G)$ be two vertices of G such that uv is a dominating edge of G. Suppose that G has a near-bipartition $(\mathcal{S},\mathcal{F})$. Without loss of generality, we may assume that G does not have vertices with degree one. At this point, we may consider just two cases:

Case 1. Suppose that u, $v \in \mathcal{F}$.

As $uv \in E(F)$, then $N(u) \cap N(v) \subseteq \mathcal{S}$, otherwise $\mathcal{F}$ has cycles. Thus, $N(u) \cap N(v)$ must be a stable set. For a remaining vertex w belonging to either $N(u) \setminus N(v)$ or $N(v) \setminus N(u)$: if it has a neighbor in $\mathcal{S}$ then it must belong to $\mathcal{F}$; if it has a neighbor z ($z \neq u$ and $z \neq v$) that must be in $\mathcal{F}$, then w must belong to $\mathcal{S}$, otherwise, the edge wz together with uv induces a cycle in $\mathcal{F}$. Thus, by checking if $N(u) \cap N(v)$ is stable and then successively applying the operations previously described according to a Breadth-First Search from $N(u) \cap N(v)$, in linear time, we can either conclude that such a near-bipartition with $u, v \in \mathcal{F}$ does not exist, or build a partition (S', F', U) of $V(G)$ such that S' is stable, $F' \supseteq \{u, v\}$ induces a forest, and U is the set of unclassified vertices. Note that, by construction, no vertex in U has neighbors in $S' \cup F' \setminus \{u, v\}$. Since any pair of adjacent vertices together with u and v induces a cycle, G has a near-bipartition $(\mathcal{S},\mathcal{F})$ with $\{u, v\} \subseteq \mathcal{F}$ if and only if $G[U]$ has an independent vertex cover, which is equivalent to U inducing a bipartite graph (see Lemma 2).

Case 2. Suppose that $u \in \mathcal{S}$ and $v \in \mathcal{F}$.

If $u \in \mathcal{S}$ and $v \in \mathcal{F}$ then $N(u) \subseteq \mathcal{F}$. Thus, $N(u)$ must induce a forest and $N(u) \cap N(v)$ must be a stable set. At this point, only the vertices belonging to $N(v) \setminus N[u]$ are unclassified.

Let $B = N(v) \setminus \{u\}$.

If G has a near-bipartition $(\mathcal{S},\mathcal{F})$ then $G[B]$ must be bipartite, so that its vertices can be partitioned into two sets (B_1, B_2) such that $B_1 \subseteq \mathcal{S}$ and $B_2 \subseteq \mathcal{F}$. Thus, we must find a bipartition of B that satisfies the following conditions:

- $N(u) \cap N(v) \subseteq B_2$;
- for each component T of $G[N(u) \setminus N[v]]$ (which is a tree) it holds that:
  - For each $w \in B_2$, $|N_T(w)| \leq 1$ (otherwise $\{w\} \cup V(T)$ induces a cycle);
  - T has at most one neighbor in B_2 (otherwise $\mathcal{F}$ has cycles).

Note that any bipartition (B_1, B_2) satisfying the above restrictions is sufficient to form a near-bipartition such that $B_1 \cup \{u\} = \mathcal{S}$. Now, we can reduce the problem of finding such a bipartition of $G[B]$ to the 2SAT problem by building a 2-CNF formula φ as follows:

1. for each vertex $w \in B$ create a variable x_w;
2. for each vertex $w \in N(u) \cap N(v)$ create a clause (x_w);
3. for each edge $w_1w_2 \in E(G[B])$ create the clauses $(x_{w_1} + x_{w_2})$ and $(\overline{x}_{w_1} + \overline{x}_{w_2})$;
4. for each vertex $w \in B$ with at least two neighbors in the same component T of $G[N(u) \setminus N[v]]$, create a clause $(\overline{x}_w)$;
5. For each component T of $G[N(u) \setminus N[v]]$, and for each pair of vertices w_1, w_2 in the neighborhood of T, create a clause $(\overline{x}_{w_1} + \overline{x}_{w_2})$;

At this point, it is easy to see that φ is satisfied if and only if $G[B]$ has a partition (B_1, B_2) as requested (variables equal to true correspond to the vertices of B_2). Since φ can be built in $O(n^2)$ time with respect to the size of $G[B]$ and 2SAT can be solved in linear time (ASPVALL; PLASS; TARJAN, 1982), a near-bipartition $(\mathcal{S}, \mathcal{F})$ of G can be found in $O(n^2)$ time (if any). □

Recall that Near-Bipartiteness can be seen as the problem of determining whether G admits an independent feedback vertex set $\mathcal{S}$ or an acyclic vertex cover $\mathcal{F}$. In contrast to the previous theorem, we show that the problems of finding a minimum independent feedback vertex set and a minimum acyclic vertex cover are both NP-hard on graphs with a dominating edge.

Theorem 3. Independent Feedback Vertex Set *is* NP*-hard when restricted to graphs having a dominating edge.*

Proof. In Positive Min-Ones-2SAT we are given a 2SAT formula φ having only positive literals and asked to decide whether there exists a satisfying assignment for φ with at most k variables set to true. Note that Positive Min-Ones-2SAT is equivalent to Minimum Vertex Cover, a well-known NP-complete problem.

Given an instance φ of Positive Min-Ones-2SAT, we can construct a graph G by applying the same construction as the Theorem 1 (disregarding negative literals). At this point, variables x_i set to *true* are equivalent to the vertices v_{x_i} assigned to $\mathcal{S}$. Therefore, φ has a satisfying truth assignment with at most k trues if and only if G is partitionable into a stable set $\mathcal{S}$ and a forest $\mathcal{F}$ such that $|\mathcal{S}| \leq k + 4$. □

Theorem 4. Acyclic Vertex Cover *is* NP*-hard when restricted to graphs having a dominating edge.*

Proof. First, we define a construction algorithm f that receives as input a CNF formula φ and outputs a graph having a dominating edge, similar to that presented in Theorem 1. The first six steps of this construction are the same as in Theorem 1. (to make it easier for the reader, we repeat their description here)

From φ we construct G as follows:

1. first consider $G = (\{u,v\},\{uv\})$;
2. add a cycle C of size 4 in G induced by $\{k_1,k_2,k_3,k_4\}$, and add edges from u for all vertices in C;
3. add a cycle $C' = l_1,m,l_2,n_1,n_2$;
4. add the edges ul_1, ul_2, vm, vn_1 and vn_2;
5. for each variable x_i of φ create vertices v_{x_i} and $v_{\overline{x}_i}$ and add edges $v_{x_i}v_{\overline{x}_i}$, uv_{x_i} and $uv_{\overline{x}_i}$;
6. for each clause C_j of φ create a vertex c_j in G and add the edge vc_j;

Now, in Step 7, we present the necessary change for our reduction.

7. add an edge $c_jv_{\overline{x}_i}$ if the clause C_j contains the literal x_i, and add an edge $c_jv_{x_i}$ if the clause C_j contains the literal $\overline{x}_i$; (Note that an edge between c_j and $v_{\overline{x}_i}$ is added when C_j contains the literal x_i)
8. Finally, for each vertex v_{x_i} representing a positive literal, we replace it with n copies $v^1_{x_1}, v^2_{x_1}, \ldots, v^n_{x_1}$, having the same neighbors as v_{x_1}, as can be seen in Figure 7 presented below.

By construction, each gadget g_{x_i} is formed by the n copies of positive literals $v^1_{x_i}, v^2_{x_i}, \ldots, v^n_{x_i}$, along with the negated literal $\overline{x}_i$.

The Figure 7 illustrates a graph, denoted as G, constructed from the formula $\varphi = (x_2+x_3)\cdot(x_1+x_3)\cdot(x_2+x_4)$. Additionally, it showcases a near-bipartition $(\mathcal{S},\mathcal{F})$ of G, which is derived from the satisfying assignment A = "$x_1 = false$", "$x_2 = true$", "$x_3 = true$", "$x_4 = false$" of φ. In the image, the white vertices represent the stable set, while the black vertices induce the forest.

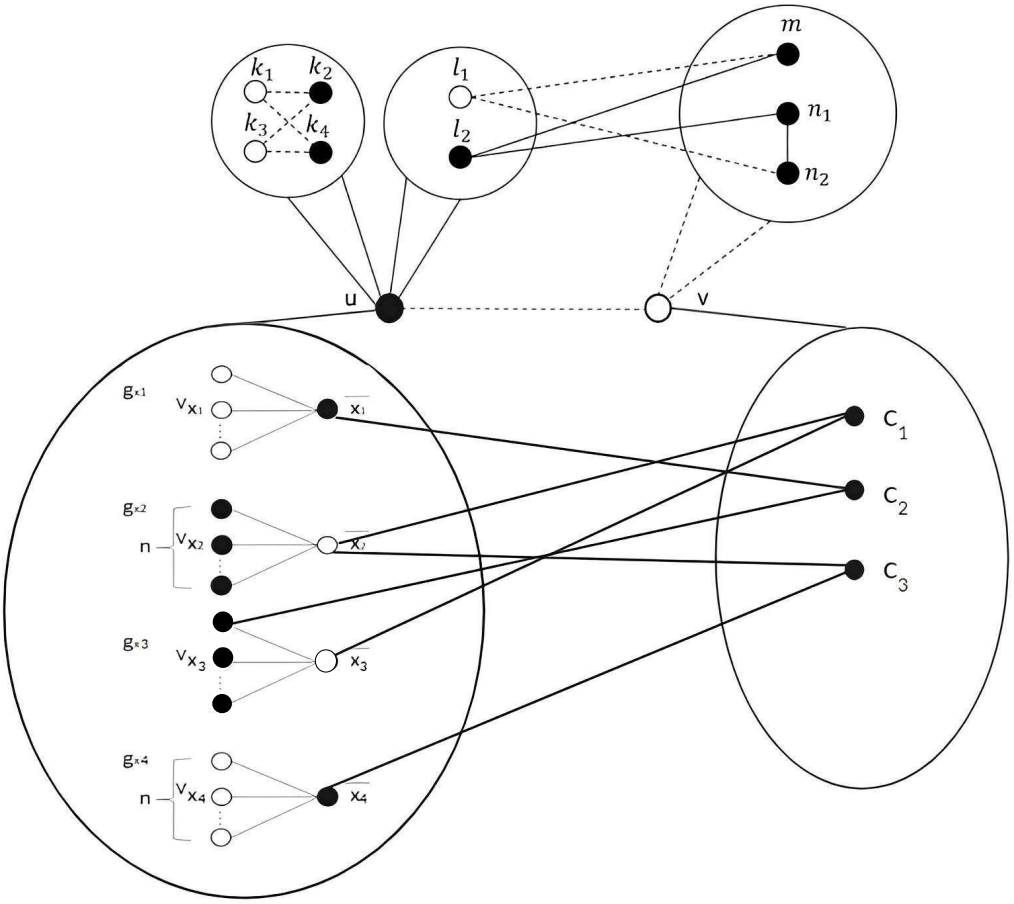

Figure 7: Example of graph and Near-Bipartition representation from the formula $(x_2 + x_3) \cdot (x_1 + x_3) \cdot (x_2 + x_4)$.

Now, given an instance φ with n variables and m clauses of Positive Min-Ones-2SAT, we denote by $G = f(\varphi)$ the graph obtained by applying the construction f from φ.

Suppose that φ has a satisfying assignment A with weight k. From A we construct a near-bipartition of G as follows: for each variable x_i of φ, if x_i equals *true* in A, then the vertices associated with the literal x_i are in $\mathcal{F}$, and if x_i equals *false* in A, then the vertex associated with the literal $\overline{x}_i$ is in $\mathcal{F}$. So, we can construct an near-bipartition $(\mathcal{S}, \mathcal{F})$ of G by setting

$$\begin{aligned}\mathcal{F} =& \big\{\{k_2, k_4, l_2, m, n_1, n_2, u\} \\ & \cup \{v^1_{x_i} \dots v^n_{x_i} : \text{“}x_i = true\text{”} \in A\} \\ & \cup \{v_{\overline{x}_i} : \text{“}x_i = false\text{”} \in A\} \\ & \cup \{c_1, c_2, \dots, c_m\}\big\},\end{aligned}$$

$$\mathcal{S} = V \setminus \mathcal{F}.$$

Note that $|\mathcal{F}| = m + k \cdot n + (n - k) + 7$, where m is the number of clause vertices,

$k \cdot n$ is the number of vertices that represent positive literals ("$x_i = true$" in A), $n - k$ is the number of vertices that represent negative literals ("$x_i = false$" in A), and 7 is the number of auxiliary vertices (k_2, k_4, l_2, m, n_1, n_2, u) in $\mathcal{F}$.

Now, let's analyze the graph induced by the vertices in $\mathcal{F}$. Every clause vertex is in $\mathcal{F}$. Also, if a clause vertex has degree two in the graph induced by $\mathcal{F}$, by construction, in the instance φ of Positive Min-Ones-2SAT, the corresponding clause has 2 false literals, which contradicts the fact that A is an assignment that satisfies φ. So, the clause vertices have degree at most 1, and do not belong to any cycle. At this point, it is easy to see that $\mathcal{F}$ induces a forest with $m + 7 + k \cdot n + (n - k)$ vertices, and $\mathcal{S} = V \setminus \mathcal{F}$ is a stable set.

Conversely, suppose that G has a near-partition $(\mathcal{S}, \mathcal{F})$ where $|\mathcal{F}| = m + 7 + k \cdot n + (n - k)$. First, let's argue that at most k gadgets g_{x_i} have vertices associated with true (x_i) in $\mathcal{F}$. By construction, each gadget g_{x_i} contains either one vertex representing $\overline{x}_i$ ($v_{\overline{x}_i}$) in $\mathcal{F}$ or n vertices representing x_i in $\mathcal{F}$ ($n \cdot v_{x_i}$). So let's suppose that $k + 1$ gadgets have $n \cdot (k + 1)$ vertices associated with true in $\mathcal{F}$. So this implies that $\mathcal{F}$ has at least $m + 7 + (k + 1) \cdot n$ vertices, which is bigger than the initially defined budget, $|\mathcal{F}| = m + 7 + k \cdot n + (n - k) \cong m + 7 + (k + 1) \cdot n - k$. So, at most k gadgets g_{x_i} have vertices associated with true in $\mathcal{F}$. With this, we will construct an assignment A to φ as follows:

- for each gadget g_{x_i} set $x_i = false$ if the vertex associated with $\overline{x}_i$ (i.e., $v_{\overline{x}_i}$), belongs to $\mathcal{F}$ and set $x_i = true$ otherwise.

Note that A has weight at most k. By the construction of G, each vertex associated with the clause has degree at most 1 in $\mathcal{F}$, given the property of the forest to be acyclic. So, for each clause at least one of its literals is *true*. Therefore, A satisfies φ with weight at most k. □

5 Results obtained in P_5-free graphs

In 2019, Bonamy, Dabrowski, Feghali, Johnson, and Paulusma (BONAMY et al., 2019) showed that Near -Bipartiteness and Independent Feedback Vertex Set can be solved in $O(n^{16})$ time.

In 1990, Bacsó and Tuza (BACSÓ; TUZA, 1990) showed that any connected P_5-free graph has a dominating clique or a dominating P_3. In 2016, Camby and Schaudt (CAMBY; SCHAUDT, 2016) generalized this result and showed that such a dominating set can be computed in polynomial time.

In this chapter, we will present, using the same approach as in Theorem 2, how to handle Near-bipartiteness Problem on graphs having a dominating clique or a dominating P_3. Our results imply a faster algorithm to solve Near-bipartiteness on P_5-free graphs with a time complexity of $O(n^4)$. Interestingly, we can observe that the same technique combined with Bacsó and Tuza's result is not very useful to get a more efficient algorithm for Independent Feedback Vertex Set on P_5-free graphs, due our Theorem 3 showing that this problem remains NP-complete on graphs having a dominating edge.

Theorem 5. *Given a graph G and a dominating triangle of G, one can determine in $O(n^2)$ time whether G is a near-bipartite graph.*

Proof. Let $\{u, v, z\} \in V(G)$ be a dominating set of G that induces a triangle. Suppose that G has a near-bipartition $(\mathcal{S},\mathcal{F})$. Without loss of generality, we can assume that G has no vertices of degree one, since these vertices can always be added to $\mathcal{F}$ without causing any contradiction. Thus, let's analyze the case where the vertices u, v of the dominating triangle belong to $\mathcal{F}$ and only the vertex z of the dominating triangle belongs to $\mathcal{S}$.

Since, $z \in S$, then $N(z) \subseteq \mathcal{F}$. Thus, $N(z)$ must induce a forest.

We can also observe that $N(u) \cap N(v) \cap N(z) = \emptyset$, because if there is at least one vertex w neighboring all the dominating vertices, this can't belong to either $\mathcal{F}$ or $\mathcal{S}$, since otherwise we will have cycles in $\mathcal{F}$ or edges in $\mathcal{S}$.

At this point, only the vertices belonging to $N(v) \cup N(u) \setminus N[z]$ are unclassified.

Let $B = N(v) \cup N(u) \setminus \{u, v, z\}$.

Previously seen that G has a near-bipartition $(\mathcal{S}, \mathcal{F})$, then we have that $G[B]$ must be bipartite. So, we have that the vertices of $G[B]$ can be partitioned into two sets (B_1, B_2) such that $B_1 \subseteq \mathcal{S}$ and $B_2 \subseteq \mathcal{F}$. Thus, we must find a bipartition of B that satisfies the following conditions:

– $N(u) \cap N(v) \subseteq B_1$, otherwise $\mathcal{F}$ will have cycles.

– $N(z) \cap (N(u) \cup N(v)) \subseteq B_2$, otherwise $\mathcal{S}$ will have edges.

– For each component T of $G[N(z) \backslash N[u] \cup N[v]]$, which is a tree, holds that:

  • For each $w \in B_2$, $|N_T(w)| \leq 1$ (otherwise $\{w\} \cup V(T)$ induces a cycle);

  • T has at most one neighbor in B_2 (otherwise $\mathcal{F}$ has cycles);

Note that any bipartition (B_1, B_2) satisfying the above restrictions is sufficient to form a near-bipartition such that $B_1 \cup \{z\} = \mathcal{S}$. Now, we can reduce the problem of finding such a bipartition of $G[B]$ to the 2SAT problem by constructing a 2-CNF formula φ, similarly as in Theorem 2. In order for the dissertation to be self-contained, we will present the resolution again below. So, the rules are:

1. for each vertex $w \in B$ create a variable x_w;
2. for each vertex $w \in N(u) \cap N(v)$ create the clause $(\overline{x}_w)$;
3. for each vertex $w \in N(z) \cap (N(u) \cup N(v))$ create the clause (x_w);
4. for each edge $w_1w_2 \in E(G[B])$ create the clauses $(x_{w_1} + x_{w_2})$ and $(\overline{x}_{w_1} + \overline{x}_{w_2})$;
5. for each vertex $w \in B$ with at least two neighbors in the same component T of $G[N(z) \backslash N[u] \cup N[v]]$, create a clause $(\overline{x}_w)$;
6. For each component T of $G[N(z) \backslash N[u] \cup N[v]]$, and for each pair of vertices w_1, w_2 in the neighborhood of T, create a clause $(\overline{x}_{w_1} + \overline{x}_{w_2})$;

At this point, it is easy to see that φ is satisfied if and only if $G[B]$ has a partition (B_1, B_2) as requested (variables equal to true correspond to the vertices of B_2). Since φ can be constructed in $O(n^2)$ time with respect to the size of $G[B]$ and 2SAT can be solved in linear time (ASPVALL; PLASS; TARJAN, 1982), a near-bipartition $(\mathcal{S}, \mathcal{F})$ of G can be found in $O(n^2)$ time (if any). □

Theorem 6. *Given a graph G and a dominating induced P_3 of G, one can determine in $O(n^4)$ time whether G is a near-bipartite graph.*

Proof. Let uvz be a induced P_3 of G such that $\{u,v,z\}$ is a dominating set in G. Suppose that G has a near-bipartition $(\mathcal{S}, \mathcal{F})$. Without loss of generality, we can assume that G has no vertices of degree one, since these vertices could always be added to the partition of $\mathcal{F}$ without generating any contradiction. At this point, we may analyze four cases:

Case 1. Suppose that the vertices $\{u, v\} \in \mathcal{F}$ and the vertex $\{z\} \in \mathcal{S}$.

In this case, the proof is similar to that of Theorem 5. However, in order for the dissertation to be self-contained, we will present the resolution again below. Additionally,

it is important to highlight that this proof also applies to the scenario where $v, z \in \mathcal{F}$ and $u \in \mathcal{S}$.

Since $z \in \mathcal{S}$, then $N(z) \subseteq \mathcal{F}$. Thus, $N(z)$ must induce a forest.

We can also observe that $N(u) \cap N(v) \cap N(z) = \emptyset$, because if there is at least one vertex w neighboring all the dominating vertices, this can't belong to either $\mathcal{F}$ or $\mathcal{S}$, since otherwise we will have cycles in $\mathcal{F}$ or edges in $\mathcal{S}$.

At this point, only the vertices belonging to $N(v) \cup N(u) \setminus N[z]$ are unclassified.

Let B = $N(v) \cup N(u) \setminus \{u, v, z\}$.

Previously seen that G has a near-bipartition $(\mathcal{S}, \mathcal{F})$, then we have that $G[B]$ must be bipartite. So, we have that the vertices of $G[B]$ can be partitioned into two sets (B_1, B_2) such that $B_1 \subseteq \mathcal{S}$ and $B_2 \subseteq \mathcal{F}$. Thus, we must find a bipartition of B that satisfies the following conditions:

- $N(u) \cap N(v) \subseteq B_1$, otherwise $\mathcal{F}$ will have the presence of cycles.
- $N(z) \cap (N(u) \cup N(v)) \subseteq B_2$, otherwise $\mathcal{S}$ will have edges.
- For each component T of $G[N(z) \setminus N[u] \cup N[v]]$, which is a tree, holds that:
  - For each $w \in B_2$, $|N_T(w)| \leq 1$ (otherwise {w} $\cup V(T)$ induces a cycle);
  - T has at most one neighbor in B_2 (otherwise $\mathcal{F}$ has cycles);

Note that any bipartition (B_1, B_2) satisfying the above restrictions is sufficient to form a near-bipartition such that $B_1 \cup \{z\} = \mathcal{S}$. Now, we can reduce the problem of finding such a bipartition of $G[B]$ to the 2SAT problem by constructing a 2-CNF formula φ respecting the following rules:

1. for each vertex $w \in B$ create a variable x_w;
2. for each vertex $w \in N(u) \cap N(v)$ create the clause $(\overline{x}_w)$;
3. for each vertex $w \in N(z) \cap (N(u) \cup N(v))$ create the clause (x_w);
4. for each edge $w_1w_2 \in E(G[B])$ create the clauses $(x_{w_1} + x_{w_2})$ and $(\overline{x}_{w_1} + \overline{x}_{w_2})$;
5. for each vertex $w \in B$ with at least two neighbors in the same component T of $G[N(z) \setminus N[u] \cup N[v]]$, create a clause $(\overline{x}_w)$;
6. For each component T of $G[N(z) \setminus N[u] \cup N[v]]$, and for each pair of vertices w_1, w_2 in the neighborhood of T, create a clause $(\overline{x}_{w_1} + \overline{x}_{w_2})$;

At this point, it is easy to see that φ is satisfied if and only if $G[B]$ has a partition (B_1, B_2) as requested (variables equal to true correspond to the vertices of B_2). Since φ can be constructed in $O(n^2)$ time with respect to the size of $G[B]$ and 2SAT can be solved in linear time (ASPVALL; PLASS; TARJAN, 1982), a near-bipartition $(\mathcal{S}, \mathcal{F})$ of G can be found in $O(n^2)$ time (if any).

Case 2: Suppose that only the vertex $v \in \mathcal{F}$ and the vertices $\{u, z\} \in \mathcal{S}$.

In this case, $N(u) \cup N(z) \subseteq \mathcal{F}$, otherwise $\mathcal{S}$ has edges. Thus, $N(u) \cup N(z)$ must induce a forest.

Also, $G[N(u) \cup N(z)]$ must contain no path between two vertices of $N(v) \cap (N(v) \cup N(z))$, otherwise $\mathcal{F}$ has cycles.

Furthermore, $N(v) \setminus (N(u) \cup N(z))$ must induce a bipartite graph. Also, for a vertex w belonging to $N(v) \setminus (N(u) \cup N(z))$ if it has a neighbor $p \neq v$ that reaches v in $G[N(u) \cup N(z)]$ then p must be in $\mathcal{S}$.

At this point, similarly as in Theorem 2, we can use a 2SAT formula to decide which unclassified vertices of $N(v) \setminus (N(u) \cup N(z))$ must be in $\mathcal{F}$ and $\mathcal{S}$. To ensure the self-contained of the dissertation, we will also provide this part of the mathematical proof for this particular case.

Let $B = N(v) \setminus \{u,v,z\}$.

As seen previously, $G[B]$ must be bipartite. So, its vertices can be partitioned into two sets (B_1, B_2) such that $B_1 \subseteq \mathcal{S}$ and $B_2 \subseteq \mathcal{F}$. Thus, we must find a bipartition of B that satisfies the following conditions:

- $N(v) \cap (N(u) \cup N(z)) \subseteq B_2$, otherwise $\mathcal{S}$ will have edges.
- for each component T of $G[(N[u] \cup N[v]) \setminus N(z)]$ (which is a tree) holds that:
  - For each $w \in B_2$, $|N_T(w)| \leq 1$ (otherwise $\{w\} \cup V(T)$ induces a cycle);
  - T has at most one neighbor in B_2 if $T \not\supseteq \{v\}$ (otherwise $\mathcal{F}$ has cycles);
  - T has no neighbor in B_2 if $T \supseteq \{v\}$ (otherwise $\mathcal{F}$ has cycles);

Note that any bipartition (B_1, B_2) satisfying the above restrictions is sufficient to form a near-bipartition such that $B_1 \cup \{u, z\} = \mathcal{S}$. Now, we can reduce the problem of finding such a bipartition of $G[B]$ to the 2SAT problem by building a 2-CNF formula φ as follows:

1. for each vertex $w \in B$ create a variable x_w;
2. for each vertex $w \in N(v) \cap (N(u) \cup N(z))$ create a clause (x_w);
3. for each edge $w_1w_2 \in E(G[B])$ create the clauses $(x_{w_1} + x_{w_2})$ and $(\overline{x}_{w_1} + \overline{x}_{w_2})$;
4. for each vertex $w \in B$ with at least two neighbors in the same component T of $G[(N[u] \cup N[v]) \setminus N(z)]$ that do not contains the vertex v, create a clause $(\overline{x}_w)$;
5. For each component T of $G[(N[u] \cup N[v]) \setminus N(z)]$, and for each pair of vertices w_1, w_2 in the neighborhood of T, create a clause $(\overline{x}_{w_1} + \overline{x}_{w_2})$;
6. For each component T of $G[(N[u] \cup N[v]) \setminus N(z)]$ that contains the vertex v, for each vertex $w_1 \in V(T)$, create a clause $(\overline{x}_{w_1})$;

At this point, it is easy to see that φ is satisfied if and only if $G[B]$ has a partition (B_1, B_2) as requested (variables equal to true correspond to the vertices of B_2). Since φ

can be built in $O(n^2)$ time with respect to the size of $G[B]$ and 2SAT can be solved in linear time (ASPVALL; PLASS; TARJAN, 1982), a near-bipartition $(\mathcal{S}, \mathcal{F})$ of G can be found in $O(n^2)$ time (if any).

Case 3: Suppose that $\{u, v, z\} \subseteq \mathcal{F}$.

In this case, any vertex with at least two neighbors in $\{u, v, z\}$ must be in $\mathcal{S}$.

Note that

$$(N(v) \setminus (N(u) \cup N(z))) \cup (N(z) \setminus (N(u) \cup N(v))) \cup (N(u) \setminus (N(v) \cup N(z)))$$

must induce a bipartite graph B.

Furthermore, for a remaining vertex w belonging to B:

If it has a neighbor that must be in $\mathcal{S}$ then it must belong to $\mathcal{F}$; On the other hand, if w has a neighbor $p \notin \{u,v,z\}$ that must be in $\mathcal{F}$, then w must belong to $\mathcal{S}$, otherwise, there is a cycle in $\mathcal{F}$.

Thus, by checking if $(N(v) \cap N(z)) \cup (N(u) \cap N(v)) \cup (N(u) \cap N(z))$ is a stable set and then successively applying the classification process previously described (a 2-coloring into $\mathcal{S}$ and $\mathcal{F}$) from $(N(v) \cap N(z)) \cup (N(u) \cap N(v)) \cup (N(u) \cap N(z))$, in linear time, we can either conclude that such a near-bipartition of G with $\{v, u, z\} \in \mathcal{F}$ does not exist, or we construct a partition (S', F', U) of $V(G)$ such that S' is stable, $F' \supseteq \{v, u, z\}$ induces a forest, and U is the set of unclassified vertices. Note that, by construction, no vertex in U has neighbors in $S' \cup F' \setminus \{v, u, z\}$.

Since any pair of adjacent vertices together with u, v and z induces a cycle, G has a near-bipartition $(\mathcal{S}, \mathcal{F})$ with $\{v, u, z\} \subseteq \mathcal{F}$ if and only if $G[U]$ has an independent vertex cover, which is equivalent to U inducing a bipartite graph (see Lemma 2), which can also be checked in linear time.

Next, we discuss the most intriguing case where the vertices of the dominating set that must be in the forest do not induce a connected component.

Case 4: Suppose that $\{u, z\} \subseteq \mathcal{F}$ *and* $v \in \mathcal{S}$. Recall that $N(v) \subseteq \mathcal{F}$.

In the following, we consider two cases to be analyzed, either u and z are in the same tree of $\mathcal{F}$ or they are in distinct trees.

Case A - u and z are in the same tree of $\mathcal{F}$. Thus, there is a path $\mathcal{P}$ between them in $\mathcal{F}$. Such a path contains exactly one neighbor of u and exactly one neighbor of z, otherwise $V(\mathcal{P})$ induces a cycle in $\mathcal{F}$.

Therefore, we enumerate each pair a_u,a_z ($a_u = a_z$ is allowed) such that $a_u \neq v$ and it is neighbor of u, $a_z \neq v$ and it is neighbor of z, and $\{a_u,a_z\} \cup N(v)$ induces a forest having a tree containing u and z.

Observe that we have $O(n^2)$ pairs a_u,a_z, and in $O(m)$ time we can check if $\{a_u,a_z\} \cup N(v)$ induces a forest having a tree containing u and z.

Now, for each enumerated pair a_u,a_z, we check if there is a near-bipartition having

$\{a_u,a_z\} \cup N(v) \subseteq \mathcal{F}$ as follows.

- $(N(u) \cup N(z)) \setminus (\{a_u,a_z\} \cup N[v])$ must induce a bipartite graph B, otherwise there is no near-bipartition having $\{a_u,a_z\} \cup N(v) \subseteq \mathcal{F}$.

  Thus, it is enough to decide if B has a bipartition $V(B) = B_1 \cup B_2$ satisfying the following:

  Let $\mathcal{T}$ be the tree of $G[\{a_u,a_z\} \cup N(v)]$ containing u and z.

  - Each vertex of B having at least two neighbors in a tree of $G[\{a_u,a_z\} \cup N(v)]$ must be in B_1.
  - Each tree of $G[\{a_u,a_z\} \cup N(v)]$ distinct from $\mathcal{T}$ has at most one neighbor in B_2.

If B has such a bipartition then $B_1 \cup \{v\} = \mathcal{S}$ and $B_2 \cup \{a_u,a_z\} \cup N(v) = \mathcal{F}$ form a near-bipartition of G. Again, such a bipartition, if any, can be found using a 2SAT formula respectiing the following rules:

1. for each vertex $w \in B$ create a variable x_w;
2. for each vertex $w \in B$ with at least two neighbors in a tree of $G[\{a_u,a_z\} \cup N(v)]$, create a clause $(\overline{x}_w)$;
4. for each edge $w_1w_2 \in E(G[B])$ create the clauses $(x_{w_1} + x_{w_2})$ and $(\overline{x}_{w_1} + \overline{x}_{w_2})$;
3. For each tree of $G[\{a_u,a_z\} \cup N(v)]$ distinct from $\mathcal{T}$, and for each pair of vertices w_1, w_2 in the neighborhood of T, create a clause $(\overline{x}_{w_1} + \overline{x}_{w_2})$;

The overall running time for this case is $O(n^4)$, because we consider $O(n^2)$ pairs and for each one the described procedure can be performed in $O(n^2)$ time.

Case B - u and z are disconnected in $\mathcal{F}$.

In this situation there is no path between vertices u and z in $\mathcal{F}$.

Therefore, $N(u) \cap N(z) \subseteq \mathcal{S}$, $G[N(u) \cup N(z)]$ is bipartite, and the vertices of $(N(u) \cup N(z)) \cap V(\mathcal{F})$ is a stable set, otherwise the vertices u and z are connected in $\mathcal{F}$. Besides that, $N(v) \subseteq \mathcal{F}$.

At this point, analogously to the previous cases, we can use 2SAT to find an appropriated classification of the vertices of $N(z) \cup N(u)$ into $\mathcal{S}$ and $\mathcal{F}$ (if any).

Let $B = N(u) \cup N(z) \setminus \{u,v,z\}$.

Previously seen that G has a near-bipartition $(\mathcal{S}, \mathcal{F})$, then we have that $G[B]$ must be bipartite. So, we have that the vertices of $G[B]$ can be partitioned into two sets (B_1, B_2) such that $B_1 \subseteq \mathcal{S}$ and $B_2 \subseteq \mathcal{F}$. Thus, we must find a bipartition of B that satisfies the following conditions:

- $N(v) \cap (N(u) \cup N(z)) \subseteq B_2$, otherwise $\mathcal{S}$ will have edges.

– $N(u) \cap N(z) \subseteq B_1$, otherwise there is a path between vertices u and z in $\mathcal{F}$.

– for each component T of $G[N[v] \setminus (N[u] \cup N[z])]$ (which is a tree) holds that:

  • For each $w \in B_2$, $|N_T(w)| \leq 1$ (otherwise $\{w\} \cup V(T)$ induces a cycle);

  • T has at most one neighbor in B_2 (otherwise $\mathcal{F}$ has cycles);

Note that any bipartition (B_1, B_2) satisfying the above restrictions is sufficient to form a near-bipartition such that $B_1 \cup \{v\} = \mathcal{S}$. Now, we can reduce the problem of finding such a bipartition of $G[B]$ to the 2SAT problem by building a 2-CNF formula φ as follows:

1. for each vertex $w \in B$ create a variable x_w;
2. for each vertex $w \in N(v) \cap (N(u) \cup N(z))$ create a clause (x_w);
3. for each vertex $w \in N(u) \cap N(z)$ create a clause $(\overline{x_w})$;
4. for each edge $w_1w_2 \in E(G[B])$ create the clauses $(x_{w_1} + x_{w_2})$ and $(\overline{x}_{w_1} + \overline{x}_{w_2})$;
5. for each vertex $w \in B$ with at least two neighbors in the same component T of $G[N[v] \setminus (N[u] \cup N[z])]$, create a clause $(\overline{x}_w)$;
6. For each component T of $G[N[v] \setminus (N[u] \cup N[z])]$, and for each pair of vertices w_1, w_2 in the neighborhood of T, create a clause $(\overline{x}_{w_1} + \overline{x}_{w_2})$;

At this point, it is easy to see that φ is satisfied if and only if $G[B]$ has a partition (B_1, B_2) as requested (variables equal to true correspond to the vertices of B_2). Since φ can be built in $O(n^2)$ time with respect to the size of $G[B]$ and 2SAT can be solved in linear time (ASPVALL; PLASS; TARJAN, 1982), a near-bipartition $(\mathcal{S}, \mathcal{F})$ of G can be found in $O(n^2)$ time (if any).

Since all the cases can be performed in $O(n^4)$ time, we conclude the proof. □

Next, we improve the Bonamy, Dabrowski, Feghali, Johnson, and Paulusma's result (BONAMY et al., 2019) concerning Near-Bipartiteness on P_5-free graphs.

Corollary 1. Near-bipartiteness *on P_5-free graphs can be solved in $O(n^2 \cdot m)$ time.*

Proof. Near-bipartite graphs are K_4-free and K_4's can be found in $O(m^2)$ time. Also, near-bipartite P_5-free graphs have either a dominating triangle or a dominating P_3 due to Bacsó and Tuza's result (BACSÓ; TUZA, 1990). Hence, it is enough to apply Theorem 5 and Theorem 6. In addition, if G is P_5-free then Case 4A of Theorem 6 can be performed in $O(n^2 \cdot m)$ time, since either $a_u = a_z$ or a_ua_z is an edge of G. □

In the light of previous demonstrations, the reader may be realizing that we are able to extend our approach to deal with Near-Bipartiteness parameterized the domination number. Actually, we can proceed as follows.

Theorem 7. *Given a graph G and a dominating set D of G with size k, one can determine whether G is near-bipartite in $O(2^k \cdot n^{2k})$ time.*

Proof. Given G and D we can "guess" the vertices of D in $\mathcal{S}$ and in $\mathcal{F}$ in $O(2^k)$ time. For the vertices of $D \cap V(\mathcal{F})$ there are at most $2k-2$ neighbors used to connect some of them in the forest $\mathcal{F}$ (at most one pair a_u,a_z for a pair u,$z \in D$). We can "guess" such a vertices in $O(n^{2k-2})$ time. Then we can procede in $O(n^2)$ time using 2SAT as in the previous results. □

5.1 Summary of Cases

Below are the results obtained in the study of the 6 cases analyzed in this work in chapter 4, *Results obtained in graphs having a dominating edge*, and chapter 5, *Results obtained in P_5-free graphs.*

Table 1: Time complexity obtained of cases analyzed in this study

Problem Analyzed	Time Complexity
Partitioning problem in $\mathcal{S},\mathcal{T}$ in graphs with dominating edge (Connected Near-Bipartiteness)	NP-Complete
Near-Bipartiteness problem in graphs with dominating edge	$O(n^2)$
Near-Bipartiteness problem for graphs with dominating edge being $\lvert\mathcal{S}\rvert$ minimum (Independent Feedback Vertex Set)	NP-hard
Near-Bipartiteness problem for graphs with dominating edge being $\lvert\mathcal{F}\rvert$ minimum (Acyclic Vertex Cover)	NP-hard
Near-Bipartiteness problem for graphs with dominating K_3	$O(n^2)$
Near-Bipartiteness problem for graphs with dominating P_3	$O(n^4)$

In addition, we will also present in Figure 8 a flowchart that visually illustrates the verification process to determine whether a given graph G, free of P_5, admits a near-bipartition. The primary objective of this flowchart is to facilitate the comprehension of this process, making it easier to follow the steps involved in this analysis.

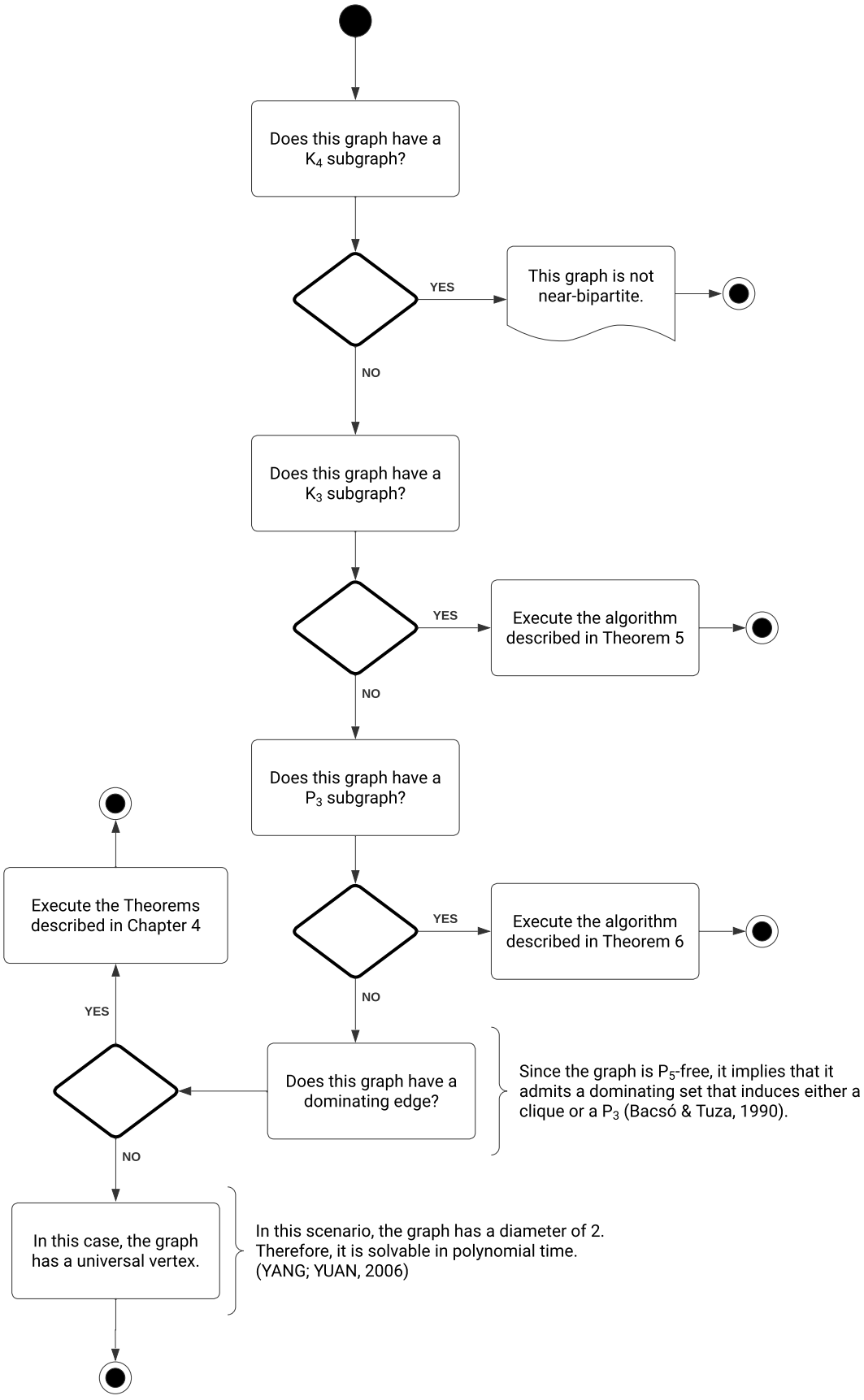

Figure 8: Flowchart: Verification of whether a P_5-free graph admits a near-bipartition.

6 Conclusion

In this dissertation, we focused on the study of the Near-Bipartition problem and its variants in graphs with dominating edges. Near-Bipartition is a fundamental problem in graph theory, which involves determining if a graph can be partitioned into two sets: a stable set and a forest.

The primary focus of this work was to analyze the Near-Bipartition in specific instances of graphs that have a dominating edge, as well as in the context of P_5-free graphs.

Regarding graphs that have a dominating edge, we presented a polynomial-time algorithm for Near-Bipartiteness and proved that Connected Near-Bipartiteness, the variant where the forest must be connected, is NP-complete. Furthermore, we showed that Independent Feedback Vertex Set, the problem of finding a near-bipartition $(\mathcal{S}, \mathcal{F})$ minimizing $|\mathcal{S}|$, and Acyclic Vertex Cover, the problem of finding a near-bipartition $(\mathcal{S}, \mathcal{F})$ minimizing $|\mathcal{F}|$, are both NP-hard when restricted to this class of graphs.

Extending our polynomial-time approach to deal with Near-Bipartiteness in graphs that have bounded dominating sets, we obtain an $O(n^2 \cdot m)$-time algorithm to solve it on P_5-free graphs. Additionally, it is mentioned in (BACSÓ; TUZA, 1990) that every connected P_5-free graph admits a dominating set that induces either a clique or a P_3. Therefore, for a connected graph without P_5, the goal is to determine if it admits Near-Bipartiteness, and this can be achieved by following these steps:

1. Check the existence of a dominating K_4 in $O(n^4)$ time.
2. Check the existence of a dominating P_3 or K_3 in $O(n^3)$ time.
3. Execute the algorithm described in Theorem 5 or Theorem 6, which was presented in Chapter 5 of this work, in $O(m.n^2) \cong O(n^4)$ time.

Thus, the study aims to demonstrate that the Near-Bipartiteness problem can be solved in $O(n^4)$ time for a graph without P_5, improving upon the existing result in the literature of $O(n^{16})$ (BONAMY et al., 2019).

Regarding the topics covered, throughout the first four chapters, we provided a summary of the subjects and main concepts that we studied and utilized for the development of this dissertation, with the objective of presenting the concepts in an understandable manner for the reader. Finally, in Chapter 4 and Chapter 5, we presented all the mathematical proofs in graphs with a dominating edge and P_5-free graphs, respectively.

Given the above, the findings presented in this work aim to contribute to a deeper understanding of the NEAR-BIPARTITION problem and its variants in graphs with dominating edges and those that are P_5-free.

Next, we will list some open problems in the context of this Master's thesis work.

6.1 Future works

This research opens avenues for further exploration in the field of graph theory and can potentially lead to the development of more efficient algorithms.

As an extension to this work, we propose:

- Continue the study of NEAR-BIPARTITENESS problem in graphs with dominating edges and P_5-free graphs, however, by parameterizing the problem, establishing the dominating set as the parameter. Our goal is to explore the possibility of obtaining an Fixed-Parameter Tractable (FPT) algorithm based on Theorem 7, which provides an XP algorithm. In other words, we aim to determine if it is possible to efficiently verify, given a graph G and a dominating set D of G with size k, whether G is near-bipartite, without resorting to a brute force approach ($O(n^{2k})$ runtime complexity).

REFERÊNCIAS

ACHLIOPTAS, Demetrios. The complexity of G-free colourability. **Discrete Mathematics**, v. 165-166, p. 21–30, 1997. ISSN 0012-365X.

AGRAWAL, Akanksha et al. Improved algorithms and combinatorial bounds for independent feedback vertex set. In: SCHLOSS DAGSTUHL-LEIBNIZ-ZENTRUM FUER INFORMATIK. 11TH International Symposium on Parameterized and Exact Computation (IPEC 2016). [S.l.: s.n.], 2017.

ASPVALL, Bengt; PLASS, Michael F.; TARJAN, Robert Endre. A linear-time algorithm for testing the truth of certain quantified Boolean formulas. English (US). **Information Processing Letters**, Elsevier, v. 14, n. 4, 1982. ISSN 0020-0190. DOI: 10.1016/0020-0190(82)90036-9.

BACSÓ, Gabor; TUZA, Zsolt. Dominating cliques in P_5-free graphs. **Periodica Mathematica Hungarica**, Springer, v. 21, n. 4, p. 303–308, 1990.

BANG-JENSEN, Jørgen; BESSY, Stéphane. Degree-constrained 2-partitions of graphs. **Theoretical Computer Science**, v. 776, p. 64–74, 2019. ISSN 0304-3975.

BONAMY, Marthe et al. Independent Feedback Vertex Set for P_5-Free Graphs. **Algorithmica**, Springer, v. 81, n. 4, p. 1342–1369, 2019.

BONAMY, Marthe et al. Independent feedback vertex sets for graphs of bounded diameter. **Information Processing Letters**, v. 131, p. 26–32, July 2018. DOI: 10.1016/j.ipl.2017.11.004.

BONAMY, Marthe et al. Recognizing graphs close to bipartite graphs. In: SCHLOSS DAGSTUHL-LEIBNIZ-ZENTRUM FUER INFORMATIK. 42ND International Symposium on Mathematical Foundations of Computer Science (MFCS 2017). [S.l.: s.n.], 2017.

BONDY, John Adrian; MURTY, Uppaluri Siva Ramachandra, et al. **Graph theory with applications**. [S.l.]: Macmillan London, 1976. v. 290.

BORODIN, O.V.; KOSTOCHKA, A.; YANCEY, M. On 1-improper 2-coloring of sparse graphs. **Discrete Math.**, v. 313, n. 22, p. 2638–2649, 2013.

BRANDSTÄDT, Andreas; BRITO, Synara, et al. Cycle transversals in perfect graphs and cographs. **Theoretical Computer Science**, Elsevier, v. 469, p. 15–23, 2013.

BRANDSTÄDT, Andreas; LE, Van Bang; SZYMCZAK, Thomas. The complexity of some problems related to Graph 3-colorability. **Discrete Applied Mathematics**, v. 89, n. 1, p. 59–73, 1998. ISSN 0166-218X.

CAMBY, Eglantine; SCHAUDT, Oliver. A new characterization of P_k-free graphs. **Algorithmica**, Springer, v. 75, n. 1, p. 205–217, 2016.

CORMEN, Thomas H. et al. **Introduction to Algorithms**. 3rd. Cambridge, MA: MIT Press, 2009. ISBN 978-0262033848.

COWEN, Lenore; GODDARD, Wayne; JESURUM, C. Esther. Defective coloring revisited. **J. Graph Theory**, v. 24, n. 3, p. 205–219, 1997.

DASGUPTA, Sanjoy; PAPADIMITRIOU, Christos H.; VAZIRANI, Umesh. **Algorithms**. 1st. Boston, MA: McGraw-Hill Education, 2006. ISBN 978-0073523408.

DROSS, François; MONTASSIER, Mickael; PINLOU, Alexandre. Partitioning a triangle-free planar graph into a forest and a forest of bounded degree. **European Journal of Combinatorics**, Elsevier, v. 66, p. 81–94, 2017.

GAREY, Michael R; JOHNSON, David S. **Computers and intractability: A Guide to the Theory of NP-completeness**. [S.l.]: freeman San Francisco, 1979. v. 174.

GOLOVACH, Petr; HEGGERNES, Pinar. Choosability of P5-Free Graphs. In: v. 5734, p. 382–391. ISBN 978-3-642-03815-0. DOI: 10.1007/978-3-642-03816-7_33.

GRÖTSCHEL, Martin; LOVÁSZ, László; SCHRIJVER, Alexander. Polynomial algorithms for perfect graphs. **Ann. Discrete Math**, v. 21, p. 325–356, 1984.

HOANG, Chinh et al. Deciding k-Colorability of P5-Free Graphs in Polynomial Time. **Algorithmica**, v. 57, p. 74–81, May 2010. DOI: 10.1007/s00453-008-9197-8.

KARP, Richard M. Reducibility among combinatorial problems. In: COMPLEXITY of computer computations. [S.l.]: Springer, 1972. P. 85–103.

LI, Shaohua; PILIPCZUK, Marcin. An improved FPT algorithm for independent feedback vertex set. **Theory of Computing Systems**, Springer, v. 64, n. 8, p. 1317–1330, 2020.

LIMA, Carlos V.G.C. et al. Decycling with a matching. **Infor. Proc. Letters**, v. 124, p. 26–29, 2017.

LIMA, Carlos Vinicius et al. On the computational complexity of the bipartizing matching problem. **Annals of Operations Research**, Springer, 2021.

LOKSHTANOV, D.; VATSHELLE, Martin; VILLANGER, Yngve. Independent Set in P_5-Free Graphs in Polynomial Time. **Proceedings of SODA**, p. 570–581, Jan. 2014. DOI: 10.1137/1.9781611973402.43.

MISRA, Neeldhara; NARAYANASWAMY, N.S., et al. Solving min ones 2-sat as fast as vertex cover. **Theoretical Computer Science**, v. 506, p. 115–121, 2013. ISSN 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2013.07.019. Available from: <https://www.sciencedirect.com/science/article/pii/S0304397513005355>.

MISRA, Neeldhara; PHILIP, Geevarghese, et al. On parameterized independent feedback vertex set. **Theoretical Computer Science**, Elsevier, v. 461, p. 65–75, 2012.

PROTTI, Fábio; SOUZA, Uéverton S. Decycling a graph by the removal of a matching: new algorithmic and structural aspects in some classes of graphs. **Discrete Mathematics & Theoretical Computer Science**, v. 20, n. 2, 2018.

SCHAEFER, Thomas J. The complexity of satisfiability problems. In: PROCEEDINGS of the tenth annual ACM Symposium on Theory of Computing. [S.l.: s.n.], 1978. P. 216–226.

SZWARCFITER, J. L. **Grafos e algoritmos computacionais**. Rio de Janeiro: Campus, 1988.

YANG, Aifeng; YUAN, Jinjiang. Partition the vertices of a graph into one independent set and one acyclic set. **Discrete Mathematics**, v. 306, n. 12, p. 1207–1216, 2006. ISSN 0012-365X.