Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Projeto Orientado em Computação I

Complexidade computacional do problema de empacotamento aberto em classes de grafos

Aluno: Luís Felipe Ramos Ferreira

Orientador: Vinicius Fernandes dos Santos

Belo Horizonte - Minas Gerais
2024

Sumário

Resumo

Seja $G = (V, E)$ um grafo e v um vértice em G. A vizinhança aberta de v em G, denotada por $N(v)$, é definida como o conjunto de vértices vizinhos de v em G. Um conjunto de empacotamento aberto em um grafo G é um conjunto de vértices $S \subseteq V(G)$ tal que, para todo $u, v \in S$, $N(u) \cap N(v) = \emptyset$, isto é, para todo par de vértices em S, suas vizinhanças são disjuntas. O tamanho de um conjunto de empacotamento aberto máximo em G é denotado por $\rho_o(G)$. O problema de se encontrar um empacotamento aberto de tamanho k em G é NP-Completo [15]. Neste trabalho, aglomeramos informações do estado da arte sobre a complexidade computacional do problema de empacotamento aberto em diferentes classes de grafos, assim como apresentamos um algoritmo guloso para encontrar o conjunto de empacotamento aberto em uma árvore.

Palavras chave: Teoria dos Grafos. Classes de Grafos. Empacotamento Aberto. Complexidade Computacional.

Abstract

Let $G = (V, E)$ be a graph and v a vertex in G. The open neighborhood of v in G, denoted by $N(v)$, is defined as the set of neighboring vertices of v in G. An open packing set in a graph G is a set of vertices $S \subseteq V(G)$ such that, for every $u, v \in S$, $N(u) \cap N(v) = \emptyset$, that is, for every pair of vertices in S, their neighborhoods are disjoint. The size of a maximum open packing set in G is denoted by $\rho_o(G)$. The problem of finding an open packing of size k in G is NP-Complete [15]. In this work, we aggregate information from the state of the art on the computational complexity of the open packing problem in different classes of graphs, as well as present a greedy algorithm to find the open packing set in a tree.

Keywords: Graph Theory. Graph Classes. Open Packing. Computational Complexity.

Complexidade computacional do problema de empacotamento aberto em classes de grafos

Luís Felipe Ramos Ferreira[1]
Orientador: Vinicius Fernandes dos Santos[1]

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brasil

1. Introdução

O estudo da complexidade de problemas em grafos é uma das principais vertentes na área de computação teórica. Problemas como clique, conjunto independente e conjunto dominante são pesquisados e estudados frequentemente. Nesse contexto, um problema que há de ser destacado é o de empacotamento aberto em um grafo. Em alto nível, ele pode ser definido como encontrar um conjunto de vértices do grafo de modo que nenhum dos vértices pertencentes ao conjunto compartilhem vizinhos. Sua versão de otimização consiste em encontrar, em um grafo G, o tamanho do maior desses conjuntos, denotado por $\rho_o(G)$.

Por exemplo, considere o grafo G abaixo. Nota-se que o conjunto de vértices $S = \{1, 2, 3\}$ não pode ser um conjunto de empacotamento aberto, uma vez que ambos vértices 1 e 3 são vizinhos do vértice 2, o que é um absurdo dada a definição de empacotamento aberto. No entanto, nota-se que o conjunto $S = \{1, 2\}$ satisfaz a propriedade necessária, portanto se trata de um empacotamento aberto. Além disso, é um empacotamento aberto máximo, dado que não podemos encontrar nenhum de tamanho maior.

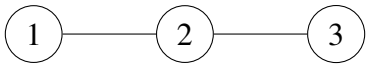

Figura 1. Grafo G

Sabe-se que o problema de decisão de se encontrar um conjunto de empacotamento aberto de tamanho k em um grafo G é NP-Completo [15] e, estando relacionado a conceitos fundamentais de teoria dos grafos como dominação e vizinhança, se trata de um problema de grande importância no cunho da computação teórica. Apesar disso, ele não possui a mesma quantidade de pesquisas e estudos como os outros problemas citados anteriormente.

1.1. Objetivos

Nesse contexto, esta monografia tem como objetivo a criação de um *survey* em volta do problema de empacotamento aberto, contendo informações sobre a complexidade de algoritmos aplicados a conhecidas classes de grafos, de modo que esta seja uma fonte de informação acerca do estado da arte em relação ao problema. Ademais, têm-se como objetivo estudar a complexidade computacional desse problema em classes de grafos que ainda não foram exploradas, como por exemplo as classes de grafos bloco e cacto.

Classe de Grafo	Complexidade	Referência
Grafo Geral	NP-Completo	[15]
Grafo Bipartido	NP-Completo	[15]
Grafo Cordal	NP-Completo	[15]
Grafo Split	NP-Completo	[28]
Grafo livre de $K_{1,3}$	NP-Completo	[32]
Grafo Ciclo	P	[15]
Grafo Caminho	P	[15]
Árvore	P	[4][16]
Grafo de Intervalo	P	[7]
Grafo com Largura Arbórea Limitada por k	P	[7]

Tabela 1. Tabela de complexidade para o problema de empacotamento aberto para determinadas classes de grafos

Atualmente, o problema de empacotamento aberto já foi estudado na literatura em algumas classes de grafos e diferentes resultados foram obtidos sobre a complexidade computacional do problema, que podem ser vistos na tabela abaixo.

As classes de grafos bloco e cacto são superclasses de árvores, como podemos ver no diagrama de Hasse de classes de grafos abaixo. É natural, portanto, questionar se o problema de empacotamento aberto é P ou NP-Completo para elas. A classe de grafos bloco, em particular, também é uma subclasse de grafos cordais.

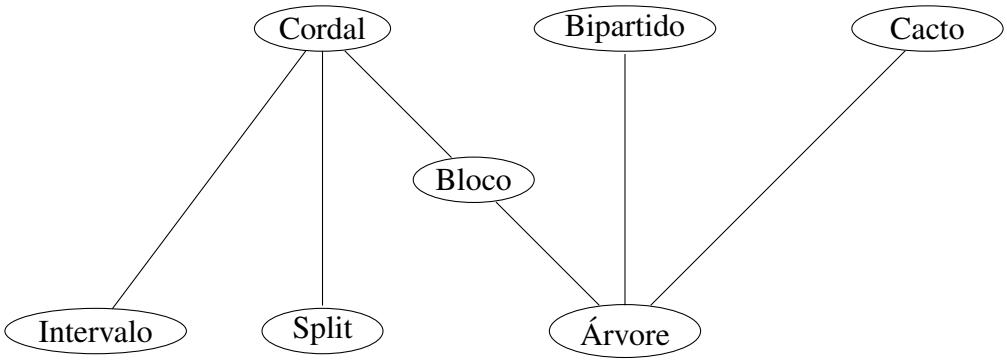

Figura 2. Diagrama de Classes de Grafos

1.2. Organização do trabalho

O trabalho aqui proposto está organizado da seguinte forma, a partir deste ponto. A seção 2 apresenta a fundamentação teórica necessária para o entendimento dos teoremas e algoritmos apresentados ao longo do *survey*. Conceitos essenciais sobre teoria de grafos são apresentados na seção 2.1 e as definições e caracterizações das principais classes de grafos citadas no trabalho na seção 2.2. Na seção 2.3 são apresentados alguns limites para o número de empacotamento aberto já conhecidos na literatura, os quais serão importantes para a definição de algoritmos nas seções seguintes.

Na seção 3 são apresentadas a complexidade computacional e os algoritmos conhecidos para o problema de empacotamento aberto nas principais classes de grafos es-

tudadas. Em particular, na seção 3.1 é apresentada a prova de que o problema é NP-Completo.

A seção 4 é utilizada para discutir possíveis trabalhos futuros na área. Em particular, são apresentadas ideias e abordagens para o estudo do problema de empacotamento aberto nas classes bloco e cacto.

A seção 5, para finalizar, apresenta a conclusão da monografia e os principais aprendizados e conceitos aprendido durante sua execução.

2. Referencial Teórico

Esta seção apresenta os principais conceitos relativos à teoria e classes de grafos necessários para o desenvolvimento e compreendimento do *survey*. Sua estrutura é baseada em [7] e rigorosamente fundamentada nas definições apresentadas em [8].

2.1. Teoria dos Grafos

Um grafo $G = (V, E)$ é uma estrutura composta por um conjunto de vértices V e um conjunto de arestas $E \subseteq \{(u, v) \mid u, v \in V\}$. Neste trabalho, quando nos referimos aos conjuntos V e E de um grafo G, utilizamos as notações $V(G)$ e $E(G)$. Chamamos de ordem do grafo o tamanho do conjunto V, isto é, o número de vértices do grafo. O grau de um vértice v, denominado $d(v)$, é o número de arestas do grafo incidentes ao vértice v. Chamamos de $\delta(G)$ o grau mínimo e $\Delta(G)$ o grau máximo de um grafo G. Um grafo G é simples se cada aresta de $E(G)$ liga dois vértices distintos e não existem duas arestas com as mesmas extremidades. Neste trabalho, todos os grafos estudados são simples.

Dizemos que o vértice u é adjacente ao vértice v em G se existe uma aresta $e \in E(G)$ cujas extremidades são u e v. Definimos a vizinhança aberta de um vértice u em G como o conjunto de vértices adjacentes a u, formalmente denotada como $N(u) = \{v \in V(G) \mid uv \in E(G)\}$. A vizinhança fechada de um vértice u é definida como $N[u] = N(u) \cup u$, isto é, a união de sua vizinhança aberta com ele mesmo. Similarmente, para um subconjunto $S \subset V(G)$, a vizinhança aberta de S é definida como $\bigcup_{v \in S} N(v)$ e sua vizinhança fechada como $\bigcup_{v \in S} N[v]$.

Um caminho $P = \{v_1, v_2, \ldots, v_{|P|}\}$ em um grafo G é uma sequência de vértices distintos tais que a aresta $v_i v_{i+1}$ existe em G para todo $i \leq |P| - 1$. O grafo G é dito conexo se existe pelo menos um caminho entre quaisquer dois pares de vértices de G. O tamanho de um caminho P é igual ao número de arestas nele, isto é, $|P| - 1$. A distância entre dois vértices u e v é o menor caminho entre eles, e o diâmetro de um grafo G é o tamanho do maior menor caminho entre qualquer par de vértices distintos de G.

Um ciclo $C = \{v_1, v_2, \ldots, v_{|C|}\}$ em um grafo simples G é uma sequência de 3 ou mais vértices distintos tais que a aresta $v_i v_{i+1}$ existe em G para todo $i \leq |P| - 1$, assim como também existe a aresta $v_1 v_{|C|}$. O tamanho de um ciclo é o número de vértices que ele possui, isto é, $|C|$. Uma corda em um ciclo C é uma aresta uv que pertence ao grafo G mas não pertence ao ciclo C, isto é, $uv \in E(G) \setminus E(C)$.

Seja G um grafo simples e $S \subseteq V(G)$. Ao remover de G todos os vértices que não pertencem a S, dizemos que $G[S]$ é um subgrafo de G induzido por S. Dizemos que um grafo G é livre de um subgrafo H se não existe um subgrafo induzido em G tal que $G \simeq H$, isto é, G é isomorfo à H.

Dizemos que dois grafos G e H são isomorfos se existe uma bijeção $f : V(G) \Rightarrow V(H)$ de modo que dois vértices u e v de G são adjacentes se e somente se $f(u)$ e $f(v)$ são adjacentes em H. Denotamos o isomorfismo entre dois grafos G e H como $G \simeq H$.

O complemento de um grafo G, denotado por $\overline{G}$ e é tal que $V(\overline{G}) = V(G)$ e $E(\overline{G}) = \{uv \mid uv \notin E(G)\}$, ou seja, existem em $\overline{G}$ exatamente as arestas que não existem em G. Nota-se também que $G = \overline{\overline{G}}$, isto é, o complemento do complemento de G é o próprio G.

Uma clique em um grafo G é um subconjunto S de $V(G)$ tal que para todo $u, v \in S$ a aresta uv existe em $E(G)$. Um conjunto independente em um grafo G, por sua vez, é um subconjunto I de $V(G)$ tal que para todo $u, v \in S$ a aresta uv não existe em $E(G)$. Podemos notar que o complemento de um conjunto independente é uma clique e vice-versa. O **número clique** de G, denotado por $\omega(G)$, é o tamanho da maior clique no grafo G.

Um conjunto de dominação é um subconjunto D de $V(G)$ tal que para todo $u \in V(G) \setminus D$, existe $v \in D$ tal que $uv \in E(G)$. O **número de dominação** de um grafo G, denotado por $\gamma(G)$, é o tamanho do menor conjunto de dominação em G. Um conjunto de dominação total, por sua vez, é um subconjunto D_t de $V(G)$ tal que para todo $u \in V(G)$, existe $v \in D_t$ tal que $uv \in E(G)$. O **número de dominação total** de um grafo G, denotado por $\gamma_t(G)$, é o tamanho do menor conjunto de dominação total de G.

Um conjunto de empacotamento é um subconjunto S de $V(G)$ tal que para todo $u, v \in S, N[u] \cap N[v] = \emptyset$, isto é, suas vizinhanças fechadas são disjuntas par a par. O **número de empacotamento** de G, denotado por $\rho(G)$, é o tamanho do maior conjunto de empacotamento de G. Um conjunto de empacotamento aberto é um subconjunto S_o de $V(G)$ tal que para todo $u, v \in S_o, N(u) \cap N(v) = \emptyset$, ou seja, suas vizinhanças abertas são disjuntas par a par. O **número de empacotamento aberto** de G, denotado por $\rho_o(G)$, é o tamanho do maior empacotamento aberto de G. O problema de se encontrar um conjunto de empacotamento aberto de tamanho k é o tema principal dessa monografia.

2.2. Classes de Grafos

Um **grafo caminho** com k vértices, denotado por P_k, é um grafo formado apenas por um caminho de tamanho k. A figura abaixo ilustra um grafo caminho de 4 vértices.

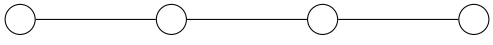

Figura 3. P_4

Um **grafo ciclo** com k vértices, denotado por C_k, é um grafo formado apenas por um ciclo de tamanho k. A figura a seguir ilustra um ciclo de 5 vértices.

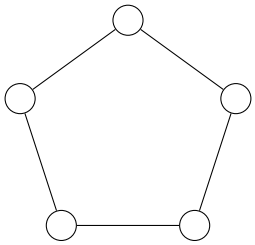

Figura 4. C_5

Uma grafo é **acíclico** se não possui um ciclo como subgrafo. Um grafo acíclico conexo é chamado de **árvore** e, por consequência, um grafo acíclico qualquer é chamado de **floresta**, ou seja, possui uma quantidade arbitrária de árvores. A figura abaixo ilustra uma árvore.

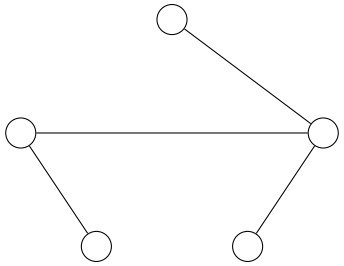

Figura 5. Exemplo de árvore

Um **grafo completo** com n vértices, denotado por K_n, é um grafo que possui todas as arestas possíveis, ou seja, para todo par de vértices $u, v \in V(K_n)$, têm-se que $uv \in E(K_n)$. A figura a seguir exemplifica um grafo completo de 5 vértices.

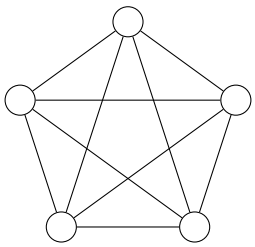

Figura 6. K_5

Um **grafo bipartido** G é um grafo cujo conjunto de vértices pode ser particionado em dois subconjuntos A e B, de modo que todas as arestas de $E(G)$ são entre um vértice de A e um vértice de B. Pode-se notar que os conjuntos A e B induzem conjuntos independentes em G. Um **grafo bipartido completo**, denotado por $K_{n,m}$ onde $n = |A|$ e $m = |B|$, é um grafo bipartido em que existem todas as arestas possíveis, ou seja, todos os vértices de A são conectados a todos os vértices de B. As figuras abaixo ilustram ambos os tipos de grafos citados.

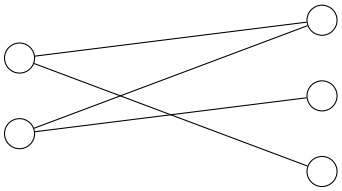

Figura 7. Exemplo de grafo bipartido

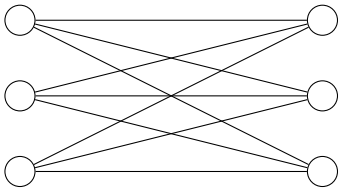

Figura 8. Exemplo de grafo bipartido completo

Um **grafo cordal** G é um grafo que não possui nenhum ciclo de tamanho 4 ou maior como subgrafo induzido. A classe de grafos cordais também possui outras caracterizações que são úteis para a análise e construção de algoritmos sobre ela. Dentre elas temos que um grafo é cordal se e somente se todo ciclo de tamanho 4 ou maior possui uma corda.

Além disso, temos a definição de que um grafo é cordal se e somente se ele possui uma **Ordem de Eliminação Perfeita (OEP)** [29]. Uma Ordem de Eliminação Perfeita é uma ordenação dos vértices de G de modo que, para todo vértice v, o conjunto de vértices vizinhos de v que se encontram após v na OEP induzem uma clique em G. A figura a seguir exemplifica um grafo cordal.

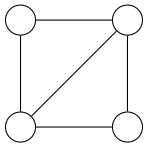

Figura 9. Exemplo de grafo cordal

Um **grafo split** é um grafo G cujo conjunto de vértices pode ser particionado em dois conjuntos K e I, de modo que K é uma clique e I um conjunto independente. A figura abaixo ilustra um grafo split.

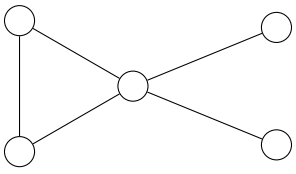

Figura 10. Exemplo de grafo split

Um **grafo de intervalo** G é o grafo de interseção de intervalos fechados no conjunto de números reais. Cada intervalo é um vértice no grafo, e se dois intervalo possuem interseção, então existe uma aresta em G entre seus vértices correspondentes. As figuras abaixo ilustram um conjunto de intervalos e o grafo de intervalo referente a ele.

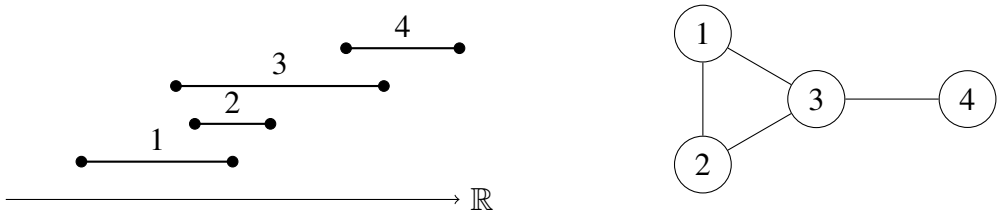

Figura 11. Exemplo de conjunto de intervalos e seu respectivo grafo de intervalo

Um **grafo bloco** é um grafo G em que todo componente bi-conexo (bloco) do grafo é uma clique. Um componente bi-conexo em um grafo é um subgrafo tal que, se qualquer vértice dele for removido, o subgrafo permanece conexo. A imagem abaixo exemplifica um grafo bloco.

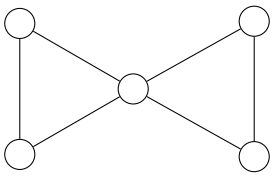

Figura 12. Exemplo de grafo bloco

Um **grafo cacto** é um grafo G em que toda aresta faz parte de no máximo um ciclo.

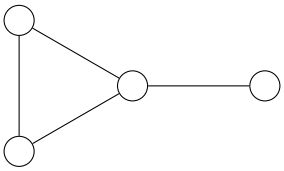

Figura 13. Exemplo de grafo cacto

2.3. Limites para o número de empacotamento aberto

As proposições e teoremas apresentadas aqui foram retiradas de [15].

Proposição 1. *Seja G um grafo de ordem $n \geq 2$ com sequência de graus $d_1, d_2, \ldots, d_n$. Então:*

$$\rho_o(G) \leq max\{k \mid d_1 + d_2 + \cdots + d_k \leq n\}$$

Proposição 2. *Seja G um grafo de ordem n com grau mínimo δ. Então:*

$$\rho_o(G) \leq n/\delta$$

3. Algoritmos e Complexidade Computacional

O conteúdo desta seção aglomera o estado da arte relativo aos conhecimentos atuais sobre a complexidade computacional do problema de empacotamento aberto nas principais classes de grafos.

Em particular, o seguinte problema de decisão está sendo considerado:

EMPACOTAMENTO ABERTO (OPK)
Instância de entrada: Um grafo G e um inteiro $k \leq |V(G)|$.
Pergunta: G possui um empacotamento aberto de tamanho igual a k?

3.1. Grafo Geral e Bipartido

Sabe-se que o problema é NP-Completo para grafos bipartidos e, portanto, para grafos gerais [15].

Para provar essa afirmação, os autores propuseram uma redução de um conhecido problema NP-Completo, denominado Cobertura Exata por Conjuntos de 3 Elementos, para o problema de empacotamento aberto.

COBERTURA EXATA POR CONJUNTOS DE 3 ELEMENTOS (X3C)
Instância de entrada: Um conjunto finito X tal que $|X| = 3q$ e uma coleção $\mathcal{C}$ de subconjuntos de 3 elementos de X.
Pergunta: $\mathcal{C}$ possui uma cobertura exata sobre X, isto é, uma subcoleção $\mathcal{C}' \subseteq \mathcal{C}$ tal que cada elemento de X ocorre em exatamente um membro de $\mathcal{C}'$?

Prova. Primeiramente, é evidente que OPK está em NP. A partir de uma possível solução S, pode-se checar se S é um empacotamento aberto em tempo polinomial. Mostraremos agora uma redução polinomial de X3C para OPK, o que implica em OPK ser um problema NP-Difícil.

Sejam $X = \{x_1, \ldots, x_{3q}\}$ e $\mathcal{C} = \{C_1, \ldots, C_m\}$ uma instância de X3C. Construiremos, a partir dela, uma instância G de OPK de modo que X3C contêm uma cobertura exata por conjuntos de 3 elementos se e somente se OPK possuir um empacotamento aberto de tamanho $k = m + 7q$.

O grafo G será construído da seguinte maneira:

- Para cada elemento $x_i \in X$ criamos o grafo H_i que consiste no grafo caminho formado pelos vértices x_i, y_i, w_i, z_i.
- Para cada conjunto $C_j \in \mathcal{C}$, criamos o grafo F_j, que consiste no grafo caminho formado pelos vértices c_j, d_j.
- Adicionamos as arestas $x_i c_j$ para todo $x_i \in C_j$.

Sua construção pode claramente ser feita em tempo polinomial. Considere, então, os conjuntos $W = \{w_1, w_2, \ldots, w_{3q}\}$, $Z = \{z_1, z_2, \ldots, z_{3q}\}$, $C = \{c_1, \ldots, c_m\}$ e $D = \{d_1, \ldots, d_m\}$. O grafo formado é bipartido. Note que podemos dividi-lo em duas partições $A = X \cup D \cup W$ e $B = C \cup Y \cup Z$, de modo que cada partição induza um conjunto independente no grafo original.

Suponha que $\mathcal{C}'$ seja uma cobertura exata por conjuntos de 3 elementos para X. Logo, $|\mathcal{C}'| = q$. Considere o conjunto $S = \{c_j \mid C_j \in \mathcal{C}'\} \cup D \cup W \cup Z$. S possui exatamente $m + 7q$ vértices e é claramente um conjunto de empacotamento aberto, já que nenhum dos vértices compartilham vizinhos.

Suponha agora que S é um conjunto de empacotamento aberto em G com $m + 7q$ vértices. Seja $S' = S \cap C$. Por construção, cada vértice de S' é adjacente com 3 vértices de

X e, como nenhum vértice de S' compartilha vizinhos, temos que existem $3|S'|$ vértices de X adjacentes a vértices de S'. Como $|X| = 3q$, temos que $|S'| \leq q$. Além disso, como cada H_i é um caminho de 4 vértices, sabemos que no máximo 2 vértices de cada um deles pode pertencer à S. Logo, S deve conter pelo menos $m + q$ vértices de $C \cup D$. Logo, $D \subset S$ e $|S'| = q$. Consequentemente, temos que $\mathcal{C}' = \{C_j \mid c_j \in S'\}$ é uma cobertura exata por conjuntos de 3 elementos de X.

□

3.2. Grafo Cordal

O problema também é NP-Completo para a classe de grafos cordais [15]. A prova proposta pelos autores é similar à anterior, com a mudança apenas na construção do grafo, de modo que ele seja cordal. Para isso, é necessário adicionar uma arestas entre todo par de vértices de X, de modo que $G[X]$ seja uma clique. É fácil notar que com essa mudança um grafo cordal é criado. Para isso, basta notar que os grafos criados dessa maneira possuem uma Ordem de Eliminação Perfeita (OEP). Considere a seguinte ordenação L dos vértices de G :

$$L = \{z_1, \ldots, z_{3q}, w_1, \ldots, w_{3q}, y_1, \ldots, y_{3q}, d_1, \ldots, d_m, c_1, \ldots, c_m, x_1, \ldots, x_{3q}\}$$

A ordenação L é uma OEP. Podemos notar isso facilmente analisando a estrutura do grafo construído anteriormente. Veja que:

- Para cada $z \in Z$, seu único vizinho é um $w \in W$, que está a sua direita em L, logo sua vizinhança induz uma clique de tamanho 1.
- Para cada $w \in W$, seu único vizinho a direita em L é um $y \in Y$, logo uma clique de tamanho 1 é induzida.
- Para cada $y \in Y$, seu único vizinho a direita em L é um $x \in X$, logo uma clique de tamanho 1 é induzida.
- Para cada $d \in D$, seu único vizinho é um $c \in C$, que está a sua direita em L, logo sua vizinhança induz uma clique de tamanho 1.
- Para cada $c \in C$, sua vizinhança à direita em L é formada apenas por elementos de X que, por construção, induzem uma clique em G.
- Para cada $x \in X$, sua vizinhança a direita em L é formada apenas por elementos de X que, por construção, induzem uma clique em G.

Os mesmos argumento utilizados para grafos bipartidos funcionam para esse caso. Podemos mostrar que, dada uma instância do problema de cobertura exata por conjuntos de 3 elementos, podemos construir um grafo cordal G de acordo com as regras citadas de modo que o problema original tenha solução se e somente se G admite um conjunto de empacotamento aberto de tamanho $m + 7q$.

3.3. Grafos Caminho

Para a classe de grafos caminho, o problema de decisão citado pode ser respondido em tempo constante [15] e o conjunto de empacotamento aberto correspondente pode ser construído em tempo linear no tamanho do grafo.

Proposição 3. *Para* $n \geq 2$*,*

$$\rho_o(P_n) = \begin{cases} \frac{n}{2} & \textit{se } n \equiv 0 \bmod 4 \\ \lfloor \frac{n+2}{2} \rfloor & c.c \end{cases}$$

Prova. Seja $G = P_n$ um grafo caminho com n vértices, denotados por $v_1, v_2, \ldots, v_n$. Suponha que $n = 4k$ para algum k inteiro, ou seja, n é múltiplo de 4. Desse modo, o conjunto de vértices de P_n pode ser particionado em k grafos P_4. No entanto, sabe-se que para cada P_4, no máximo 2 vértices podem ser escolhidos. Qualquer escolha de 3 ou mais vértices viola a propriedade de empacotamento aberto. Portanto, $\rho_o(P_n)$ não pode ser maior que $2k = n/2$. No entanto, o conjunto de vértices $S = \{v_i \mid i \equiv 1 \text{ ou } i \equiv 2 \bmod 4\}$ é um empacotamento aberto do grafo com exatamente $n/2$ vértices. Como não pode haver um maior, $\rho_o(P_n) = n/2$ se $n \equiv 0 \bmod 4$.

Suponha agora que n não seja múltiplo de 4. Note que o conjunto de vértices $S = \{v_i \mid i \equiv 1 \text{ ou } i \equiv 2 \bmod 4\}$ é um empacotamento aberto do grafo com exatamente $\lfloor (n+2)/2 \rfloor$ vértices, logo, $\rho_o(P_n) \geq \lfloor (n+2)/2 \rfloor$. No entanto, pela Proposição 1, sabe-se que $\rho_o(P_n) \leq max\{k \mid d_1 + d_2 + \cdots + d_k \leq n\} = max\{k \mid 1 + 1 + (k-2)2 \leq n\} = max\{k \mid k \leq (n+2)/2\}$. Logo, $\rho_o(P_n) = \lfloor (n+2)/2 \rfloor$ se $n \not\equiv 0 \bmod 4$. □

Considere, por exemplo, o grafo caminho com 4 vértices abaixo.

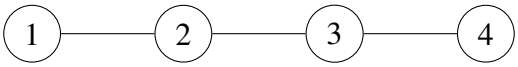

Figura 14. P_4

Pela Proposição 3, temos que o tamanho do maior conjunto de empacotamento aberto de P_4 é $\frac{4}{2} = 2$ e um desses conjuntos é $\{v_1, v_2\}$.

Considere agora o grafo caminho com 6 vértices abaixo.

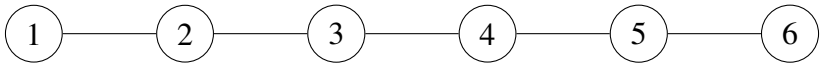

Figura 15. P_6

Pela Proposição 3, temos que o tamanho do maior conjunto de empacotamento aberto de P_6 é $\lfloor \frac{6+2}{2} \rfloor = 4$ e um desses conjuntos é $\{v_1, v_2, v_5, v_6\}$.

3.4. Grafos Ciclo

A classe de grafos ciclo é outra que possui complexidade polinomial para o problema de empacotamento aberto. Assim como os grafos caminho, o problema de decisão pode ser respondido em tempo constante [15].

Proposição 4. *Para* $n \geq 3$*,*

$$\rho_o(C_n) = \begin{cases} \frac{n}{2} - 1 & \textit{se } n \equiv 2 \bmod 4 \\ \lfloor \frac{n}{2} \rfloor & c.c \end{cases}$$

Prova. Seja $G = C_n$ um grafo ciclo com n vértices, denotados por $v_1, v_2, \ldots, v_n$. Suponha que $n \not\equiv 2 \bmod 4$. Sabe-se, pela Proposição 2, que $\rho_o(C_n) \leq n/\delta(C_n) = \lfloor n/2 \rfloor$. No entanto, o conjunto $S = \{v_i \mid i \equiv 0 \text{ ou } i \equiv 3 \bmod 4\}$ é um conjunto de empacotamento aberto de C_n com exatamente $\lfloor n/2 \rfloor$ vértices. Logo, se $n \not\equiv 2 \bmod 4$, então $\rho_o(C_n) = \lfloor n/2 \rfloor$.

Suponha agora que $n \equiv 2 \bmod 4$. Provaremos que $\rho_o(C_n) \leq n/2 - 1$. Caso contrário, então existe um conjunto de empacotamento aberto S de C_n com $n/2$ vértices. Desse modo, $\bigcup_{v \in S} N(v) = V(C_n)$ e, em particular, cada vértice $v \in S$ é adjacente a algum outro vértice $u \in S$. Isso implica que $C_n[S]$, o subgrafo induzido por S em C_n, é um conjunto de grafos K_2 disjuntos, ou seja, $|S|$ é par. No entanto, assumimos que $|S| = n/2 = \frac{4q+2}{2} = 2q + 1$, portanto chegamos a um absurdo. Logo, $\rho_o(C_n) \leq 2n - 1$. No entanto, note que o conjunto $S = \{v_i \mid i \equiv 0 \text{ ou } i \equiv 3 \bmod 4\}$ é um conjunto de empacotamento aberto de tamanho $2n - 1$. Logo, $\rho_o(C_n) = 2n - 1$.

□

Considere, por exemplo, o grafo ciclo com 6 vértices abaixo.

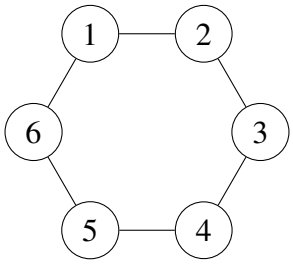

Figura 16. C_6

Pela Proposição 4, temos que o tamanho do maior conjunto de empacotamento aberto de C_6 é $\frac{6}{2} - 1 = 2$ e um desses conjuntos é $\{v_3, v_4\}$.

Considere agora o grafo ciclo com 8 vértices abaixo.

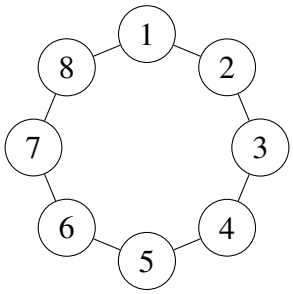

Figura 17. C_8

Pela Proposição 4, temos que o tamanho do maior conjunto de empacotamento aberto de C_8 é $\lfloor \frac{8}{2} \rfloor = 4$ e um desses conjuntos é $\{v_3, v_4, v_7, v_8\}$.

3.5. Árvores

O problema pode ser respondido em tempo polinomial quando restrito à árvores. Sabe-se por [4] que, dada uma árvore T, $\rho_o(T) = \gamma_t(T)$, ou seja, o número de empacotamento aberto de uma árvore é igual ao número de dominação total dessa árvore. Por [16], sabe-se que $\gamma_t(T)$ pode ser encontrado em tempo polinomial, logo, o problemas pode ser respondido em tempo polinomial quando restrito à arvores. No entanto, essa abordagem permite apenas encontrar o tamanho do maior empacotamento aberto em uma árvore e não como encontrar tal conjunto.

Nesse sentido, apresentamos o seguinte algoritmo guloso, fortemente baseado no algoritmo proposto por [4] para o conjunto de dominação total, que, dada uma árvore T, encontra um conjunto de empacotamento aberto S de tamanho máximo em tempo polinomial.

Seja T uma árvore enraizada com n vértices. Assumimos que os vértices de T estão rotulados de 1 até n, de modo que para $i < j$, temos que $d(1, i) \leq d(1, j)$, isto é, o vértice i está em um nível menor ou igual ao do vértice j. Tal rotulação pode ser facilmente construída com uma busca em largura. A imagem 18 ilustra uma árvore com as condições citadas.

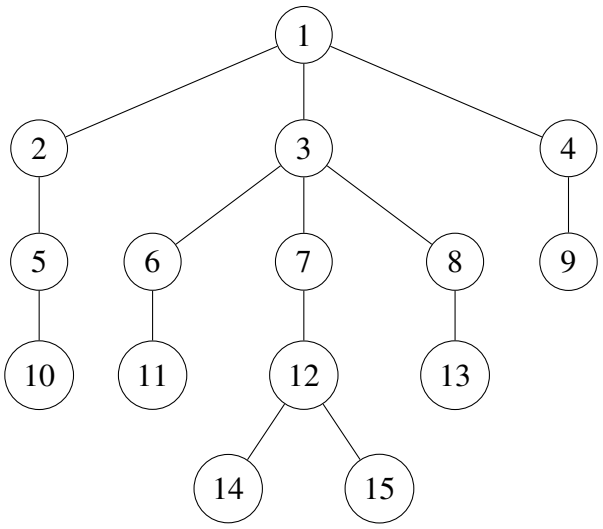

Figura 18. Árvore T enraizada

No algoritmo, inicialmente definimos um vetor denotado rotulo, de modo que rotulo$[i]$ contenha o atual rótulo do vértice i. Cada vértice pode possuir um dos três seguintes rótulos:

- **LIVRE:** não existe qualquer restrição sobre o vértice.
- **UTILIZADO:** o vértice está na solução.
- **PROIBIDO:** o vértice está proibido de estar na solução.

Inicialmente, todos os vértices estão marcados com o rótulo **LIVRE** e o conjunto de solução S está vazio. Para cada vértice de n até 1, adicionamos ou proibimos ele de acordo com seu rótulo. Se ele possuir rótulo **LIVRE**, o adicionamos em S e marcamos todos os vértices que compartilham vizinhos com ele com o rótulo **PROIBIDO**.

Se possui rótulo **UTILIZADO** ou **PROIBIDO**, o ignoramos. Abaixo apresentamos o pseudocódigo do algoritmo apresentado.

A matriz M utilizada no algoritmo é uma matriz binária bidimensional na qual a posição i, j é igual a 1 se, e somente se, os vértices i e j compartilham vizinhos. Essa matriz pode facilmente ser calculada em tempo quadrático no número de vértices do grafo.

Algoritmo 1 EMPACOTAMENTO ABERTO EM ÁRVORES

Entrada Uma árvore $T = (V, E)$ enraizada em 1 com $V = \{1, 2, \ldots, n\}$
Saída Um conjunto de empacotamento aberto de T

```
1:  S ← ∅
2:  for i ← 1 to n do
3:      rotulo[i] ← LIVRE
4:  end for
5:  for i ← n to 1 do
6:      if rotulo[i] = LIVRE then
7:          rotulo[i] ← UTILIZADO
8:          S ← S ∪ {i}
9:          for j ← 1 to n do
10:             if M[i][j] then
11:                 rotulo[j] ← PROIBIDO
12:             end if
13:         end for
14:     end if
15: end for
```

Teorema 1. *O Algoritmo 1 encontra o conjunto de empacotamento aberto máximo de uma árvore T.*

Prova. Em primeiro lugar, note que o algoritmo encontra, de fato, um conjunto de empacotamento aberto. A cada iteração, os vértices que compartilham vizinhos com os vértices na solução são marcados com o rótulo **PROIBIDO**, garantindo que o conjunto final seja de empacotamento aberto.

Suponha, por contradição, que o conjunto de empacotamento aberto $S = \{v_1, v_2, \ldots, v_{|S|}\}$, encontrado pelo algoritmo nessa ordem, não é ótimo. Seja $S^* = \{v_1^*, v_2^*, \ldots, v_{|S^*|}^*\}$ uma solução ótima encontrada nessa ordem de tal modo que o prefixo em comum entre S e S^* seja máximo, isto é, S^* é uma solução que maximiza i tal que $v_x = v_x^*$ para todo $x \leq i$.

Considere o vértice v_{i+1} e seja o índice desse vértice na árvore igual a d. Observe que nenhum vértice que compartilha vizinhos com v_{i+1} e que tenha índice maior que d faz parte de S, uma vez que v_{i+1} não teria sido escolhido pelo algoritmo se esse fosse o caso. Note que nenhum desses vértices faz parte de S^* também, uma vez que ele teria de fazer parte do prefixo em comum com S, o que é um absurdo.

Sabemos, portanto, que v_{i+1} não pertence a S^*, assim como nenhum dos vértices que compartilham vizinhos com v_{i+1} com índice maior que d. Logo, deve existir em S^*

um vértice que compartilha vizinhos com v_{i+1} e com índice menor que d. Caso contrário, haveria uma contradição na maximalidade de S^*.

Mostraremos que podemos substituir um desses vértices em S^* por v_{i+1}, o que contrariaria a premissa de que o maior prefixo em comum entre as soluções possui tamanho i. Sabemos que nenhum vértice que compartilha vizinhos com v_{i+1} e que tenha índice maior que d faz parte de S^*, logo a troca é segura considerando os vizinhos de v_{i+1} com índice menor que d. Note que os vértices que compartilham vizinhos com v_{i+1} na árvore são ou um irmão de v_{i+1}, isto é, um vértice que compartilha o mesmo vértice pai de v_{i+1}, ou seu avô, isto é, o vértice pai de seu pai.

No caso do irmão, note que a troca é segura. Como eles compartilham os mesmos irmãos e o mesmo avô, nenhuma regra de empacotamento aberto será quebrada ao fazer a troca. No caso do avô, a troca também é segura. Note que, se o vértice avô fazia parte de S^*, então nenhum de seus netos poderia fazer parte. Logo, ao adicionar v_{i+1} na solução e remover o seu vértice avô, nenhuma regra de empacotamento aberto é quebrada.

Portanto, encontramos uma solução de tamanho igual à ótima, mas com prefixo em comum com a solução S maior do que i, o que é um absurdo. Logo, a suposição inicial é falsa, o que nos permite concluir que o algoritmo encontra um empacotamento aberto de tamanho máximo.

□

O algoritmo 1 possui complexidade $\mathcal{O}(n^2)$, onde $n = |V(T)|$. O gargalo principal está na computação da matriz M, enquanto o resto do algoritmo é executado em tempo linear.

Para exemplificar, considere a árvore T na imagem 18.

Inicialmente, todos os vértices estão marcados com o rótulo **LIVRE** e $S = \emptyset$. O vértice 15 é o primeiro a ser checado e, como seu rótulo é **LIVRE**, ele é adicionado a S e marcado como **UTILIZADO**. Os vértices 14 e 7, por sua vez, são marcados com o rótulo **PROIBIDO**, já que compartilham o vértice 12 como vizinho com o vértice 15.

Como 14 está marcado como **PROIBIDO**, o algoritmo segue para o vértice 13, marcado como **LIVRE**. Seu rótulo então passa a ser **UTILIZADO**, ele é adicionado a S e o vértice 3 é marcado como proibido. Os próximos vértices analisados são 12 e 11, marcados como **LIVRE**. Os mesmos passos são seguidos assim como nos outros casos.

O próximo vértice checado é o 11, marcado como **LIVRE**. Ele é adicionado em S, marcado como **UTILIZADO**, e o vértice 2 é marcado como **PROIBIDO**, uma vez que compartilha vizinhos com o vértice 10. O vértice 9, marcado como **LIVRE**, também é adicionado a S e o vértice 1, que compartilha vizinhos com 9, é marcado como **PROIBIDO**.

O vértice 8, marcado como **LIVRE**, é adicionado a S, marcado como **UTILIZADO**, e os vértices 1, 6 e 7 marcados como proibidos, uma vez que compartilham vizinhos com o vértice 8. Como 7 e 6 estão proibidos, o algoritmo irá ignorá-los e iremos checar o vértice 5, marcado como **LIVRE**. Mais uma vez, seu rótulo passa a ser **UTILIZADO**, ele é adicionado a S e 1 é marcado como **PROIBIDO**.

O próximo vértice checado é o 4, marcado como **LIVRE**. Ele é adicionado na solução, marcado como **UTILIZADO**, e os vértices 2 e 3 marcados com o rótulo **PROIBIDO**. Por fim, como os vértices 1, 2 e 3 estão proibidos, o algoritmo irá ignorá-los e irá parar. Por fim, teremos $S = \{15, 13, 12, 11, 10, 9, 8, 5, 4\}$, com $|S| = \rho_o(T) = 9$.

3.6. Grafos de Intervalo

Encontrar um empacotamento aberto de tamanho máximo em um grafo de intervalo pode ser feito em tempo polinomial [7]. O algoritmo proposto para solucionar o problema utiliza programação dinâmica e possui complexidade $\mathcal{O}(n^3)$, onde n é a ordem do grafo G.

Para apresentar o algoritmo, alguns conceitos devem ser definidos. Sabe-se que, para um grafo de intervalo G de grau n, um conjunto de intervalos $\mathcal{F} = \{(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)\}$ cujo grafo de interseção é isomórfico a G pode ser encontrado em tempo linear [18]. Por conveniência, assumimos que F está ordenado de acordo com o tempo de final de cada intervalo e, em caso de empate, de acordo com o tempo de início. Além dos intervalos originais, criaremos um intervalo artificial (a_0, b_0), tal que $b_0 < a_i$ para todo intervalo $i \in \mathcal{F}$, isto é, um intervalo cujo final é menor que o início de qualquer outro intervalo.

Denotaremos por $q(i, j)$, para $i < j$, o tamanho do maior empacotamento aberto de G que seja um subconjunto de $\{1, \ldots, j\}$ e que contenha i e j mas não contenha nenhum vértice k tal que $i < k < j$. Trivialmente, $q(0, j) = 1$ para todo j, pois a solução irá conter apenas o próprio vértice j. Os valores de $q(i, j)$ podem então ser calculados a partir da seguinte recorrência:

$$q(i,j) = \begin{cases} 1, & \text{se } i = 0 \\ -\infty, & \text{se } N(i) \cap N(j) \neq \emptyset \\ \max\limits_{\substack{0 \leq h < i \\ N(h) \cap N(j) = \emptyset}} q(h,i) + 1, & \text{c.c.} \end{cases}$$

Teorema 2. *Se G for um grafo de intervalo, o algoritmo 2 encontra $\rho_o(G)$, o tamanho do maior empacotamento aberto de G.*

Prova. Inicialmente, vamos provar o caso base. Como já citado anteriormente, se $i = 0$, temos que $q(i, j) = 1$, uma vez que apenas o vértice j irá fazer parte da solução. É importante lembrar que o vértice 0 é artificial e por isso não irá fazer parte da solução final.

No caso em que i e j compartilham vizinhos, evidentemente não pode haver uma solução, isto é, não pode haver empacotamento aberto que contenha ambos os vértices i e j, portanto um valor simbólico de menos infinito é utilizado.

O terceiro caso deve ser analisado com mais cuidado. Note que, dadas as definições apresentadas, um conjunto de empacotamento aberto (EA) parcial S' pode ser estendido com um novo vértice v se esse vértice não compartilhar vizinhos com nenhum $u \in S'$. Seja agora S' um conjunto de EA em que i é o maior vértice (intervalo mais à direita na ordenação) e h o segundo maior. Iremos provar que um novo vértice j pode ser

Algoritmo 2 EMPACOTAMENTO ABERTO EM GRAFOS DE INTERVALO

Entrada Um grafo de intervalo $G = (V, E)$ e seu respectivo conjunto de intervalos $\mathcal{F}$
Saída Um conjunto de empacotamento aberto de G

1: Compute a matriz M
2: Crie a matriz q de tamanho nxn
3: **for** $0 < i \leq n$ **do**
4: $\quad q(0, i) \leftarrow 1$
5: **end for**
6: **for** $0 < j < n$ **do**
7: $\quad val \leftarrow 1$
8: $\quad max \leftarrow 1$
9: $\quad$ **for** $0 < i < j$ **do**
10: $\quad\quad$ **if** $M(j, i) = 0$ **then**
11: $\quad\quad\quad$ **for** $0 \leq h < i$ **do**
12: $\quad\quad\quad\quad$ **if** $q(h, i) \geq val$ e $M_{j,h} = 0$ **then**
13: $\quad\quad\quad\quad\quad val \leftarrow q(h, i) + 1$
14: $\quad\quad\quad\quad$ **end if**
15: $\quad\quad\quad$ **end for**
16: $\quad\quad$ **else**
17: $\quad\quad\quad val \leftarrow -\infty$
18: $\quad\quad$ **end if**
19: $\quad\quad q(i, j) \leftarrow val$
20: $\quad\quad$ **if** $q(i, j) > max$ **then**
21: $\quad\quad\quad max \leftarrow q(i, j)$
22: $\quad\quad$ **end if**
23: $\quad$ **end for**
24: **end for**
25: **return** max

adicionado a S' se, e somente se, h e j não compartilham vizinhos. Assumimos que i e j não possuem vizinhos em comum. Caso contrário, sequer entraríamos nessa condição da recorrência.

Primeiramente, nota-se que, se j pode ser inserido em S', então $N(j) \cap N(h) = \emptyset$, pela própria definição de empacotamento aberto.

Agora, suponha que j e h não compartilham vizinhos. Pela hipótese inicial seria um absurdo que j e h fossem vizinhos, dado que isso implicaria em j e i também serem vizinhos, dado que i está a direita de h e tal relação de vizinhança implicaria em i e j, dois elementos de S', compartilharem um vizinho. Portanto, podemos assumir que $b_h < a_j$.

Suponha agora que existe um vértice h' menor que h que pertence a S' e possui um vizinho w em comum com j. Note que, nesse caso, temos que $a_w < b_{h'} < b_h$,mas ao mesmo tempo teríamos que $b_w > a_j > b_h$, o que implicaria em w ser um vizinho em comum entre h e h', o que é um absurdo dado que assumimos que h e h' pertenciam a S', um conjunto de EA. Portanto, os conjuntos de EA que podem ser estendidos por j são exatamente aqueles em que i é o maior vértice, h o segundo e j e h não compartilham vizinhos. Na recorrência, portanto, basta escolher o maior dos conjuntos obtidos ao estender a solução atual como cada um dos possíveis j e somar 1, e assim teremos o tamanho do maior conjunto de EA em que j é o maior vértice e i o segundo maior.

Como, ao final da execução do algoritmo teremos o valor de $q(i, j)$ para todo $0 < i < j < |V|$, basta pegar o maior desses valores e teremos o tamanho do empacotamento aberto máximo do grafo.

□

Para iniciar a execução do algoritmo, assumimos que o conjunto de intervalos referentes a G era dado, embora ele possa ser encontrado em tempo linear, como citado anteriormente. Podemos assumir também que todos os intervalos são inteiros e que o tamanho deles é linear em n, permitindo que a ordenação deles também possa ser feita em tempo linear.

Para checar se dois vértices i e j compartilham vizinhos de maneira eficiente, é computada previamente a matriz M, que armazena um valor binário onde M_{ij} é 1 se e somente se os vértices i e j compartilharem vizinhos. A matriz M pode ser encontrado em tempo $\mathcal{O}(n^3)$, bastando checar, para todo par de vértices, se suas listas de adjacência são disjuntas ou não.

O algoritmo possui três laços aninhados, e cada um deles executará uma quantidade de vezes limitada por $\mathcal{O}(n)$. Como a matriz M já foi computada previamente, verificar que dois vértices não possuem vizinhos em comum pode ser feito em tempo constante. Logo, a complexidade final do algoritmo é $\mathcal{O}(n^3)$.

O algoritmo proposto apenas retorna o tamanho do maior conjunto de empacotamento aberto de um grafo de intervalo, mas não encontra quais vértices compõe esse conjunto. Para isso, basta fazer uma pequena modificação no algoritmo e armazenar em $q(i, j)$ os vértices que fazem parte daquele conjunto de EA.

Para compreender melhor o algoritmo, considere o seguinte conjunto de intervalos e seu correspondente grafo G retirados do exemplo utilizado em [7].

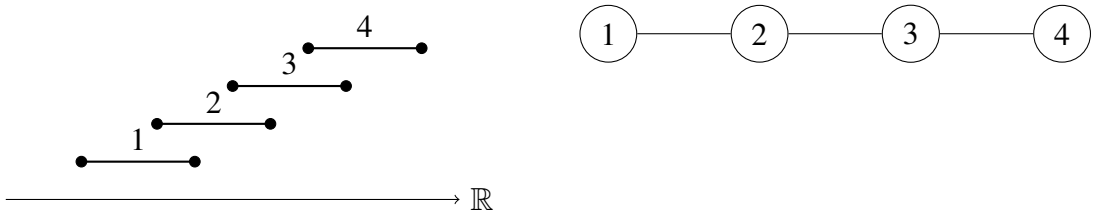

Figura 19. Exemplo de conjunto de intervalos e seu respectivo grafo de intervalo G

A matriz M do grafo G ficará preenchida da seguinte forma:

	0	1	2	3	4
0	0	0	0	0	0
1	0	1	0	1	0
2	0	0	1	0	1
3	0	1	0	1	0
4	0	0	1	0	1

Como o vértice 0 é artificial e não possui vizinhos, a linha e a coluna correspondentes à ele serão formadas por 0. Note também que $M_{i,i} = 1$ para todo i, uma vez que um vértice compartilha vizinhos com ele mesmo, nesse caso.

A partir da construção da matriz M podemos iniciar a execução do algoritmo principal. Inicialmente, preenchemos a matriz de programação dinâmica com os valores do caso base, isto é, $q(0, j) = 1$ para todo $j \in V(G)$.

	0	1	2	3	4
0		1	1	1	1
1					
2					
3					
4					

Considere agora $j = 1$. Como 0 é o único inteiro menor do que 1 na ordenação e $q(0, 1)$ já está preenchido, podemos partir para $j = 2$. Para $j = 2$, temos que $i = 1$ é o único inteiro menor que j na ordenação e que ainda não foi preenchido. Como $M_{2,1} = 0$, isto é, os vértices 1 e 2 não compartilham vizinhos, podemos buscar $q(h, 1)$. O único valor que h pode ter é 0 e, como $M_{2,0} = 0$, podemos calcular $q(1, 2) = q(0, 1) + 1 = 1 + 1 = 2$.

	0	1	2	3	4
0		1	1	1	1
1			2		
2					
3					
4					

Para $j = 3$, 1 e 2 são os valores inteiros menores que 3 na ordenação que ainda não

forma preenchidos. No entanto, como $M_{3,1} = 1$, temos $q(1,3) = -\infty$. No entanto, como $M_{3,2} = 0$, podemos procurar algum $q(h,2)$ que possa ser estendido por j. Nesse caso, h pode ser apenas 0, uma vez que $M_{3,1} = 1$. Logo, $q(2,3) = q(0,2) + 1 = 1 + 1 = 2$.

	0	1	2	3	4
0		1	1	1	1
1			2	$-\infty$	
2				2	
3					
4					

Por fim, para $j = 4$, temos que os possíveis valores de i são $\{1,2,3\}$. Como $M_{4,1} = 0$, podemos calcular $q(1,4) = q(0,1) + 1 = 1 + 1 = 2$. Para $i = 2$, como $M_{4,2} = 1$, temos que $q(2,4) = -\infty$. Para finalizar, temos o caso $i = 3$. Os possíveis valores de h nesse caso são $\{0,1\}$, uma vez que 2 compartilha vizinho com 4. Como $q(1,3) = -\infty$, temos que $q(3,4) = q(0,3) + 1 = 1 + 1 = 2$.

	0	1	2	3	4
0		1	1	1	1
1			2	$-\infty$	2
2				2	$-\infty$
3					2
4					

Temos como resultado final portanto que $\rho_o(G) = 2$, o que pode ser checado pela proposição 3.

4. Trabalhos Futuros

4.1. Grafos Bloco

Como abordado na seção 2, a classe de grafos bloco é a classe grafos que admitem a propriedade de que toda componente bi-conexa é uma clique [13]. Outras caracterizações importantes também são conhecidas para essa classe [21], como as seguintes:

- Uma grafo G é bloco se e somente se for livre de de $C_{n\geq 4}$ e de diamante [2]. Denotamos por diamante o grafo formado ao adicionar uma corda ao grafo C_4.
- Um grafo é bloco se e somente se existir apenas um caminho induzido entre qualquer par de vértices de G [24].

Os grafos bloco já foram estudados sobre diversas óticas diferentes na literatura e a complexidade computacional de diferentes problemas já foi descoberta para ela. Por se tratar de uma subclasse de grafos cordais e de grafos distância-hereditários, muitos resultados podem ser reaproveitados. Em particular, a tabela abaixo destaca os principais resultados já obtidos até o momento para a classe de grafos bloco.

Problema	Complexidade	Referência
Coloração	P	[11]
Conjunto Independente	P	[30] [10]
Dominação	P	[25]
Dominação total	P	[5]

Tabela 2. Tabela de complexidade para diferentes problemas restritos à classe de grafos bloco

O problema de empacotamento aberto (OPK), até onde foi encontrado pelos autores, não foi estudado quando restrito à essa classe e, por ser uma superclasse de árvores e subclasse de cordais, é natural questionar qual seria sua complexidade.

É importante notar que, apesar de ser uma superclasse de árvores, ela não mantém a propriedade de que, quando restrito à classe, um grafo G possui um número de empacotamento aberto igual ao número de dominação total, isto é, $\gamma_t(G) = \rho_o(G)$. Considere, por exemplo, o grafo G abaixo.

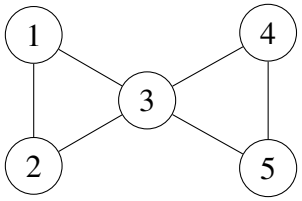

Figura 20. G

Claramente G é um grafo bloco. O número de empacotamento aberto de G é 1. Como todo par de vértices compartilha vizinhos, apenas um deles pode ser escolhido para estar na solução. Note, no entanto, que o número de dominação total de G é 2, e um dos possíveis conjuntos de dominação total é $\{1, 3\}$.

Logo, uma nova abordagem deve ser desenvolvida para calcular o número de empacotamento aberto de um grafo bloco. Algumas propriedades dos grafos bloco podem ser exploradas para iniciar uma busca pela solução do problema. Primeiramente, é importante notar que para cada bloco de tamanho 3 ou maior, apenas um vértice pode pertencer ao conjunto de empacotamento aberto. Isso ocorre devido ao fato de que cada bloco é uma clique.

Em segundo lugar, quando dois ou mais blocos de tamanho 3 ou maior compartilham um vértice de corte, a escolha de um vértice pertencente à um desses blocos para o conjunto de empacotamento aberto elimina a possibilidade de que qualquer outro faça parte do conjunto.

Em um caso mais específico, considere a classe de grafos bloco que também possuem as seguintes restrições:

- Todo bloco possui tamanho 3 ou maior.
- Todo bloco possui ao menos um vértice que não é um vértice de corte.

Uma possível abordagem para encontrar o número de empacotamento aberto nessa classe seria a de construir um grafo de interseção dos blocos do grafo e encontrar o tamanho de seu conjunto independente máximo. Por se tratar de um grafo bloco, por definição,

podemos encontrar esse conjunto em tempo polinomial. Essa redução e abordagem podem ser formalizadas e utilizadas como apoio para atacar o problema geral para grafos bloco.

Uma outra possível maneira de enfrentar o problema é a de construir um algoritmo guloso, similar ao proposto para a classe de árvores no Algoritmo 1. A ideia central dessa abordagem seria a de escolher gulosamente vértices a partir de blocos com poucos vértices de corte.

4.2. Grafos Cacto

Um grafo cacto é um grafo que admite a propriedade de que toda aresta pertence a no máximo um ciclo. Assim como os grafos bloco, essa classe já foi estudada na literatura sobre diversas óticas e diversos algoritmos foram encontrados para ela. Como também se trata de uma superclasse de árvores, é natural questionar a complexidade computacional do problema de empacotamento aberto.

No estudo de algoritmos para a classe de grafos cacto é comum a utilização de uma estrutura de árvore [6], denotada por T_{BC}, construída da seguinte maneira. Utilizando uma busca em profundidade, constrói-se o grafo de intersecção dos blocos de G, denotado por G'. A partir disso, seleciona-se um vértice arbitrário de G', que contenha pelo menos dois vértices de corte de G, como raiz da árvore T_{BC} e ele é marcado. Todos os vértices adjacentes a essa raiz são considerados filhos de nível um e também são marcados. Se houver arestas entre os vértices desse nível, elas são descartadas. Cada vértice de nível um é analisado individualmente para encontrar os vértices que são adjacentes a eles, mas não estão marcados. Esses vértices são tomados como filhos dos respectivos vértices de nível um e posicionados no nível dois. Esses filhos no nível dois são marcados e, se houver alguma aresta entre eles, elas são descartadas. Este processo continua até que todos os vértices sejam marcados.

A árvore T_{BC} pode ser uma abordagem para a resolução do problema de empacotamento aberto para a classe de grafos cacto. Utilizando ela, algoritmos polinomiais que usam de programação dinâmica foram descobertos para a solução de problemas como conjunto dominante mínimo e conjuntos independente máximo.

5. Conclusão

A monografia teve como tema principal a complexidade computacional do problema de empacotamento aberto quando restrito a diferentes classes de grafos. O trabalho se propôs a organizar um *survey* sobre o tema, aglomerando e organizando os principais estudos já realizados e os principais resultados já obtidos.

Foram apresentadas as complexidades computacionais para diferentes e importantes classes de grafos, como os grafos cordais e os grafos de intervalo. Além disso, foi apresentado um algoritmo guloso para computar, em tempo polinomial, um conjunto de empacotamento aberto de tamanho máximo em árvores. Por fim, foi estudado o problema de empacotamento aberto quando restrito às classes de grafos bloco e cacto, e abordagens para solucionar o problema foram propostas para futuras pesquisas.

Referências Bibliográficas

[1] M. Doost Ali, B. Samadi, A. Khodkar, and H. R. Golmohammadi. On the packing number in graphs. *Australasian Journal of Combinatorics*, 71(3):468–475, 2018.

[2] Hans-Jürgen Bandelt and Henry Martyn Mulder. Distance-hereditary graphs. *Journal of Combinatorial Theory, Series B*, 41(2):182–208, 1986.

[3] A. A. Bertossi. Total domination in interval graphs. *Inform. Process. Lett*, 23:131–134, 1986.

[4] Boštjan Brešar, Kirsti Kuenzel, and Douglas F Rall. Graphs with a unique maximum open packing. *arXiv preprint arXiv:1901.09859*, 2019.

[5] Gerard J Chang. Total domination in block graphs. *Operations research letters*, 8(1):53–57, 1989.

[6] Kalyani Das. Cactus graphs and some algorithms, 2014.

[7] S. R. Dias and V. F. dos Santos. *Problema do Empacotamento Aberto em Classes de Grafos*. Dissertação de mestrado, Universidade Federal de Minas Gerais, 2023.

[8] Reinhard Diestel. *Graph Theory*. Springer, 5th edition, 2017.

[9] Rodney G Downey, Michael R Fellows, et al. *Fundamentals of parameterized complexity*, volume 4. Springer, 2013.

[10] Fănică Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16(1):47–56, 1974.

[11] Martin Charles Golumbic. *Algorithmic graph theory and perfect graphs*. Elsevier, 2004.

[12] Majid Hajian, Michael A Henning, and Nader Jafari Rad. A classification of cactus graphs according to their total domination number. *Bulletin of the Malaysian Mathematical Sciences Society*, 43(2):1555–1568, 2020.

[13] Frank Harary. A characterization of block-graphs. *Canadian Mathematical Bulletin*, 6(1):1–6, 1963.

[14] M. A. Henning. Packing in trees. *Discrete Mathematics*, 186:145–155, 1997.

[15] M. A. Henning and P. J. Slatter. Open packing in graphs. *JCMCC*, 29:3–16, 1999.

[16] Michael A Henning. A survey of selected recent results on total domination in graphs. *Discrete Mathematics*, 309(1):32–63, 2009.

[17] Michael A. Henning and Anders Yeo. *Complexity and Algorithmic Results*, pages 19–29. Springer New York, New York, NY, 2013.

[18] W. L. Hsu and T. H. Ma. Substitution decomposition on chordal graphs and applications. In Wen-Lian Hsu and R. C. T. Lee, editors, *ISA'91 Algorithms*, pages 52–60, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.

[19] Richard Karp. Reducibility among combinatorial problems (1972). 2021.

[20] J. M. Keil. The complexity of domination problems in circle graphs. *Discrete Appli. Math*, 42:51–63, 1993.

[21] Lilian Markenzon and Christina Fraga Esteves Maciel Waga. Characterizing block graphs in terms of one-vertex extensions. *TEMA (São Carlos)*, 20:323–330, 2019.

[22] A. A. McRae. Generalizing np-completeness proofs for bipartite and chordal graphs. *Ph.D Thesis*, 1994.

[23] M. Mohammadi and M. Maghasedi. A note on the open packing number in graphs. *Mathematica Bohemica*, 144(2):221–224, 2018.

[24] H Mulder. An observation on block graphs. *Bulletin of the Institute of Combinatorics and its Applications*, 77:57–58, 2016.

[25] Falk Nicolai and Thomas Szymczak. Homogeneous sets and domination: A linear time algorithm for distance—hereditary graphs. *Networks: An International Journal*, 37(3):117–128, 2001.

[26] J. Pfaff, R. C. Laskar, and S. T. Hedetniemi. Np-completeness of total and connected domination and irredundance for bipartite graphs. *Technical report*, 428:85–103, 1983.

[27] J. Pfaff, R. C. Laskar, S. T. Hedetniemi, and S. M. Hedetniemi. On the algorithmic complexity of total domination. *Algebraic Discrete Methods*, 5:420–425, 1984.

[28] Igor Ramos, Vinícius F Santos, and Jayme L Szwarcfiter. Complexity aspects of the computation of the rank of a graph. *Discrete Mathematics & Theoretical Computer Science*, 16(PRIMA 2013), 2014.

[29] Donald J Rose. Triangulated graphs and the elimination process. *Journal of Mathematical Analysis and Applications*, 32(3):597–609, 1970.

[30] Donald J Rose, R Endre Tarjan, and George S Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on computing*, 5(2):266–283, 1976.

[31] I Sahul Hamid and S Saravanakumar. On open packing number of graphs. *Iranian Journal of Mathematical Sciences and Informatics*, 12(1):107–117, 2017.

[32] M. A. Shalu and V. K. Kirubakaran. Open packing in H-free graphs and subclasses of split graphs. *Algorithms and Discrete Applied Mathematics*, pages 239–251, 2024.

[33] Jerzy Topp and Lutz Volkmann. On packing and covering numbers of graphs. *Discrete mathematics*, 96(3):229–238, 1991.