UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

ANA CAROLYNE PEREIRA DE SOUZA

Caracterizações, Reconhecimento e Conjuntos Independentes Máximos de Grafos de Intervalo

Goiânia
2024

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO NO REPOSITÓRIO INSTITUCIONAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio do Repositório Institucional (RI/UFG), regulamentado pela Resolução CEPEC no 1240/2014, sem ressarcimento dos direitos autorais, de acordo com a Lei no 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo dos Trabalhos de Conclusão dos Cursos de Graduação disponibilizado no RI/UFG é de responsabilidade exclusiva dos autores. Ao encaminhar(em) o produto final, o(s) autor(a)(es)(as) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do Trabalho de Conclusão de Curso de Graduação (TCCG)

Nome(s) completo(s) do(a)(s) autor(a)(es)(as): Ana Carolyne Pereira de Souza

Título do trabalho: Caracterizações, Reconhecimento e Conjuntos Independentes Máximos de Grafos de Intervalo

2. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador) Concorda com a liberação total do documento [X] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante: a) consulta ao(à)(s) autor(a)(es)(as) e ao(à) orientador(a); b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo do TCCG. O documento não será disponibilizado durante o período de embargo.

Casos de embargo:

- Solicitação de registro de patente;

- Submissão de artigo em revista científica;

- Publicação como capítulo de livro.

Obs.: Este termo deve ser assinado no SEI pelo orientador e pelo autor.

Documento assinado eletronicamente por **Julliano Rosa Nascimento**, **Professor do Magistério Superior**, em 04/06/2025, às 15:52, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

Documento assinado eletronicamente por **Ana Carolyne Pereira De Souza**, **Discente**, em 04/06/2025, às 15:52, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

ANA CAROLYNE PEREIRA DE SOUZA

Caracterizações, Reconhecimento e Conjuntos Independentes Máximos de Grafos de Intervalo

Trabalho de Conclusão apresentado à Coordenação do Curso de Ciência da Computação do Insituto de Informática da Universidade Federal de Goiás, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Área de concentração: Ciência da Computação.

Orientador: Prof. Julliano Rosa Nascimento

Goiânia
2024

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Souza, Ana Carolyne Pereira de
Caracterizações, reconhecimento e conjuntos independentes máximos de grafos de intervalo [manuscrito] / Ana Carolyne Pereira de Souza. - 2024.
XLIV, 44 f.: il.

Orientador: Prof. Dr. Julliano Rosa Nascimento.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Goiás, Instituto de Informática (INF), Ciência da Computação, Goiânia, 2024.
Bibliografia.
Inclui lista de figuras.

1. teoria dos grafos. 2. grafos de intervalo. 3. modelo de interseção. 4. algoritmo. 5. conjunto independente. I. Nascimento, Julliano Rosa, orient. II. Título.

CDU 004

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

BANCA EXAMINADORA

ANA CAROLYNE PEREIRA DE SOUZA

Caracterizações, Reconhecimento e Conjuntos Independentes Máximos de Grafos de Intervalo

Trabalho de conclusão de curso apresentado à Universidade Federal de Goiás como parte dos requisitos para a obtenção do título de Bacharel em Ciência da Computação.
Orientador: Prof. Dr.: Julliano Rosa Nascimento

Aprovado em 09/12/2024.

Examinadores:

Profa. Dra. Diane Castonguay
Universidade Federal de Goiás
Instituto de Informática

Prof. Dr. Julliano Rosa Nascimento
Universidade Federal de Goiás
Instituto de Informática

Documento assinado eletronicamente por **Diane Castonguay**, **Professor do Magistério Superior**, em 09/12/2024, às 20:13, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

Documento assinado eletronicamente por **Julliano Rosa Nascimento**, **Professor do Magistério Superior**, em 09/12/2024, às 20:23, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **5024418** e o código CRC **74A70E54**.

Referência: Processo nº 23070.062617/2024-76 SEI nº 5024418

Ana Carolyne Pereira de Souza

Graduanda em Ciência da Computação pela Universidade Federal de Goiás. Durante a sua graduação, participou ativamente da Empresa Júnior Level 5, foi monitora no Instituto de Informática em Algoritmos e Estrutura de Dados 2 e pesquisadora voluntária de iniciação científica em Teoria dos Grafos. Atualmente, é pesquisadora de iniciação científica em Tecnologias Assistivas e estagiária na Secretaria do Estado da Admnistração de Goiás.

Dedico este trabalho, primeiramente, a Deus, à minha família, aos meus amigos e ao meu orientador, que sempre me apoiaram.

Agradecimentos

Gostaria de agradecer, em primeiro lugar, a Deus pela oportunidade de realizar o meu sonho e pelas pessoas incríveis que Ele permitiu que eu conhecesse durante a minha jornada acadêmica. À minha família, pelo suporte. Em especial, gostaria de agradecer à minha mãe, Núbia, que é a minha fonte de resiliência e a mulher mais incrível e forte que eu conheço. Mãe, muito obrigada pelo seu apoio, por sempre acreditar no meu potencial e por diversas vezes ter se sacrificado para que eu conseguisse alcançar meus objetivos. Ao meu pai, Marcilino, que é o meu maior exemplo de determinação e perseverância. Pai, muito obrigada por sempre me tranquilizar nos momentos difíceis e me encorajar a ir atrás dos meus sonhos. À minha avó, Ivanilda, que generosamente me acolheu e cuidou de mim com todo o carinho e esmero do mundo. Vocês são meu alicerce e as maiores inspirações da minha vida. Saibam que sou extremamente grata pelo zelo e amor incondicional que sempre demonstraram por mim.

Às minhas queridas amigas Ana Luísa, Beatriz, Fernanda e Geovanna, que me acolheram desde o primeiro momento em que nos conhecemos e se tornaram minha segunda família. Sou muito grata a vocês por tornarem essa jornada mais leve e divertida, por cada risada compartilhada e apoio nos momentos difíceis, sempre me fazendo sentir amparada e motivada a continuar.

Ao Caio, que é meu maior exemplo de generosidade e minha principal inspiração para que eu seguisse pela área da Computação. Caio, muito obrigada por sempre acreditar em mim, por tornar possível a realização do meu sonho e por me proporcionar o suporte necessário para atravessar esse processo da maneira mais confortável e enriquecedora possível. Sou eternamente grata a Deus por permitir que eu te conhecesse. Saiba que você é uma das pessoas mais especiais para mim e que sempre guardarei você no meu coração com muito carinho.

Finalmente, ao meu orientador, Julliano, que abriu as portas para que eu me apaixonasse pela Teoria da Computação, especificamente, pelos Grafos. Agradeço pela honra de ser sua orientanda, por toda a dedicação, compreensão, paciência e, principalmente, pela exigência que me impulsionou a sempre buscar o melhor. O senhor é uma grande inspiração, e sua trajetória me motiva a seguir cada vez mais firme no pelo caminho acadêmico.

Eu não quero ser lembrada como alguém que se conformou com os papéis tradicionais de gênero, mas sim como uma mulher que quebrou barreiras e inspirou outras mulheres através da tecnologia e da educação.

Autor desconhecido.

Resumo

SOUZA, Ana Carolyne Pereira de. Caracterizações, reconhecimento e conjuntos independentes máximos de grafos de intervalo. 2024. 44 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Instituto de Informática, Universidade Federal de Goiás, Goiânia, 2024.

Um grafo G é um grafo de intervalo se seu conjunto de vértices pode ser representado através de um modelo de interseção de intervalos na reta real. Esta monografia apresenta um estudo sobre algumas propriedades de grafos de intervalo, como a ausência de ciclos induzidos com quatro ou mais vértices e triplas asteroidais. Com base nas propriedades apresentadas, dado um grafo de intervalo G, desenvolvemos um algoritmo polinomial que usa o esquema de eliminação perfeita, obtido através do algoritmo de busca em largura lexicográfica, para criar um modelo de intervalo para um grafo de intervalo G. Além disso, mostramos um algoritmo polinomial para determinar um conjunto independente máximo em um grafo de intervalo G. Também apresentamos uma caracterização de grafos de intervalo unitário como sendo grafos de intervalo livres de $K_{1,3}$.

Palavras–chave

Teoria dos Grafos, Grafos de Intervalo, Modelo de Interseção, Algoritmo, Conjunto Independente.

Abstract

SOUZA, Ana Carolyne Pereira de. Caracterizations, recognition, and maximum independent sets of interval graphs. 2024. 44 f. Trabalho de Conclusão de Curso Bacharelado em Ciência da Computação) - Instituto de Informática, Universidade Federal de Goiás, Goiânia, 2024.

A graph G is an interval graph if its vertex set can be represented by an interval intersection model on the real line. This monograph presents a study of some properties of interval graphs, such as the absence of induced cycles with four or more vertices and asteroidal triples. Based on the presented properties, given an interval graph G, we develop a polynomial-time algorithm that uses the perfect elimination scheme, obtained through the lexicographic breadth-first search algorithm, to create an interval model for the interval graph G. Additionally, we present a polynomial-time algorithm for determining a maximum independent set in an interval graph G. We also provide a characterization of unit interval graphs as interval graphs free of $K_{1,3}$. .

Keywords

Graph Theory, Interval Graphs, Intersection Model, Algorithm, Independent Set.

Sumário

Lista de Figuras

Lista de Algoritmos

CAPÍTULO 1

Introdução

A Teoria dos Grafos é uma área fundamental da matemática que tem aplicabilidade em diversos campos, como a ciência da computação, biologia, redes de comunicação, entre outros. Grafos são capazes de representar relações entre objetos e o estudo dessas estruturas pode possibilitar a resolução de vários problemas complexos de maneira eficiente. Dentro dessa teoria, os grafos de intervalo são uma classe que chamam bastante atenção não somente pelo interesse teórico, mas também devido às suas características e aplicações que exemplificam bem a utilidade dos grafos em modelar e solucionar diversos problemas reais.

Formalmente, um *grafo* G é um par ordenado $G = (V, E)$, que consiste de um conjunto V de vértices e um conjunto E de arestas de forma que $E(G) \subseteq [V]^2$. Dizemos que G é um *grafo de intervalo* se V pode ser representado através de um modelo de interseção de intervalos na reta real.

A fim de evidenciar o papel dos grafos de intervalo em serem utilizados para modelar problemas, considere o Problema do Gene, formulado pelo biólogo americano Seymour Benzer, o qual está relacionado com a estrutura fina dos genes [17]. Benzer estava interessado em entender como os subelementos, isto é, as partes menores que compõem os genes, estão organizados. O intuito era saber se esses subelementos poderiam ser dispostos em uma sequência única.

Para esta investigação, Benzer utilizou um organismo modelo, um microorganismo, e estudou suas formas padrão mutantes. Os mutantes surgem quando uma parte específica da estrutura genética está danificada. Sendo assim, Benzer realizou testes de recombinações para verificar se as partes danificadas de dois mutantes diferentes se intersectavam. Esses testes forneciam dados sobre insterseções das porções danificadas. O objetivo, então, era determinar se esses dados era compatível com uma estrutura linear dos genes.

No que se relaciona aos grafos, o problema pode ser modelado como a verificação de se um grafo que representa as insterseções entre os subelementos pode ser estruturado como um caminho, ou seja, uma sequência linear. Nesse caso, cada vértice do grafo corresponde a um subelemento, e uma aresta entre dois vértices indica que os

subelementos representados por esses vértices se intersectam.

O problema então torna-se determinar se um grafo é um caminho, ou seja, se existe uma sequência linear dos vértices onde cada vértice está conectado com o próximo, sem que haja ciclos ou estruturas mais complexas. Isso envolve verificar a ausência de ciclos de comprimento maior ou igual a 4 e garantir que determinadas condições sobre as arestas sejam satisfeitas. Sendo assim, seja um grafo $G = (S,E)$, onde os vértices correspondem aos subelementos mutantes e as arestas indicam se eles se intersectam, bem como a ausência delas indicam que não há interseção. Além disso, tome I_{s_i} a representação de cada subelemento mutante e a interseção de I_{s_i} e I_{s_j} indica a relação entre um subelemento s_i e um subelemento s_j. Veja na Figura 1.1 a representação do problema descrito acima.

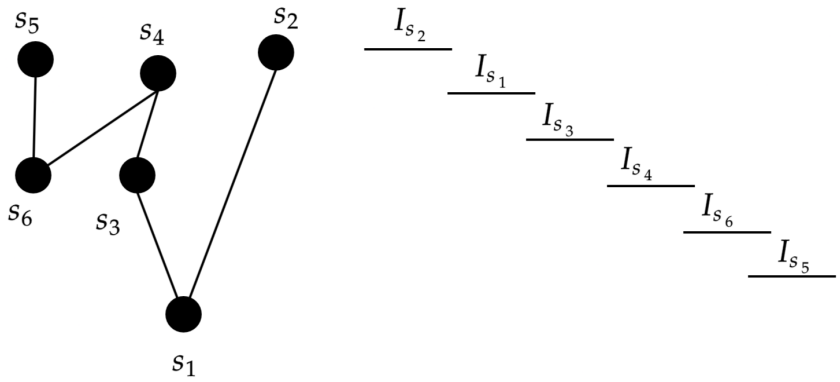

Figura 1.1: *Exemplo de representação da relação entre subelementos.*

Além da aplicação supracitada, os grafos de intervalo também possuem aplicações na engenharia, como em desenhos de circuitos [22], na medicina, como em diagnóstico médico [19], mapeamento do DNA [15] e genética [3].

Objetivo e Trabalhos Relacionados

Dada a importância e aplicabilidade dos grafos de intervalo em problemas no mundo real, esta monografia possui como objetivo explorar algumas propriedades de grafos intervalo. A compreensão de certas propriedades relacionadas a grafos, em especial os grafos de intervalo, pode ser um desafio para iniciantes da área, assim temos também como objetivo apresentar essas informações de maneira mais didática. Para isso, apresentaremos propriedades, algoritmos e reescritas mais simples de demonstrações existentes na literatura, acompanhadas de exemplos ilustrativos.

Um outro aspecto do presente estudo compreende a solução eficiente do problema do conjunto independente máximo em grafos de intervalo. Além disso, o trabalho

inclui a implementação em linguagem *python* do algoritmo utilizado para resolver o problema. A seguir, abordaremos alguns aspectos sobre conjuntos independentes.

Um *conjunto independente* de um grafo G é um subconjunto I de vértices de G no qual não há relação de adjacência entre os vértices de I. O conjunto I será *máximo* se for o maior conjunto independente possível de G. Dado um grafo G, um problema clássico é encontrar um conjunto independente máximo em G, onde a cardinalidade deste conjunto é denotada por $\alpha(G)$.

O conceito de conjunto independente máximo é bastante relevante quando aplicado em grafos de intervalo, pois nos permite modelar e resolver problemas de forma eficiente, ou seja, de maneira que o tempo e os recursos necessários para encontrar a solução sejam reduzidos, muitas vezes utilizando algoritmos com baixa complexidade computacional.

Uma aplicação prática relevante que relaciona conjuntos independentes e grafos de intervalo está no problema de escalonamento de tarefas [21], no qual cada tarefa é representada por um intervalo na reta real e sua duração é expressa pelo comprimento do intervalo. Nesse contexto, o problema de encontrar um conjunto independente máximo em um grafo de intervalo equivale a determinar o maior subconjunto de tarefas que podem ser realizadas sem conflitos temporais, ou seja, cujos intervalos não apresentem interseção.

Podemos modelar este problema da seguinte forma: cada vértice do grafo corresponde a uma tarefa, e existe uma aresta entre dois vértices u e v se, e somente se, os intervalos associados às tarefas T_u e T_v se sobrepõem, ou seja, $T_u \cap T_v \neq \emptyset$. O objetivo é maximizar $|S|$, onde $S \subseteq V$ é um conjunto independente, isto é, para todo par $u, v \in S$, não existe aresta $(u, v) \in E$. Quanto mais vizinhos um determinado vértice v possui, maior será o tempo associado à tarefa representada por v. Resolver esse problema no contexto de escalonamento permite selecionar o maior conjunto de tarefas que podem ser realizadas sem sobreposição temporal, contribuindo para a organização eficiente de processos e a otimização do uso de recursos disponíveis. A seguir, veja na Figura 1.2 o exemplo descrito acima.

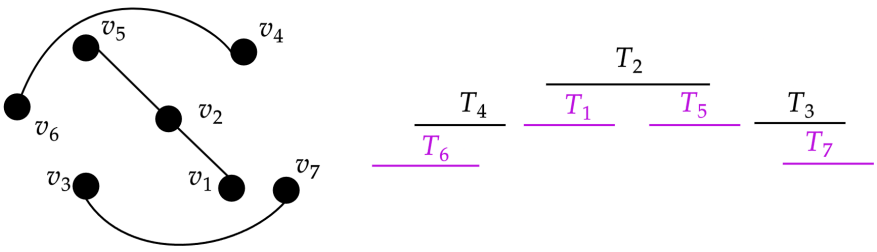

Figura 1.2: *Exemplo de escalonamento de tarefas.*

No exemplo supracitado, é possível realizar no máximo 4 tarefas sem que hajam conflitos temporais. Um exemplo de conjunto independente máximo é composto pelas tarefas T_1, T_5, T_6 e T_7, pois elas não se intersectam em nenhum ponto da reta real. Note

que outros conjuntos independentes maximais também podem ser formados, como por exemplo as tarefas T_1, T_4 e T_7, que, assim como o conjunto anterior, atendem à condição de não haver interseção temporal entre as tarefas, no entanto, não formam o maior conjunto de tarefas possível.

Embora encontrar o conjunto independente máximo seja bastante útil em diversos contextos, como em aprendizado de máquina [20] e rotulagem de mapas [2], a complexidade computacional do problema de decisão relacionado é NP-completa para grafos arbitrários [11]. Devido à complexidade de se determinar um conjunto independente máximo em grafos, os pesquisadores têm focado em restringir as características desses grafos, buscando determinar $\alpha(G)$ quando G pertence a subclasses específicas, o que pode ter complexidade diferente dependendo da subclasse. Por exemplo, existe um algoritmo que determina $\alpha(G)$ em tempo linear quando G é uma árvore [16].

Uma generalização de grafos de intervalo envolve a transição do modelo de intervalos na reta real para um modelo em que consideramos arcos em uma circunferência, resultando no que chamamos de *modelo de arcos*. Neste contexto, os grafos de *arco circular* surgem como uma superclasse dos grafos de intervalo, onde os *arcos* desempenham um papel análogo aos intervalos, e a *circunferência* substitui a reta real. Para essa superclasse, mostrou-se que, dado um grafo de arco circular com n vértices e m arestas, é possível resolver o problema do conjunto independente máximo em tempo linear, $O(n+m)$ [14]. Como os grafos de intervalo são um caso particular dos grafos de arco circular, esse resultado implica que determinar $\alpha(G)$ também possui complexidade linear para G um grafo de intervalo.

Uma classe relacionada, superclasse de grafos cordais, é a conhecida como grafos fracamente cordais. Um grafo é dito *fracamente cordal* quando não contém um ciclo ou o grafo complementar de um ciclo com pelo menos 5 vértices como subgrafo induzido. Para estes grafos, com n vértices e m arestas, é possível determinar seu maior conjunto independente através de um algoritmo de tempo $O(mn)$ [13]. É importante mencionar que todo grafo de intervalo é cordal, propriedade que será demonstrada em 3.1.

Vimos anteriormente algumas classes de grafos para as quais o problema do conjunto independente pode ser determinado de forma eficiente. No entanto, a título de exemplo, determinar o conjunto independente máximo de um grafo planar é categorizado como um problema NP-difícil. Um *grafo planar* é um grafo que pode ser imerso no plano de tal forma que suas arestas não se cruzem. Vale destacar que, mesmo ao considerarmos grafos planares com grau máximo igual a três, o problema permanece sendo NP-difícil [11]. Outro exemplo em que determinar o conjunto independente máximo é NP-difícil envolve os grafos *k-regulares*, para $k \geq 3$ fixo [18]. Um grafo k-regular é aquele em que todos os vértices possuem grau k.

Organização do Texto

Esta monografia reúne os resultados decorrentes de um trabalho de iniciação científica realizada pela autora, que abordou a classe dos grafos de intervalo. É relevante mencionar que um resultado parcial desse trabalho, referente a um algoritmo para geração de modelo de intervalo, resultou em um artigo aceito para apresentação na 12ª Escola Regional de Informática de Goiás (ERI-GO 2024) como artigo completo a ser publicado nos anais do evento.

Esta monografia está dividida em mais três capítulos: o intitulado "Conceitos Básicos" (Capítulo 2), aborda definições detalhadas sobre grafos, grafos de intervalo, conjunto independente máximo e notações assintóticas. Em "Resultados" (Capítulo 3), apresenta demonstrações e algoritmos sobre grafos de intervalo. Finalmente, na "Conclusão" (Capítulo 4) são apresentados comentários finais e trabalhos futuros. Ademais, no "Apêndice A", segue nossa implementação de código de programa em linguagem *python*.

CAPÍTULO 2

Conceitos Básicos

Nesta seção, apresentaremos uma fundamentação teórica necessária para o entendimento dos teoremas apresentados no Capítulo 3. Conceitos básicos sobre grafos presentes nesta monografia podem ser encontrados em Bondy e Murty [5], Diestel [9] e De Mello et al. [8]. Notações e definições específicas sobre grafos de intervalo foram retiradas de Oliveira [21] e Golumbic [12]. Fundamentos de complexidade computacional foram extraídos de Cormen et al. [7]. A Seção 2.1 apresenta conceitos básicos sobre grafos e outros fundamentos necessários para o entendimento de algumas propriedades dos grafos. Na Seção 2.2, apresentamos algumas características fundamentais dos grafos de intervalo, as quais serão necessárias para a compreensão dos resultados apresentados no Capítulo 3. Na Seção 2.3, exibimos algumas definições básicas para a interpretação da complexidade dos algoritmos apresentados na Seção 3.2.

2.1 Sobre Grafos e outros Fundamentos

Sejam A e B dois conjuntos. Dizemos que A e B são *disjuntos*, se $A \cap B = \emptyset$ e *sobrepostos*, caso contrário.

Um grafo simples $G = (V(G), E(G))$ é formado por um conjunto $V(G)$, cujos elementos são os vértices, e um conjunto $E(G)$, cujos elementos são as arestas. O conjunto $E(G)$ é disjunto de $V(G)$ e satisfaz a condição $E(G) \subseteq [V(G)]^2$.

O conjunto $[V(G)]^2$ é definido como o conjunto de todos os pares de vértices distintos em $V(G)$, isto é:

$$[V(G)]^2 = \{\{u,v\} \mid u,v \in V(G) \text{ e } u \neq v\}.$$

Um *subgrafo* de um grafo $G = (V(G), E(G))$ é um grafo $G' = (V'(G'), E'(G'))$ tal que $V'(G') \subseteq V(G)$ e $E'(G') \subseteq E(G)$. Se G' contém todas as arestas $(u,v) \in E(G)$ com $u,v \in V'(G')$, então G' é chamado de subgrafo induzido por $V'(G')$. , então G' é um *subgrafo induzido* de G, denotado por $G[V']$. Dizemos que V' *induz* ou gera G' em G. Um exemplo de grafo G com $V(G) = \{v_1, v_2, \ldots, v_7\}$ e $E(G) =$

$\{(v_1,v_2),(v_1,v_6),(v_2,v_5),(v_3,v_4),(v_3,v_6),(v_4,v_5),(v_5,v_7),(v_6,v_7)\}$ pode ser visto na Figura 2.1.

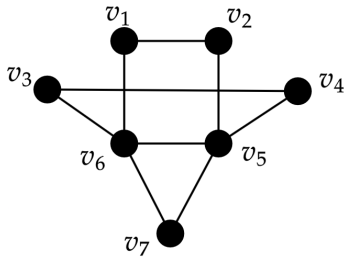

Figura 2.1: *Grafo G.*

É importante ressaltar que um grafo simples não possui *laços* (arestas que contenham vértices não-distintos) e nem *multi-arestas* (arestas idênticas), por definição. Utilizaremos apenas a notação *grafo* quando formos nos referir a um grafo simples. Seja G um grafo e considere $(u,v) \in E(G)$. Diremos que u e v são *adjacentes* caso estejam conectados pela aresta (u,v). Ainda, considere que a aresta (u,v) é dita *incidente* aos vértices u e v se conecta ambos. A *vizinhança aberta* (ou, simplesmente, *vizinhança*) de $v \in V(G)$ é o conjunto de todos os vértices adjacentes a esse vértice, excluindo o vértice v, ou seja, $N_G(v) = \{w \mid (v,w) \in E(G)\}$. A *vizinhança fechada* de v é definida pelo conjunto $N[v] = N(v) \cup \{v\}$. Denominamos *grau* $d_G(v)$ de um vértice $v \in V(G)$ o número de arestas incidentes a este vértice. O *grau máximo* de um grafo G, denotado por $\Delta(G)$, é o maior grau de um vértice em G.

Quando o grafo G for claro pelo contexto os subscritos das notações de vizinhança e grau serão omitidos.

Um *caminho em um grafo* $G = (V(G),E(G))$ é uma sequência de vértices $v_1,v_2,\ldots,v_k$ tal que, para cada i (onde $1 \leq i < k$), existe uma aresta $(v_i,v_{i+1}) \in E(G)$. Um *grafo caminho* P_n é um grafo com $V(P_n) = \{v_1,v_2,\ldots,v_n\}$ e $E(P_n) = \{(v_i,v_{i+1}) \mid 1 \leq i < n\}$. Um *grafo ciclo* C_n é definido a partir de um grafo caminho com n vértices, com a adição da aresta (v_1,v_n). Um *grafo completo*, denotado por K_n, é um grafo com n vértices no qual há uma aresta entre cada par de vértices distintos. Uma *árvore* é um grafo conexo e acíclico tal que $|E| = |V| - 1$.

Um grafo é considerado *conexo* se existe um caminho entre qualquer par de vértices, caso contrário ele é chamado *desconexo*.

Veja na Figura 2.2 um exemplo de um grafo G conexo e um grafo D desconexo.

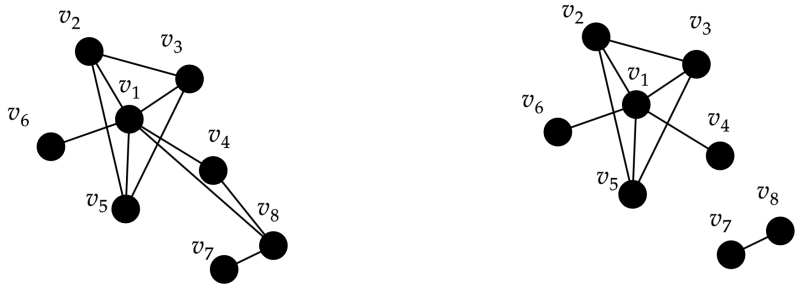

Figura 2.2: *Exemplo de grafo conexo e grafo desconexo.*

O *complemento* (ou *grafo complementar*) de um grafo G é um grafo $\overline{G}$ nos mesmos vértices de G tais que dois vértices u e v de $\overline{G}$ são adjacentes se e somente se u e v não são adjacentes em G. Veja na Figura 2.3 um exemplo de um grafo G e seu grafo complementar $\overline{G}$.

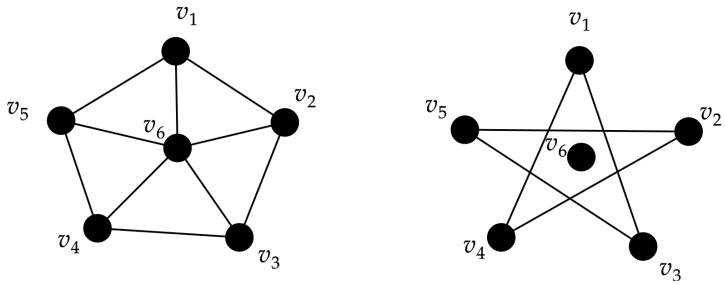

Figura 2.3: *Exemplo de complemento* $\overline{G}$.

Um conjunto M é um *maximal* em relação a uma determinada propriedade P se não existe um conjunto M' com a propriedade P tal que $M \subset M'$. Um conjunto M é dito *máximo* se não existe conjunto M' tal que $|M| < |M'|$.

Seja G um grafo. Uma *clique* C de G é um subconjunto de $V(G)$ tal que $G[C]$ é um grafo completo. A *cardinalidade* da maior clique em G é o *número de clique* de G e é denotada por $\omega(G)$. Um *conjunto independente* I de G é um subconjunto de $V(G)$ tal que $G[C]$ é um grafo sem arestas. A cardinalidade do maior conjunto independente de G é o *número de independência* de G e é denotada por $\alpha(G)$.

Seja $r \geq 2$ um inteiro. Diremos que um grafo G é *r-partido* se $V(G)$ admitir uma partição em r conjuntos independentes tal que cada aresta possui seus extremos em partes diferentes. Chamamos o grafo 2-partido de *bipartido*. Um grafo *bipartido completo* é composto por dois conjuntos de vértices, X e Y, onde cada vértice em X está conectado a todos os vértices em Y e vice-versa. Se o conjunto X tem m vértices e o conjunto Y tem n vértices, o grafo bipartido completo é denotado como $K_{m,n}$. Um *grafo estrela* S_n é um tipo de grafo bipartido, onde um dos conjuntos da partição contém um único vértice central v_c, e o outro conjunto contém os demais vértices, os quais são todos adjacentes ao vértice central, mas não entre si.

Um *grafo cordal* é aquele em que todo ciclo C_n, com $n \geq 4$ possui uma *corda*, isto é, uma aresta ligando dois vértices não consecutivos do ciclo. Um vértice v é um vértice *simplicial* se o grafo induzido por $N[v]$ for uma clique.

Um *esquema (ou ordem) de eliminação perfeita (EEP)* de um grafo G é uma sequência dos vértices de G, $\sigma = (v_1, v_2, \ldots, v_n)$, onde cada vértice v_i é um vértice simplicial do subgrafo induzido por $\{v_i, v_{i+1}, \ldots, v_n\}$. Isto é, para cada vértice v, sua vizinhança $N(v)$, que ocorre depois de v na ordenação, induz um grafo completo. Todo grafo cordal admite um esquema de eliminação perfeita.

Seja G um grafo. Três vértices distintos e dois a dois não adjacentes $u, v, w \in V(G)$ formam uma *tripla asteroidal* quando, para quaisquer dois deles, existe um caminho que os liga e não passa pela vizinhança do terceiro. Um grafo que não possui triplas asteroidais é chamado de grafo *sem tripla asteroidal* (STA). A seguir, considere o grafo C_6 composto por $V(C_6) = \{v_1, v_2, \ldots, v_6\}$ e $E(C_6) = \{\{(v_i, v_{i+1}) \mid 1 \leq i < 6\} \cup \{(v_6, v_1)\}$. A título de exemplo, os três vértices v_1, v_3, v_5 formam uma tripla asteroidal em C_6, uma vez que é possível ter um caminho entre quaisquer dois vértices v_i e v_j que não passe por $N(v_k)$, para todo $i, j, k \in \{1, 3, 5\}$ distintos. Veja na Figura 2.4 o exemplo descrito acima, onde os vértices roxos representam a tripla asteroidal.

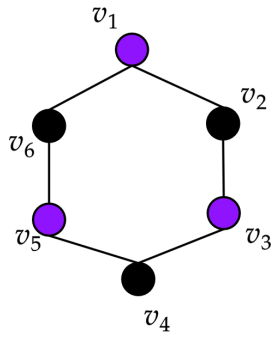

Figura 2.4: *Em roxo há um exemplo de tripla asteroidal no grafo C_6.*

Dois grafos G e H são chamados de *isomorfos* se existe uma correspondência bijetiva entre os conjuntos de vértices de G e H. Formalmente, dois grafos $G = (V(G), E(G))$ e $H = (V(H), E(H))$ são isomorfos se existe uma função bijetiva $f : V(G) \to V(H)$ tal que, para quaisquer dois vértices $u, v \in V(G)$, temos que

$$(u, v) \in E(G) \iff (f(u), f(v)) \in E(H).$$

Na Figura 2.5, é apresentado um exemplo de um grafo G e um grafo H, onde G e H são isomorfos. A função f que estabelece o isomorfismo entre os dois grafos é dada por:

$$f(v_1) = v_a, \quad f(v_2) = v_b, \quad f(v_3) = v_c, \quad f(v_4) = v_d.$$

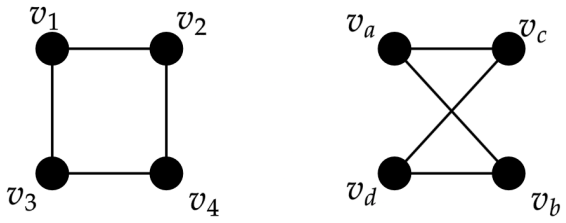

Figura 2.5: *Exemplo de grafos isomorfos.*

Um grafo G é *livre* de um grafo H se e somente se G não contém nenhum subgrafo induzido que seja isomorfo a H. Seja $\mathcal{C}$ uma classe de grafos. Se para todo grafo $G \in \mathcal{C}$, G é livre de um determinado grafo H, dizemos que H é grafo *proibido* para a classe $\mathcal{C}$.

Uma *k-coloração* de um grafo G é uma atribuição de k cores aos vértices de G de forma que vértices adjacentes recebam cores diferentes, isto é, uma k-coloração é uma função $f : V(G) \to \{1,2,\ldots,k\}$ tal que $f(u) \neq f(v)$ para todo par de vértices adjacentes u e v em G.

2.2 Sobre Grafos de Intervalo

Um *intervalo real fechado* entre dois pontos a e b (onde $a \leq b$) é o conjunto de todos os números reais x tais que $a \leq x \leq b$. Esse intervalo inclui os pontos extremos a e b. É denotado por $[a,b]$. Ou seja:

$$[a,b] = \{x \in \mathbb{R} \mid a \leq x \leq b\}.$$

Um *intervalo real aberto* entre dois pontos a e b (onde $a < b$) é o conjunto de todos os números reais x tais que $a < x < b$. Esse intervalo não inclui os pontos extremos a e b. É denotado por (a,b). Logo:

$$(a,b) = \{x \in \mathbb{R} \mid a < x < b\}.$$

Seja $\mathcal{F}$ uma família de conjuntos. O *grafo de interseção* de $\mathcal{F}$ é o grafo obtido representando-se por um vértice distinto cada conjunto de $\mathcal{F}$ e fazendo dois de tais vértices adjacentes precisamente quando os conjuntos correspondentes têm interseção não-vazia. Note que, dada uma família $\mathcal{F}$, o grafo de interseção associado é bem definido, porém o contrário não é verdade: um mesmo grafo pode ser o grafo de interseção de diferentes famílias. Se G é o grafo de interseção de uma família $\mathcal{F}$, dizemos que $\mathcal{F}$ é um *modelo* de G.

Um grafo G é um *grafo de intervalo* se G é o grafo de interseção de uma família $\mathcal{R}$ de intervalos da reta real. Em outras palavras, um grafo G é um grafo de intervalo

precisamente quando existir uma correspondência entre $V(G)$ e uma família de intervalos $\mathcal{R} = \{I_v \mid v \in V(G)\}$ da reta real tal que, para todo $u, w \in V(G)$ distintos, $I_u \cap I_w \neq \emptyset$ se e somente se $(u,w) \in E(G)$. Chamamos $\mathcal{R}$ um *modelo de intervalo* de G. Um exemplo de grafo de intervalo pode ser visto na Figura 2.6.

Considere o grafo G com $V(G) = \{v_1, v_2, \ldots, v_9\}$ e $E(G) = \{(v_1,v_3),(v_1,v_9),(v_1,v_8),(v_1,v_7),(v_1,v_4),(v_1,v_5),(v_5,v_6),(v_4,v_2)\}$. Ilustraremos a criação de um modelo de intervalo para G. Precisamos assegurar que os intervalos correspondentes aos vértices adjacentes tenham uma interseção. Portanto, criaremos uma interseção a todos vértices adjacentes a v_1. Temos $N(v_1) = \{v_3, v_4, v_5, v_7, v_8, v_9\}$. Note que $v_2, v_6 \notin N(v_1)$, portanto, estes não irão sobrepor o intervalo referente à v_1, mas vão sobrepor os intervalos referentes a v_4 e v_5.Veja na Figura 2.6:

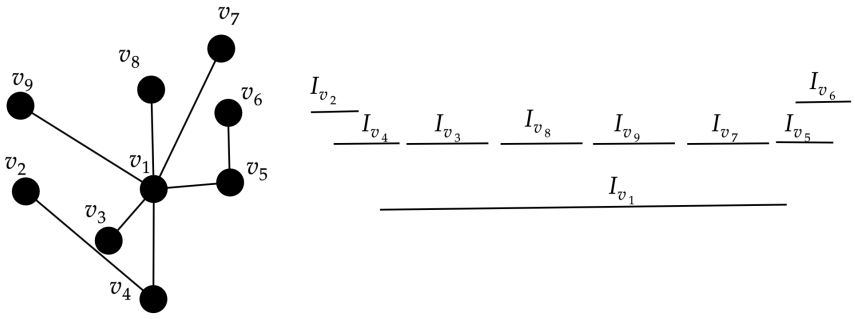

Figura 2.6: *Exemplo de um possível modelo de intervalo para o grafo G.*

Assumimos que todos os extremos de intervalo são distintos e denotamos os extremos esquerdo e direito de um intervalo I_v respectivamente por $\ell(I_v)$ e $r(I_v)$. Para qualquer modelo de intervalo $\mathcal{R}$, assumiremos que $\ell(I_v) > 0$, para todo $I_v \in \mathcal{R}$. O *tamanho* (ou *comprimento*) de I_v é representado por $|I_v|$ e é calculado através da seguinte fórmula: $|I_v| = r(I_v) - \ell(I_v)$. Um intervalo I_v é *posterior* a um intervalo I_u se $\ell(I_v) > r(I_u)$. Por conveniência, dado um intervalo I_v de um modelo de intervalo e o vértice correspondente v, podemos usar indistintamente I_v ou v quando o contexto não criar ambiguidades. Similarmente, para um dado índice i, podemos usar I_i para representar o intervalo I_{v_i}. Um grafo é de *intervalo unitário* se existir um modelo de intervalo deste grafo tal que todos os intervalos possuam tamanho unitário, isto é, $|I_v| = 1$ para todo $v \in V(G)$.

Definimos, a seguir, as operações de expansão, compressão e translação de intervalos. Sejam $\mathcal{R}$ um modelo de intervalo e $I = [a,b] \in \mathcal{R}$.

- A *expansão* de I é obtida por $(\mathcal{R} \setminus I) \cup I'$, onde $I' = [a-d, b+e]$, com $d,e > 0$ e $d,e \in \mathbb{R} \setminus I$. Intuitivamente, essa operação aumenta o comprimento do intervalo.
- A *compressão* de I é obtida por $(\mathcal{R} \setminus I) \cup I'$, onde $I' = [a+d, b-e]$, com $d,e > 0$, $d,e \in I$ e $d < e$. A compressão reduz o comprimento do intervalo dentro dos limites originais.

- A *translação à direita* de I é obtida por $(\mathcal{R} \setminus I) \cup I'$, onde $I' = [a+c, b+c]$ para $c \in \mathbb{R}$, $c > 0$. A *translação à esquerda* de I é obtida por $(\mathcal{R} \setminus I) \cup I'$, onde $I' = [a-c, b-c]$ para $c \in \mathbb{R}$, $c > 0$. Intuitivamente, as translações consistem em deslocar o intervalo para uma nova posição na reta real, sem alterar seu comprimento.

2.3 Sobre Complexidade Computacional

A *complexidade computacional* é um ramo da teoria da computação que se concentra em classificar problemas computacionais de acordo com sua dificuldade e relacionar essas classes entre si. Problemas de complexidade *polinomial* podem ser resolvidos por algoritmos que rodam em tempo no máximo $c \cdot n^k$, para constantes reais positivas c, k e n o tamanho da entrada. Se $k = 1$, diremos que o problema tem complexidade *linear*. Problemas de complexidade polinomial são considerados *eficientes*, pois podem ser resolvidos de forma prática na maioria dos casos. No entanto, há problemas classificados como *NP-completos*, que são considerados difíceis de serem resolvidos computacionalmente, pois ainda não se conhece um algoritmo eficiente que possa resolvê-los.

A notação O assintoticamente limita uma função $f(n)$ superiormente. Para uma função $g(n)$, dizemos que $f(n) = O(g(n))$ (comumente conhecida por notação "Big-Oh") se existem constantes reais positivas c e n_0 tais que:

$$0 \leq f(n) \leq c \cdot g(n) \quad \text{para todo} \quad n \geq n_0.$$

A notação O é utilizada para fornecer um limite superior para uma função dentro de um fator constante. Essa notação é frequentemente usada para descrever o tempo de execução de algoritmos, analisando sua estrutura geral.

A notação $\Omega(g(n))$ fornece um limite assintótico inferior para uma função $f(n)$. Formalmente, dizemos que $f(n) = \Omega(g(n))$ se existem constantes positivas c e n_0 tais que:

$$f(n) \geq c \cdot g(n) \geq 0 \quad \text{para todo} \quad n \geq n_0.$$

Isso significa que $f(n)$ cresce, no mínimo, na mesma taxa que $g(n)$ a partir de um certo valor de n.

Finalmente, a notação $\Theta(g(n))$ é usada quando a função $f(n)$ tem tanto um limite superior quanto um limite inferior assintótico em torno de $g(n)$. Formalmente, dizemos que $f(n) = \Theta(g(n))$ se existem constantes positivas c_1, c_2, n_0 tais que:

$$0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \quad \text{para todo} \quad n \geq n_0.$$

Ou seja, a notação Θ descreve uma função que cresce exatamente na mesma taxa que $g(n)$, dentro de fatores constantes.

CAPÍTULO 3

Resultados

Este capítulo se destina à apresentação dos nossos resultados, bem como a apresentação de provas encontradas na literatura sobre grafos de intervalo de maneira mais didática. Está organizado nas seguintes seções. A Seção 3.1 apresenta caracterizações de grafos de intervalo que serviram como base para o desenvolvimento de um algoritmo polinomial para o reconhecimento destas, que é apresentado na Seção 3.2. Na Seção 3.2, apresentamos um algoritmo que, dado um grafo de intervalo, gera um modelo de interseção para G. Na Seção 3.3, mostramos outra característica dos grafos de intervalo, onde o número de independência $\alpha(G)$ de um grafo de intervalo pode ser determinado em tempo polinomial. Finalmente, na Seção 3.4 mostramos uma observação sobre grafos de intervalo unitário serem livres de $K_{1,3}$.

3.1 Caracterizações de Grafos de Intervalo

Iniciamos a seção investigando os grafos de intervalo no que diz respeito a serem cordais, isto é não possuem ciclos induzidos com 4 ou mais vértices. Usamos o Teorema 3.1 como ponto de partida, onde mostramos que o grafo C_n, quando $n \geq 4$, não possui modelo de intervalo.

Teorema 3.1 *Seja $n \geq 4$. Então o grafo C_n não é um grafo de intervalo.*

Prova. Suponha, por contradição, que exista um modelo de intervalo $\mathcal{R}$ para o grafo C_n, com $n \geq 4$, $V(C_n) = \{v_1, v_2, v_3, \ldots, v_{n-1}, v_n\}$ e $E(C_n) = \{(v_i, v_{i+1}) \mid 1 \leq i < n\} \cup \{(v_n, v_1)\}$. Veja na Figura 3.1 uma representação do grafo descrito acima.

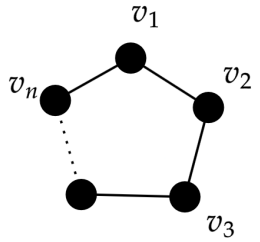

Figura 3.1: *Exemplo de grafo C_n, onde $n \geq 4$.*

Ao tentarmos criar uma representação de grafo de intervalo para o grafo C_n, devemos associar a v_i, para todo $1 \leq i \leq n$, um intervalo I_{v_i} sobre a reta real, de modo que dois intervalos I_{v_i} e I_{v_j} se sobreponham se, e somente se, v_i e v_j são adjacentes em C_n. Da definição de grafo ciclo, temos que $(v_i, v_{i+1}) \in E(C_n)$, para todo $1 \leq i \leq n-1$. Logo, no modelo de intervalo temos $I_{v_i} \cap I_{v_{i+1}} \neq \emptyset$. Observe que $(v_i, v_{i+2}) \notin E(C_n)$, para todo $1 \leq i \leq n-1$, assim temos que $I_{v_i} \cap I_{v_{i+2}} = \emptyset$. Para que isso ocorra, podemos dispor consecutivamente, sem perda de generalidade, $I_{v_{i+1}}$ à direita de I_{v_i}, para todo $1 \leq i \leq n-1$. Porém, ainda da definição de grafo ciclo, temos que $(v_n, v_1) \in E(C_n)$, logo no modelo de intervalo temos $I_{v_n} \cap I_{v_1} \neq \emptyset$. Da disposição realizada, temos que $I_{v_n} \cap I_{v_2} \neq \emptyset$, uma contradição, pois $(v_2, v_n) \notin E(C_n)$. Entretanto, note que v_n também é adjacente a v_{n-1}, ou seja, $I_{v_n} \cap I_{v_{n-1}} \neq \emptyset$. Porém, chegamos a uma absurdo, uma vez que não conseguimos dispor I_{v_n} entre I_{v_1} e $I_{v_{n-1}}$ de modo que as relações de adjacências de C_n sejam respeitadas, pois a reta real não permite uma disposição linear que satisfaça todas as relações de adjacência para C_n quando $n \geq 4$. Portanto, não existe representação de um modelo de intervalo para C_n quando $n \geq 4$. □

Veja na Figura 3.2 uma tentativa de representação de um modelo de intervalo para C_5, isto é, quando $n \geq 4$.

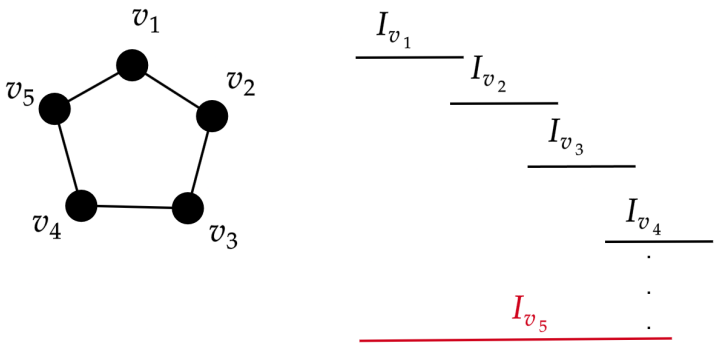

Figura 3.2: *Exemplo de tentativa de representação de um modelo de intervalo para C_5*

Podemos perceber que, caso um grafo G possua um ciclo com $n \geq 4$ vértices como subgrafo induzido, a ideia do Teorema 3.1 se aplica diretamente para tal subgrafo, logo podemos concluir o seguinte corolário.

Corolário 3.2 *Seja G um grafo. Se G é de intervalo, então G é cordal.*

Relembramos também que três vértices distintos e dois a dois não adjacentes formam uma *tripla asteroidal* quando, para quaisquer dois deles, existe um caminho que os liga e não passa pela vizinhança do terceiro. No Teorema 3.3, provaremos que, caso G tenha uma tripla asteroidal, então G não é um grafo de intervalo.

Teorema 3.3 *Seja G um grafo. Se G é de intervalo, então G não possui tripla asteroidal.*

Prova. Suponha, por contradição, que G seja um grafo de intervalo que possui tripla asteroidal, denotada por $\{v_1, v_2, v_3\} \subseteq V(G)$. Pela definição de tripla asteroidal, temos que, para todo par $i, j \in \{1,2,3\}$, existe um caminho $P_{i,j}$ que conecta v_i e v_j sem passar pela vizinhança de v_k, com $k \in \{1,2,3\} \setminus \{i, j\}$.

No modelo de intervalo de G, podemos assumir que I_{v_1}, I_{v_2} e I_{v_3} estão dispostos em sequência, com $I_{v_1} \cap I_{v_2} = \emptyset$ e $I_{v_2} \cap I_{v_3} = \emptyset$, já que $\{v_1, v_2, v_3\}$ é um conjunto independente. Como o caminho $P_{1,3} : u_1, u_2, \ldots, u_\ell$ conecta $u_1 = v_1$ e $u_\ell = v_3$, para algum $\ell \geq 3$, temos que os intervalos I_{u_i}, para todo $1 \leq i \leq \ell$, são dispostos em sequência. Porém, para algum $i \in \{1, \ldots, \ell\}$, $I_{v_2} \cap I_{v_i} = \emptyset$, já que I_{v_2} se situa entre $I_{u_1=v_1}$ e $I_{u_\ell=v_3}$, uma contradição.

Portanto, em um grafo de intervalo, a disposição linear dos intervalos impede a formação de caminhos alternativos que evitem a vizinhança de um terceiro vértice, impossibilitando a existência de uma tripla asteroidal. A seguir, na Figura 3.3, temos uma tentativa de montar um modelo de intervalo para um grafo que possui tripla asteroidal.

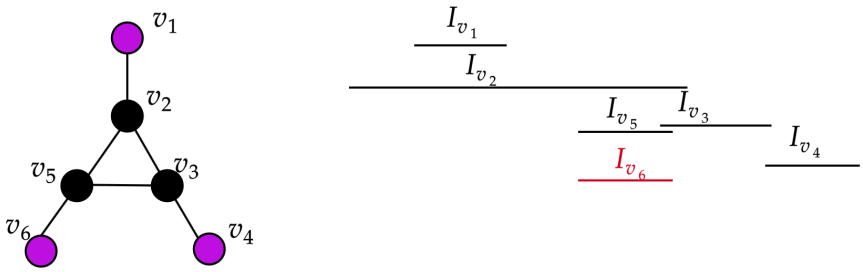

Figura 3.3: *Exemplo de tentativa de modelo de intervalo para um grafo que possui tripla asteroidal.*

□

Do Corolário 3.2 e do Teorema 3.3 sabemos que se G é um grafo de intervalo, então G é cordal e G não possui triplas asteroidais. A demonstração da recíproca, entretanto, é consideravelmente complicada. Por isso optamos apenas por enunciar a caracterização completa de grafos de intervalo no Teorema 3.4, um célebre resultado apresentado em 1962 por Lekkeikerker e Boland [17].

Teorema 3.4 *[17] Um grafo G é um grafo de intervalo se, e somente se, G é cordal e não possui triplas asteroidais.*

3.2 Geração de um Modelo de Intervalo

Nesta seção, traremos algoritmos necessários para a criação de um modelo de intervalo para um grafo de intervalo G. Primeiro, mostramos como decidir em tempo

polinomial se um grafo de entrada qualquer G é um grafo de intervalo, através do Algoritmo 1 cuja prova de corretude segue na Proposição 3.6.

Algoritmo 1: ÉGRAFODEINTERVALO(G)

Entrada: Grafo $G = (V(G), E(G))$
Saída: `SIM`, se G é grafo de intervalo, ou `NÃO`, caso contrário

1 **para cada** *ciclo induzido* $C \subseteq V(G)$ *;* `// Checa se` G `é cordal.`
2 **faça**
3 **se** $|C| \geq 4$ **então**
4 **retorna** `NÃO`
5 **fim**
6 **fim**
7 **para cada** *tripla de vertices* $u, v, w \in V(G)$*;* `// Checa se` G `não possui triplas asteroidais.`
8 **faça**
9 **se** *existe um caminho entre u e v em* $G - N(w)$
10 ***e*** *existe um caminho entre u e w em* $G - N(v)$
11 ***e*** *existe um caminho entre v e w em* $G - N(u)$
12 **então**
13 **retorna** `NÃO`
14 **fim**
15 **fim**
16 **retorna** `SIM`

Proposição 3.5 *Seja G um grafo com n vértices. O Algoritmo 1 decide se G é um grafo de intervalo em tempo* $O(n^3)$.

Prova. Considere que o grafo G de entrada para o Algoritmo 1 possua n vértices e m arestas. As linhas 1 a 6 checam se G possui ciclos induzidos com 4 ou mais vértices. Tal checagem pode ser executada com uma modificação do algoritmo de busca em profundidade, que executa em $O(n+m)$.

As linhas 7 a 15 testam se G possui uma tripla asteroidal. Como o número de vértices de G é n, há um total de n^3 triplas possíveis. Para cada tripla, é necessário calcular $G - N(w)$ e, em seguida, testar a existência de caminhos entre os pares de vértices especificados. O cálculo de $G - N(w)$ pode ser realizado em $O(n+m)$, e o teste de conectividade com algoritmos como busca em largura (BFS) ou profundidade (DFS) também possui complexidade $O(n+m)$. Como esse processo é repetido para todas as triplas, a complexidade total dessas operações é $O(n^3)$.

Por fim, na linha 16 o algoritmo simplesmente retorna que de fato G é um grafo de intervalo. A complexidade total de tempo de execução do Algoritmo 1 é então da

ordem de $O(n^3)$, que é polinomial. A corretude do Algoritmo 1 segue diretamente da caracterização apresentada no Teorema 3.4. □

Dado um grafo G, uma vez obtido o retorno `SIM` do Algoritmo 1, podemos prosseguir para a construção de um modelo de intervalo de G. Para tal, utilizaremos a busca em largura lexicográfica [12], discutida a seguir.

O Algoritmo de Busca em Largura Lexicográfica atribui rótulos inteiros aos vértices de um grafo de maneira ordenada, priorizando os vértices que possuem adjacências entre si. Inicialmente, o algoritmo rotula os vértices com $\emptyset$, o que significa que nenhum vértice tem prioridade de escolha inicial. Em seguida, o algoritmo processa os vértices em ordem decrescente: partimos de um valor i igual ao número de vértices do grafo e, a cada iteração, reduz i até 1. Sendo assim, o algoritmo cria uma ordem linear, onde cada vértice recebe um número a partir do maior valor disponível.

A cada passo, o algoritmo seleciona o vértice não numerado que possui o maior rótulo acumulado. Ao atribuirmos a v o número i, indicamos a posição desse vértice na ordem final. O algoritmo então atualiza os rótulos dos vértices adjacentes a v, incrementando o rótulo de cada vizinho não numerado com o valor de i, permitindo que os vértices mais próximos de v se tornem candidatos prioritários a serem rotulados nas próximas iterações. O procedimento completo segue no Algoritmo 2, extraído de Golumbic [12].

Algoritmo 2: LexBFS(G)

Entrada: Grafo $G = (V(G), E(G))$ com n vértices
Saída: Uma ordem σ dos vértices de G

1 **para cada** $v \in V(G)$ **faça**
2 $\quad r(v) \leftarrow \emptyset$; `// Inicializa o rótulo de cada vértice como vazio`
3 **fim**
4 **para** $i \leftarrow n$ **até** 1 **faça**
5 $\quad$ **Selecione:** Escolha um vértice v nao numerado com o maior rótulo $r(v)$;
6 $\quad \sigma(v) \leftarrow i$; `// Atribui o número` i `ao vértice` v
7 $\quad$ **Atualize: para cada** $w \in N(v)$ *não numerado* **faça**
8 $\quad\quad r(w) \leftarrow r(w) \cup \{i\}$; `// Adiciona` i `ao rótulo do vértice` w
9 $\quad$ **fim**
10 **fim**
11 **retorna** σ ; `// Retorna a ordem dos vértices`

Proposição 3.6 *Seja G um grafo com n vértices. O Algoritmo 2 retorna uma ordem σ de G em tempo $O(n^2)$.*

Prova. Seja G um grafo com n vértices, entrada para o Algoritmo 2. A linha 1 é executada para cada vértice de G, o que resulta em um tempo de execução $O(n)$. A linha 4 é

repetida de n até 1, o que totaliza $O(n)$ passos. A linha 5 requer encontrar um vértice não numerado com maior rótulo. Podemos encontrar tal vértice simplesmente fazendo uma busca linear em cada vértice, em tempo $O(n)$. A linha 6 é simplesmente uma atribuição, que usa tempo $O(1)$. A linha 7 atualiza rótulos dos vizinhos, logo requer $O(\Delta)$ passos, onde Δ é o grau máximo de G. Assim, o laço de repetição do bloco das linhas 4 a 10 tem complexidade $O(n^2)$. O passo 11 é apenas um retorno, que executa em tempo constante $O(1)$. A complexidade total então é a soma das complexidades de todos os passos, sendo da ordem de $O(n^2)$, que é polinomial. □

A partir do retorno do algoritmo LexBFS(G), σ define um esquema de eliminação perfeita para um grafo de intervalo G. A ordem do esquema de eliminação perfeita coincide com a sequência que será utilizada para posicionar cada intervalo na reta real. Veja na Figura 3.4 um exemplo de grafo e um de seus esquemas de eliminação perfeita:

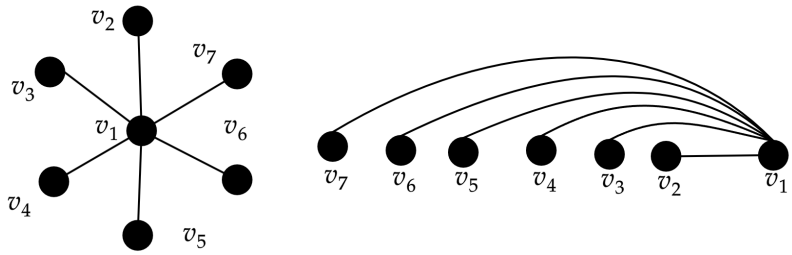

Figura 3.4: *Exemplo de grafo S_7 e seu EEP.*

Algoritmo 3: GERAMODELOINTERVALO(G)

Entrada: Grafo de intervalo $G = (V(G), E(G))$

Saída: Modelo de intervalo $\mathcal{R}$ de G

1 Seja $\sigma = (v_1, v_2, \ldots, v_n)$ um EEP retornado pelo algoritmo LEXBFS(G);
2 $I_n \leftarrow [0,1]$;
3 **para** $i \leftarrow n-1$ **até** 1 **faça**
4 $\quad G' \leftarrow G[v_{i+1}, \ldots, v_n]$;
5 $\quad \mathscr{I}_A \leftarrow \{j \in \{i+1, \ldots, n\} \mid v_j \in N_{G'}(v_i)\}$;
6 $\quad \mathscr{I}_{\overline{A}} \leftarrow \{i+1, \ldots, n\} \setminus \mathscr{I}_A$;
7 $\quad I_C \leftarrow \bigcap_{j \in \mathscr{I}_A} I_j$;
8 $\quad I_F \leftarrow I_C \setminus \left(\bigcup_{j \in \mathscr{I}_{\overline{A}}} I_j \right)$;
9 $\quad$ **se** $I_F = \emptyset$ **então**
10 $\quad\quad$ Sem criar novas sobreposições, expanda I_j, para todo $j \in \mathscr{I}_A$ ou comprima $I_{j'_i}$, para todo $j' \in \mathscr{I}_{\overline{A}}$ até $I_F \neq \emptyset$;
11 $\quad$ **fim**
12 $\quad$ Posicione I_{v_i} em alguma porção contígua mais à esquerda de I_F.
13 **fim**
14 **retorna** $\mathcal{R} = \{I_{v_1}, I_{v_2}, \ldots, I_{v_n}\}$; `// Retorna o modelo de intervalo`

Teorema 3.7 *Seja G um grafo de intervalo. O Algoritmo 3 quando executado sobre G produz corretamente um modelo de intervalo $\mathcal{R}$ para G.*

Prova. Seja $\sigma = (v_1, v_2, \ldots, v_n)$ um EEP retornado pelo algoritmo LEXBFS(G). O Algoritmo 3 percorre a sequência σ de v_n até v_1, para determinar o intervalo I_i correspondente ao vértice v_i. O passo base ocorre na Linha 2, onde I_n recebe $[0,1]$, o que é um modelo de intervalo válido para $G[v_n]$. Por hipótese de indução, considere que $\mathcal{R}' = \{I_{i+1}, \ldots, I_n\}$ seja um modelo de intervalo para $G' = G[v_{i+1}, \ldots, v_n]$. Vamos mostrar que $R = \{I_i, \ldots, I_n\}$ é um modelo de intervalo para $G[v_i, \ldots, v_n]$.

Na Linha 5 definimos o conjunto $\mathscr{I}_A$ de índices j tais que v_j é adjacente a v_i em G' e na Linha 6 definimos o conjunto $\mathscr{I}_{\overline{A}}$ de índices j tais que v_j não é adjacente a v_i em G'. A ideia é criar interseção entre I_{v_i} e I_{v_J} para todo $j \in \mathscr{I}_A$ e não criar interseção entre I_{v_i} e I_{v_j} para todo $j \in \mathscr{I}_{\overline{A}}$. Para tal, na Linha 7 é definido I_C pela interseção dos intervalos dos vizinhos de v_i como um intervalo candidato a posicionar I_{V_i}. Sabemos que isso é possível, já que no EEP σ, o conjunto de vizinhos de v_i em G' forma uma clique. Na Linha 8, obtemos um intervalo final I_F a partir do intervalo candidato I_C, onde são removidas as porções dos intervalos dos vértices que v_i não deve intersectar. Observe que I_F pode ser vazio. Assim, na Linha 10 ajustamos o modelo expandindo os intervalos dos

vizinhos de v_i para que I_F tenha possibilidade de intersectar I_{V_i}, cujo posicionamento é feito na Linha 12. Há uma expansão possível à esquerda ou à direita, graças a ausência de ciclos induzidos com 4 ou mais vértices e triplas asteroidais. Observe também que I_F pode ser composto por um ou mais intervalos disjuntos, por isso na Linha 12 escolhemos uma porção contígua de I_F para posicionar I_{V_i}. Desta maneira, como I_C garante as interseções necessárias entre vizinhos de v_i e I_F garante que não hajam interseções entre não vizinhos de v_i, concluímos o desejado. Dado o Algoritmo 3 acima, Veja um exemplo na Figura 3.6 de um modelo de intervalo criado para o grafo estrela S_7, onde mostraremos os primeiros passos para criar o modelo e, em seguida, traremos seu modelo final.

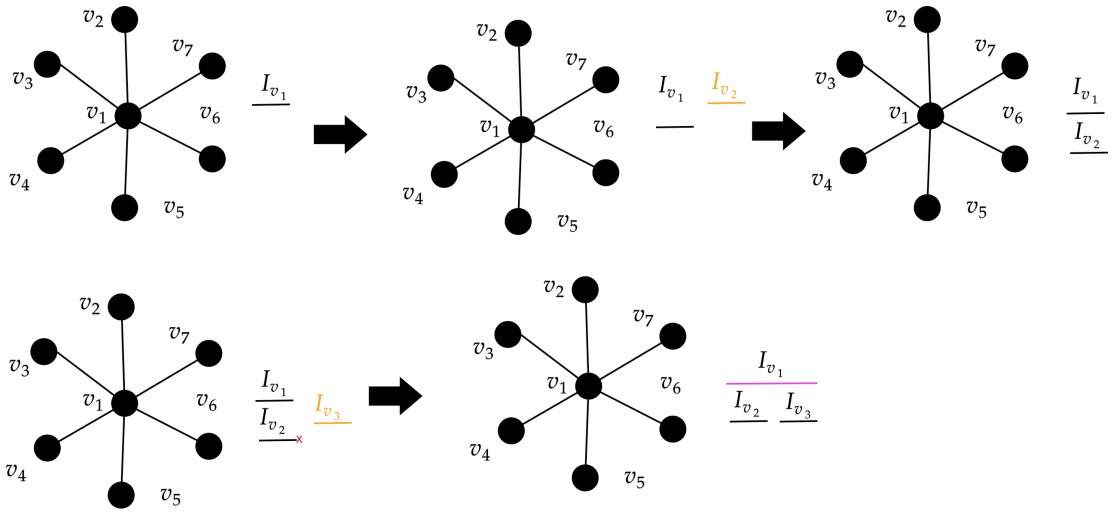

Figura 3.5: *Primeiros passos para criar um modelo para S_7*

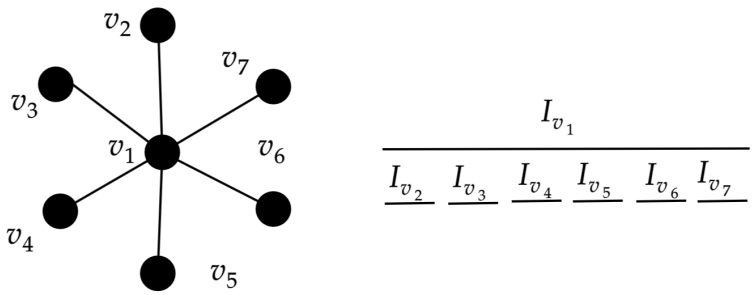

Figura 3.6: *Exemplo do modelo de intervalo final para o grafo S_7.*

□

Teorema 3.8 *Seja G um grafo de intervalo. O Algoritmo 3 quando executado sobre G produz em tempo polinomial um modelo de intervalo $\mathcal{R}$ para G.*

Prova. A Linha 1 chama o Algoritmo 2 como subrotina, que já observamos na Proposição 3.6 que usa tempo $O(n^2)$. A Linha 2 realiza apenas uma atribuição dos extremos de

I_n, o que requer tempo constante. O laço de repetição da Linha 3 é executado para $n-1$ vértices. Vamos analisar a complexidade de cada uma dessas iterações. As atribuições das Linhas 4 a 6 dependem de $n-i$ vértices, tendo complexidade $O(n)$. As Linhas 7 e 8, embora calculem uniões e interseções de intervalos também requerem tempo $O(n)$ uma vez que precisamos apenas dos pontos extremos de cada intervalo. Para as Linhas 9 a 11 devem ser expandidos $\Theta(|\mathscr{I}_A|)$ intervalos, uma vez que temos que produzir um intervalo I_F com a interseção de todos os intervalos vizinhos de v_i não vazia para posicionar I_i. Para expandir cada um deles sem criar novas sobreposições precisa-se checar as extremidades de cada outro intervalo, o que requer $O(n)$ operações, logo as Linhas 9 a 11 requerem tempo $O(n^2)$. Após isso, a Linha 12 determina as extremidades de I_i a fim de posicioná-lo em I_F o que requer tempo constante. Em suma, cada bloco das Linhas 4 a 12 gasta tempo $O(n^3)$, repetidos $O(n)$ vezes temos um tempo total de execução do Algoritmo 3 da ordem de $O(n^4)$. □

3.3 Número de Independência em Grafos de Intervalo

Como evidenciado na Introdução, no problema de escalonamento de tarefas, um conjunto independente máximo em um grafo de intervalo corresponde ao maior grupo de tarefas cujos intervalos não se sobrepõem. No Teorema 3.9, mostraremos um algoritmo que prova que o número de independência $\alpha(G)$ pode ser determinado em tempo polinomial quando G é um grafo de intervalo. O algoritmo apresentado foi encontrado por Feofiloff [10]. É importante ressaltar que $\alpha(G)$ pode ser determinado em tempo linear para grafos cordais, mas, para fins didáticos, traremos o algoritmo a seguir que é determinado em tempo polinomial.

Teorema 3.9 *[10] Seja G um grafo. Se G é grafo de intervalo, então $\alpha(G)$ pode ser determinado em tempo polinomial.*

Prova. Considere um grafo de intervalo G onde cada vértice $v \in V(G)$ corresponde a um intervalo $I_v = [\ell(v), r(v)]$, onde $\ell(v)$ e $r(v)$ representam respectivamente, o início e o término do intervalo. Inicialmente, vamos ordenar os intervalos I_v pelos seus términos $r(v)$ em ordem crescente. Considere que, sejam $a[p \dots r]$ e $b[p \dots r]$ dois vetores de números inteiros positivos tais que $[a[i] \leq b[i]]$ para cada i. Portanto, para cada i, o intervalo $a[i] \dots b[i]$ não é vazio. Diremos que o par (a,b) de vetores é um *conjunto de intervalos*.

Além disso, tome p e r como números inteiros que representam, respectivamente, o intervalo com o menor tempo de término e o intervalo com o maior tempo de término de cada vetor. O conjunto X é o conjunto que armazenará o maior conjunto

independente e o índice k indica o último intervalo adicionado a X. O índice m é um contador que começa em $p+1$ e percorre até r. O objetivo é verificar cada intervalo subsequente para determinar se ele pode ser incluído no conjunto X de forma que a propriedade de disjunção seja mantida. O algoritmo abaixo mostra como encontrar X:

Algoritmo 4: CONJUNTODISJUNTOMÁXIMO(a,b,p,r)

Entrada: Dois vetores de inteiros positivos $a[p\dots r]$ e $b[p\dots r]$ tais que $a[i] \leq b[i]$ para cada i, com b ordenado em ordem crescente e p,r inteiros positivos com $p \leq r$

Saída: Conjunto X de índices de um subconjunto disjunto máximo

```
1 X ← {p} ;                    // Inicializa o conjunto com o índice inicial
2 k ← p ;                      // Define o índice de referência inicial
3 para m ← p+1 até r faça
4     se a[m] > b[k] então
5         X ← X ∪ {m} ;        // Adiciona o índice ao conjunto X
6         k ← m ;              // Atualiza o índice de referência
7     fim
8 fim
9 retorna X ;                  // Retorna o conjunto máximo disjunto
```

Relembramos que um intervalo I é *posterior* a um intervalo J se $\ell(I) > r(J)$, ou seja, I começa depois de J, garantindo que não há interseção entre eles. Note que o algoritmo escolhe um intervalo com o menor término e remove da coleção todos os intervalos que não são posteriores ao intervalo. Com isso, concluímos que o algoritmo produz um conjunto onde todos os intervalos são disjuntos. Entretanto, é necessário provar que o conjunto é máximo. Para isso, antes de tudo, vamos introduzir a seguinte propriedade.

Afirmação 1. (Propriedade da Escolha Gulosa). *Todo intervalo de término mínimo pertence a algum conjunto disjunto máximo.*

Prova (da Afirmação 1). Seja m um intervalo de término mínimo e X um conjunto disjuntos máximo. Primeiro, considere que $m \in X$. Neste caso, não há o que discutir, m já pertence ao conjunto disjunto máximo. Agora, suponha que $m \notin X$ e seja q o intervalo com o menor término em X. Note que m termina antes de q, o que implica que todos os intervalos de X começam após o término de q, portanto, o conjunto $X \setminus \{q\} \cup \{m\}$ é disjunto. Além disso, o novo conjunto possui a mesma cardinalidade de X, tornando-o máximo. ■

Temos uma segunda propriedade, evidenciada a seguir.

Afirmação 2. (Propriedade de Subestrutura Ótima). *Seja m um intervalo de término mínimo e X um conjunto independente máximo. Se $m \in X$, então $X \setminus \{m\}$ é um conjunto disjunto máximo de todos os intervalos posteriores a m.*

Prova (da Afirmação 2). Considere que P é o conjunto de todos os intervalos posteriores a m. Como visto acima, evidentemente $X \setminus \{m\}$ é o conjunto disjunto máximo de P. Entretanto, a fim de encontrar uma contradição, suponha que existe um conjunto disjunto Y maior que X. Como Y é um conjunto disjunto máximo de intervalos posteriores a m e m termina antes de todos os intervalos em Y, então o conjunto $Y \cup \{m\}$ também é máximo. Observe que $|Y \cup \{m\}| = |Y| + 1$ pois estamos adicionando o intervalo m a Y. Por suposição, $|Y| > |X - \{m\}|$. Assim, temos: $|Y \cup \{m\}| = |Y| + 1 > |X - \{m\}| + 1 = |X|$, uma vez que somar um intervalo em X e subtrair m resulta no próprio conjunto X. Entretanto, note que Y possui mais elementos que X, o que contradiz que X é um conjunto disjunto máximo. Assim, concluímos que $X \setminus \{m\}$ é um conjunto disjunto máximo para os intervalos posteriores a m. ■

Concluimos a demonstração calculando o tempo de execução do Algoritmo 4.

Afirmação 3. *O Algoritmo 4 pode ser executado em tempo $O(n \log n)$, onde n é o número de intervalos de entrada.*

Prova (da Afirmação 3). Após a ordenação inicial dos intervalos, que pode ser feita em tempo $O(n \log n)$, a operação principal do algoritmo é uma única varredura dos n intervalos, que é realizada em tempo $O(n)$ (linhas 3 a 6). Assim, a parte dominante do tempo de execução é a ordenação, e a operação de seleção dos intervalos disjuntos é realizada em tempo linear. Portanto, o tempo total necessário para executar o algoritmo é $O(n \log n)$, que é polinomial. ■

Como resultado, podemos concluir que se G é um grafo de intervalo, $\alpha(G)$ pode ser determinado em tempo polinomial, uma vez que o Algoritmo 8 encontra o maior conjunto de intervalos disjuntos. Em um grafo de intervalo, um conjunto independente máximo de vértices corresponde diretamente ao maior conjunto de intervalos disjuntos. Portanto, ao encontrar o maior conjunto de intervalos disjuntos, o algoritmo determina $\alpha(G)$, que é o número de independência do grafo G. O algoritmo realiza essa tarefa em tempo $O(n \log n)$, que é polinomial. □

3.4 Observações sobre Grafos de Intervalo Unitário

A classe dos grafos de intervalo unitário tem sido alvo de estudos dado que herda as soluções polinomiais aplicáveis a grafos de sua superclasse grafos de intervalo. Além disso, em alguns casos, problemas em grafos de intervalo unitário podem ser resolvidos

de forma ainda mais eficiente do que em grafos de intervalo. Por exemplo, o problema do corte máximo, que é NP-completo para grafos de intervalo [1], torna-se polinomial para grafos de intervalo unitário [6].

No Teorema 3.10, a seguir, apresentamos uma propriedade de grafos de intervalo unitário com relação ao seu subgrafo proibido $K_{1,3}$. A demonstração deste teorema foi originalmente realizada por Bogart e West [4]. Neste relatório, adotamos a prova descrita em Oliveira [21], com uma abordagem mais detalhada.

Teorema 3.10 *Se G é um grafo de intervalo, então G é um grafo de intervalo unitário se e somente se G é livre de $K_{1,3}$.*

Prova.($\Rightarrow$) Supomos que G é um grafo de intervalo unitário. Vamos mostrar que G é livre de $K_{1,3}$. Suponha, por contradição, que G possui pelo menos um subgrafo isomorfo a $K_{1,3}$. Considere que $K_{1,3}$ é formado por $V(G') = \{v_1, v_2, v_3, v_4\}$ e $E(G') = \{(v_1, v_i) \mid 2 \leq i \leq 4\}$, ou seja, G não é livre de $K_{1,3}$.

Ao tentarmos montar um modelo de intervalo unitário de G, notamos que se todos os intervalos associados a esses vértices possuem intervalo unitário e $V(G) = \{v_2, v_3, v_4\}$ não se sobreponham, não seria possível ter três vértices adjacentes ao vértice v_1. Isso acontece porque não seria possível que os intervalos I_v (onde v é o índice referente a cada $v \in V(G) = \{v_2, v_3, v_4\}$) sobreponham I_{v_1} de forma que ambos os intervalos sejam unitários e exista $N(v_1) = \{v_2, v_3, v_4\}$. Sendo assim, a existência de um subgrafo $K_{1,3}$ indica que pelo menos um dos intervalos associados ao grafo não possui comprimento unitário, o que contradiz que G é um grafo de intervalo unitário.

A recíproca do Teorema 4 a seguir tem como objetivo construir um modelo de grafo unitário dado um modelo de intervalo qualquer livre de $K_{1,3}$, mas antes vamos descrever um modelo de intervalo para o grafo G.

($\Leftarrow$) Evidentemente, um grafo $K_{1,3}$ não pode ser representado por um modelo de intervalo unitário. No entanto, a fim de encontrar uma contradição, suponha que G não é livre de $K_{1,3}$. Seja $\mathcal{F} = \{I_v \mid v \in V(G)\}$ um modelo de intervalo que possui o menor número possível de interseções com outros intervalos, ou seja, intervalos incluídos em algum outro. Seja $I_j \subseteq I_k$ para algum par $\{j,k\} \subseteq V(G)$, onde nenhum intervalo está totalmente contido em outro. Pela minimalidade do número de inclusões entre intervalos, que assumimos anteriormente, temos que $\ell(j)$ não pode ser movido para a esquerda além do $\ell(k)$ sem que haja uma inclusão adicional entre intervalos, temos que $\exists x \in V(G)$ tal que $I_x \cap I_k \neq \emptyset$, ou seja, existe interseção entre I_x e I_k. Além disso, temos que $\ell(x) < \ell(k)$, ou seja, $\ell(x)$ está à esquerda de $\ell(k)$. Analogamente, $\exists y \in V(G)$ tal que $I_y \cap I_k \neq \emptyset$ e $r(k) < \ell(y)$. Entretanto, considerando os conjuntos citados acima, de acordo com a definição de grafo de intervalo, G possui um subgrafo isomorfo a $K_{1,3}$, gerando, portanto, uma contradição, uma vez que se G é um grafo de intervalo não pode conter

subrgrafos isomorfos a $K_{1,3}$. Sendo assim, apresentaremos a seguir a construção de um modelo de intervalo unitário para abordar os dois possíveis casos de G (denominados, respectivamente, *Caso 1* e *Caso 2*), transformando o conjunto $\mathcal{R}$ de forma iterativa conforme descrito a seguir.

Em cada iteração, considere I como o intervalo não unitário com o menor extremo esquerdo e considere p um ponto em I como essencial para a delimitação do intervalo unitário. *Caso 1*: Se não houver nenhum extremo direito sobreposto com I, defina $p = \ell(I)$. *Caso 2*: Caso contrário, defina p como o maior desses pontos (nesse caso, p pertenceria a um intervalo já ajustado para comprimento unitário, garantindo que não haja contradição entre a ausência de inclusões de intervalos em $\mathcal{R}$ e a escolha de I).

Assim, com o propósito de garantir que os intervalos sejam ajustados dentro dos limites definidos, tome $p \leq \min\{\ell(I)+1, r(I)\}$. Ajuste o modelo comprimindo ou expandindo proporcionalmente a porção do modelo definida por $[p, r(I)]$ para que ela se encaixe na porção $[p, \ell(I)+1]$ (translade a porção $[r(I), \infty)$ do modelo para $[\ell(I)+1, \infty)$). A ordem entre os extremos dos intervalos permanece inalterada, os intervalos ajustados em iterações anteriores mantêm seu tamanho unitário, e agora $|I_v| = 1$, para todo $v \in V(G)$. Enquanto houver intervalos não unitários, aplicaremos o algoritmo até que todos os intervalos estejam devidamente ajustados. Claramente, quando nenhum intervalo puder ser selecionado como I, o modelo de intervalo resultante será um modelo de intervalo unitário de G. □

As Figuras 3.7 e 3.8 ilustram as compressões descritas no Teorema 3.10.

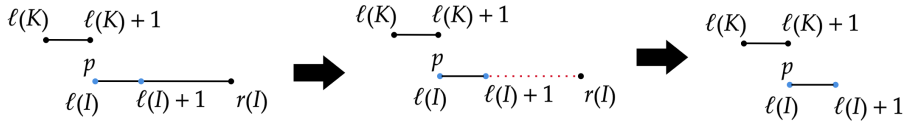

Figura 3.7: *Caso 1.*

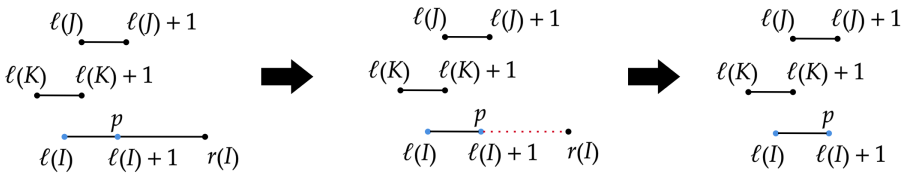

Figura 3.8: *Caso 2.*

CAPÍTULO 4

Conclusão

Nesta monografia, mostramos algumas propriedades relacionadas aos conjuntos independentes máximos em grafos de intervalo e algoritmos associados a essa classe.

Inicialmente, no Teorema 3.1, mostramos que um grafo G é de intervalo se, e somente não possui ciclos induzidos onde $n \geq 4$. A prova usa a definição de modelo de intervalo para demonstrar que, ao tentar representar os vértices e arestas de C_n com intervalos na reta real, ocorre uma contradição.

Além disso, no Teorema 3.3 mostramos que um grafo G de intervalo não pode possuir uma tripla asteroidal. Para isso, utilizamos a definição de tripla asteroidal e a disposição linear dos intervalos em um modelo de intervalo para demonstrar que a existência de uma tripla asteroidal leva a uma contradição. A disposição sequencial dos intervalos impede a formação de caminhos que evitem a vizinhança de um terceiro vértice, o que impossibilita a existência de uma tripla asteroidal em grafos de intervalo.

Na seção 3.4, caracterizamos um grafo de intervalo unitário como aquele livre de $K_{1,3}$. A partir disso, apresentamos um algoritmo para ajustar os intervalos e garantir que um grafo de intervalo tenha modelo unitário, removendo qualquer subgrafo isomorfo a $K_{1,3}$. O algoritmo ajusta iterativamente os intervalos até que todos tenham comprimento unitário, sem violar as condições do grafo.

No Teorema 3.9, apresentamos um algoritmo eficiente para determinar $\alpha(G)$ em um grafo de intervalos. O algoritmo descrito tem uma complexidade polinomial $O(n \log n)$, alcançada através de uma ordenação inicial seguida de uma varredura linear dos intervalos.

Finalmente, também implementamos um algoritmo, proposto por Feofiloff [10], em linguagem *python*, com o objetivo de determinar intervalos disjuntos a partir de uma lista de intervalos. A implementação pode ser encontrada em https://github.com/AnaCarolynePds/Descobrindo-alpha-G-em-um-grafo-de-intervalo. O código está disponível no Apêndice A.

Como trabalho futuro, sugerimos a implementação de algoritmos para outros problemas em grafos, como encontrar a cardinalidade de uma clique máxima e outros problemas clássicos em grafos.

Referências

[1] ADHIKARY, R.; BOSE, K.; MUKHERJEE, S.; ROY, B. **Complexity of maximum cut on interval graphs**. *Discrete & Computational Geometry*, 70(2):307–322, 2023.

[2] AGARWAL, P. K.; VAN KREVELD, M.; SURI, S. **Label placement by maximum independent set in rectangles**. *Computational Geometry*, 11(3-4):209–218, 1998.

[3] BENZER, S. **On the topology of the genetic fine structure**. *Proceedings of the National Academy of Sciences*, 45(11):1607–1620, 1959.

[4] BOGART, K. P.; WEST, D. B. **A short proof that "proper = unit"**. *arXiv preprint math/9811036*, 1998.

[5] BONDY, J. A.; MURTY, U. S. R.; OTHERS. **Graph theory with applications**, volume 290. Macmillan London, 1976.

[6] BOYACI, A.; EKIM, T.; SHALOM, M. **A polynomial-time algorithm for the maximum cardinality cut problem in proper interval graphs**. *Information Processing Letters*, 121:29–33, 2017.

[7] CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Algoritmos: teoria e prática**. *Editora Campus*, 2:296, 2002.

[8] DE MELLO, C. P.; CERIOLI, M. R.; DE SOUZA, C. C.; DE SOUZA, S. D. **Sheila morais de almeida**. *Universidade Estadual de Campinas*, 2005.

[9] DIESTEL, R. **Graph theory**. Springer (print edition); Reinhard Diestel (eBooks), 2024.

[10] FEOFILOFF, P. **Minicurso de análise de algoritmos**. *Departamento de Ciência da Computação, Instituto de Matemática e Estatística, Universidade de São Paulo*, 2017. Disponível em: https://www.ime.usp.br/~pf/livrinho-AA/.

[11] GAREY, M. R.; JOHNSON, D. S. **Computers and Intractability: A Guide to the Theory of NP-Completeness**. W. H. Freeman and Co., New York, USA, 1979.

[12] GOLUMBIC, M. C. **Algorithmic graph theory and perfect graphs**. Elsevier, 2004.

[13] HAYWARD, R. B.; SPINRAD, J.; SRITHARAN, R. **Weakly chordal graph algorithms via handles**. In: *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, p. 42–49, 2000.

[14] HSU, W.-L.; SPINRAD, J. P. **Independent sets in circular-arc graphs**. *Journal of Algorithms*, 19(2):145–160, 1995.

[15] KARP, R. M. **Mapping the genome: some combinatorial problems arising in molecular biology**. In: *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, p. 278–285, 1993.

[16] KLEINBERG, J.; TARDOS, E. **Algorithm design**. Pearson Education India, 2006.

[17] LEKKEIKERKER, C.; BOLAND, J. **Representation of a finite graph by a set of intervals on the real line**. *Fundamenta Mathematicae*, 1962.

[18] MOHAR, B. **Face covers and the genus problem for apex graphs**. *Journal of Combinatorial Theory, Series B*, 82(1):102–117, 2001.

[19] NÖKEL, K. **Temporally distributed symptoms in technical diagnosis**, volume 517. Springer Science & Business Media, 1991.

[20] NOURANIZADEH, A.; MATINKIA, M.; RAHMATI, M.; SAFABAKHSH, R. **Maximum entropy weighted independent set pooling for graph neural networks**. *arXiv preprint arXiv:2107.01410*, 2021.

[21] OLIVEIRA, F. S. **Sobre Ordens e Grafos de Intervalo**. PhD thesis, Universidade Federal do Rio de Janeiro, 2011.

[22] WARD, S. A.; HALSTEAD, R. H. **Computation structures**. MIT press, 1990.

APÊNDICE A

Códigos de Programas

```
def verifica_intersecao(intervalo1, intervalo2):
    left1, right1 = intervalo1
    left2, right2 = intervalo2
    return left1 <= right2 and left2 <= right1  # retorna true
       caso exista intersecao


def imprimir_matriz(matriz):
    for linha in matriz:
        for elemento in linha:
            print(elemento, end=' ')
        print()


# intervalos = [[1,3], [2,5], [8,9], [4,6], [10,20]] # cria a
   lista de intervalos nao necessariamente ordenados
intervalos = [[1,3], [2,9], [3,6], [2,4]]
intervalos_ordenados = []  # lista vazia que recebera a lista de
    intervalos ordenados
scd = []  # lista vazia que recebera os intervalos disjuntos
intervalos_ordenados = sorted(intervalos, key=lambda x: x[1])  #
    a funcao sorted ordena em ordem crescente


# M = [[]]
M = [[0 for _ in range(len(intervalos))] for _ in range(len(
   intervalos))]
for i in range(len(intervalos)):
    for j in range(len(intervalos)):
        if (verifica_intersecao(intervalos[i], intervalo[j]) and
            (i != j)):
```

```
            M[i][j] = 1
            M[j][i] = 1


imprimir_matriz(M)


while intervalos_ordenados:
    menor_intervalo = intervalos_ordenados.pop(0)  # apos
       ordenado, o menor intervalo e retirado da lista e
       adicionado em scd
    scd.append(menor_intervalo)

    intervalos_ordenados = [intervalo for intervalo in
       intervalos_ordenados if not verifica_intersecao(
       menor_intervalo, intervalo)]  # list comprehension
    '''
    caso o ultimo valor adicionado em scd NAO tenha interseccao
       com outro valor do intervalo, o intervalo sem interseccao
        e mantido na lista intervalos_ordenados.
    Em seguida, apos atualizar a lista intervalos_ordenados,
       refaz todo o processo ate que nao tenha mais nenhum valor
        a se verificar em numeros ordenados
    '''


print("Lista de intervalos SCD:")
print(scd)
quantidade = len(scd)
print("Existem %d colecoes disjuntas na lista de intervalos
   fornecidas." %quantidade)
```