BACHAREL EM CIÊNCIA DA COMPUTAÇÃO

SOBRE GRAFO SANDUÍCHE

MARIANA FONTES GONÇALVES

Rio Verde, GO

2025

INSTITUTO FEDERAL GOIANO - CAMPUS RIO VERDE
BACHAREL EM CIÊNCIA DA COMPUTAÇÃO

SOBRE GRAFO SANDUÍCHE

MARIANA FONTES GONÇALVES

Trabalho de Conclusão de Curso apresentado ao Instituto Federal Goiano - Campus Rio Verde, como requisito parcial para a obtenção do Grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. André da Cunha Ribeiro

Rio Verde, GO

NOVEMBRO, 2025

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO

PARA DISPONIBILIZAR PRODUÇÕES TÉCNICO-CIENTÍFICAS NO REPOSITÓRIO INSTITUCIONAL DO IF GOIANO

Com base no disposto na Lei Federal nº 9.610, de 19 de fevereiro de 1998, AUTORIZO o Instituto Federal de Educação, Ciência e Tecnologia Goiano a disponibilizar gratuitamente o documento em formato digital no Repositório Institucional do IF Goiano (RIIF Goiano), sem ressarcimento de direitos autorais, conforme permissão assinada abaixo, para fins de leitura, download e impressão, a título de divulgação da produção técnico-científica no IF Goiano.

IDENTIFICAÇÃO DA PRODUÇÃO TÉCNICO-CIENTÍFICA

- ☐ Tese (doutorado)
- ☐ Dissertação (mestrado)
- ☐ Monografia (especialização)
- ☑ TCC (graduação)
- ☐ Artigo científico
- ☐ Capítulo de livro
- ☐ Livro
- ☐ Trabalho apresentado em evento
- ☐ Produto técnico e educacional - Tipo:

Nome completo do autor:
Mariana Fontes Gonçalves

Matrícula:
2022102201940297

Título do trabalho:
Sobre Grafo Sanduíche

RESTRIÇÕES DE ACESSO AO DOCUMENTO

Documento confidencial: ☑ Não ☐ Sim, justifique:

Informe a data que poderá ser disponibilizado no RIIF Goiano: 03 / 12 / 2025

O documento está sujeito a registro de patente? ☐ Sim ☑ Não

O documento pode vir a ser publicado como livro? ☐ Sim ☑ Não

DECLARAÇÃO DE DISTRIBUIÇÃO NÃO-EXCLUSIVA

O(a) referido(a) autor(a) declara:

• Que o documento é seu trabalho original, detém os direitos autorais da produção técnico-científica e não infringe os direitos de qualquer outra pessoa ou entidade;

• Que obteve autorização de quaisquer materiais inclusos no documento do qual não detém os direitos de autoria, para conceder ao Instituto Federal de Educação, Ciência e Tecnologia Goiano os direitos requeridos e que este material cujos direitos autorais são de terceiros, estão claramente identificados e reconhecidos no texto ou conteúdo do documento entregue;

• Que cumpriu quaisquer obrigações exigidas por contrato ou acordo, caso o documento entregue seja baseado em trabalho financiado ou apoiado por outra instituição que não o Instituto Federal de Educação, Ciência e Tecnologia Goiano.

Rio Verde
Local

28 / 11 / 2025
Data

gov.br Documento assinado digitalmente
MARIANA FONTES GONCALVES
Data: 28/11/2025 13:34:38-0300
Verifique em https://validar.iti.gov.br

Assinatura do autor e/ou detentor dos direitos autorais

Ciente e de acordo:

Assinatura do(a) orientador(a)

gov.br Documento assinado digitalmente
ANDRE DA CUNHA RIBEIRO
Data: 28/11/2025 13:21:21-0300
Verifique em https://validar.iti.gov.br

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema Integrado de Bibliotecas do IF Goiano - SIBi

Gonçalves, Mariana Fontes
G635s Sobre Grafo Sanduíche / Mariana Fontes Gonçalves. Rio Verde 2025.

57f. il.

Orientador: Prof. Dr. André da Cunha Ribeiro.
Tcc (Bacharel) - Instituto Federal Goiano, curso de 0219201 - Bacharelado em Ciência da Computação - Integral - Rio Verde (Campus Rio Verde).
1. grafo sanduíche. 2. partição (k,l). 3. grafo split. 4. problema sanduíche. I. Título.

Regulamento de Trabalho de Curso (TC) – IF Goiano - Campus Rio Verde

ANEXO V - ATA DE DEFESA DE TRABALHO DE CURSO

Aos 1**2** dias do mês de novembro de **dois mil e vinte e cinco**, às 9 horas, reuniu-se a Banca Examinadora composta por: Prof. **André da Cunha Ribeiro** (orientador), Prof. **Marcio Antonio Ferreira Belo Filho** e Prof. **Diane Castonguay**, para examinar o Trabalho de Curso (TC) intitulado **SOBRE GRAFO SANDUÍCHE**, de **Mariana Fontes Gonçalves**, estudante do curso de **Bacharelado em Ciência da Computação** do IF Goiano – Campus Rio Verde, sob Matrícula nº **2022102201940297**. A palavra foi concedida à estudante para a apresentação oral do TC, em seguida houve arguição da candidata pelos membros da Banca Examinadora. Após tal etapa, a Banca Examinadora decidiu pela **APROVAÇÃO** da estudante. Ao final da sessão pública de defesa foi lavrada a presente ata, que, foi assinada pelo professor orientador e pelo membro da banca prof Márcio Belo, uma vez que o professor orientador assinou pelo membro externo Diane Castonguay.

Rio Verde, 12 de novembro de 2025.

André da Cunha Ribeiro

Orientador

Marcio Antonio Ferreira Belo Filho

Membro da Banca Examinadora

Diane Castonguay

Membro da Banca Examinadora

Documento assinado eletronicamente por:

- **Andre da Cunha Ribeiro**, **PROFESSOR ENS BASICO TECN TECNOLOGICO** , em 18/11/2025 10:13:56.
- **Marcio Antonio Ferreira Belo Filho**, **PROFESSOR ENS BASICO TECN TECNOLOGICO** , em 18/11/2025 10:20:33.

Este documento foi emitido pelo SUAP em 17/11/2025. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifgoiano.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 764203
Código de Autenticação: 59b0d76dcc

AGRADECIMENTOS

Agradeço primeiramente a Deus, por ser minha força, meu refúgio e meu guia em todos os momentos. Foi Ele quem me sustentou nas dificuldades, me deu coragem quando pensei em desistir e iluminou meus passos em cada etapa desta caminhada. Sem a presença e a graça de Deus, nada disso seria possível. A Ele, toda honra, toda glória e toda gratidão.

Agradeço com todo o meu coração aos meus professores, que com dedicação, paciência e amor pelo ensino contribuíram de forma essencial para a minha formação acadêmica e pessoal: Dr. Adriano Bailão, Dr. André Ribeiro, Dr. Douglas Cedrim, Dra. Heyde Francielle, Dr. Heverton Barros, Me. Andrea Proto, Me. Fábio Montanha, Me. Leonel Diógenes, Dr. Márcio Belo, Dr. Márcio Vilela e Me. Rafael Carvalho.

Cada um de vocês deixou uma marca em minha trajetória, com ensinamentos que levarei para toda a vida. Obrigada por cada aula, cada palavra de incentivo, cada gesto de apoio e cada momento compartilhado. Levo comigo não apenas o conhecimento, mas também o carinho, o respeito e a admiração que sinto por todos. Guardarei cada um de vocês no meu coração e jamais esquecerei o quanto contribuíram para que eu chegasse até aqui.

Em especial, agradeço ao Dr. André Ribeiro, meu orientador, por todos os conselhos, pela paciência, por ter me apoiado e acreditado em mim desde o início do curso até aqui. Sua orientação, incentivo e confiança foram fundamentais para que eu pudesse crescer, aprender e concluir esta etapa tão importante da minha vida.

Estendo também minha gratidão ao Instituto Federal Goiano — Campus Rio Verde, instituição que me acolheu desde o início da minha jornada acadêmica e me proporcionou oportunidades que foram essenciais para o meu crescimento pessoal e profissional. Agradeço com carinho à equipe da Assistência Estudantil, que sempre esteve presente, me apoiando e ajudando a tornar possível a realização desse sonho.

Agradeço profundamente à minha família, que sempre foi a base de tudo na minha vida. Ao meu pai, por me ensinar a ter força e perseverança; às minhas irmãs, pelo carinho e incentivo em todos os momentos. À minha mãe, que já se foi para um lugar melhor, mas que me aconselhou a seguir este caminho e sempre acreditou em mim — suas palavras continuam vivas no meu coração e me trouxeram até aqui.

Agradeço com muito amor ao meu esposo, que esteve ao meu lado em todos os momentos, inclusive nos mais difíceis. Obrigada por acreditar em mim mesmo quando eu duvidava, por me apoiar, me incentivar e me dar forças para continuar. Essa conquista também é sua, porque você caminhou comigo passo a passo.

...Isto é água... Isto é água. (WALLACE, David Foster, 2005).

RESUMO

GONCALVES, M.F. **Sobre Grafo Sanduíche.** NOVEMBRO, 2025. 57 f. Monografia – (Curso de Bacharel em Ciência da Computação), Instituto Federal Goiano - Campus Rio Verde. Rio Verde, GO.

Este trabalho apresenta uma revisão bibliográfica sobre o Problema do Grafo Sanduíche, destacando sua formulação, principais aplicações e comportamento em diferentes cenários de complexidade computacional. Inicialmente, são abordados os conceitos fundamentais da Teoria dos Grafos, da complexidade computacional e da partição (k,l), que ocorre quando seu conjunto de vértices pode ser dividido em k conjuntos independentes e l conjuntos que são cliques, que servem de base para a análise. Em seguida, são discutidas cinco variações do problema: split $(1,1)$, bipartido $(2,0)$, partição $(2,1)$, partição $(2,2)$ e $(k,l)\Delta$-Limitado. Os resultados mostram que as variações split $(1,1)$ e bipartido $(2,0)$ são resolvíveis em tempo polinomial. As partições $(2,1)$ e $(2,2)$ são NP-completos, comprovadas por reduções do 3-SAT. Para os casos do $(k,l)\Delta$-Limitado, verificou-se que o problema é polinomial quando $k \leq 2$ ou $\Delta \leq 3$, e torna-se NP-completo quando $k \geq 3$ e $\Delta \geq 4$, evidenciando como a estrutura e o grau máximo do grafo impactam diretamente a complexidade. Conclui-se que a análise por meio da partição (k,l) é uma ferramenta eficaz para compreender os limites entre casos tratáveis e intratáveis, além de destacar a relevância teórica e prática desse problema em diferentes contextos.

Palavras-chave: grafo sanduíche, partição (k,l), grafo split, problema sanduíche.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

SIGLAS

2-CNF	Forma Normal Conjuntiva em que cada cláusula possui no máximo dois literais.
2-SAT	Versão do problema SAT em que cada cláusula da fórmula tem no máximo 2 literais.
3-CNF	Forma Normal Conjuntiva em que cada cláusula possui exatamente três literais.
3-SAT	Versão do problema SAT em que cada cláusula da fórmula tem exatamente 3 literais.
BFS	Breadth-First Search (Busca em Largura) — algoritmo de busca em grafos que percorre vértices por níveis.
CNF	Conjunctive Normal Form (Forma Normal Conjuntiva) — forma de uma fórmula booleana escrita como uma conjunção (AND) de cláusulas, onde cada cláusula é uma disjunção (OR) de literais.
DFS	Depth-First Search (Busca em Profundidade) — algoritmo de busca em grafos que explora vértices aprofundando-se nos caminhos antes de retroceder.
NP	Classe de problemas verificáveis em tempo polinomial.
NP-completo	Classe de problemas mais difíceis da classe NP.
P	Classe de problemas solucionáveis em tempo polinomial.
SAT	Boolean Satisfiability Problem (Problema da Satisfatibilidade Booleana) — problema de decidir se existe uma atribuição de valores verdadeiro/falso que satisfaça uma fórmula booleana.
SCC	Strongly Connected Component (Componente Fortemente Conexa) — subgrafo no qual todos os vértices são alcançáveis entre si.

LISTA DE SÍMBOLOS

G	Grafo original.
$G(V, E)$	Representação de um grafo G com conjunto de vértices V e arestas E.
G_1, G_2	Subgrafos de G.
G_3	Grafo formado pelo conjunto de vértices V e pelas arestas proibidas E_3.
$G_1(V, E_1)$, $G_2(V, E_2)$	Grafos inferior e superior do problema do grafo sanduíche.
V	Conjunto de vértices do grafo.
E	Conjunto de arestas do grafo.
E_1, E_2	Subconjuntos de arestas associados, respectivamente, aos grafos G_1 e G_2.
E_3	Conjunto de arestas proibidas, que não podem estar presentes no grafo solução.
n	Número de vértices do grafo.
m	Número de arestas do grafo.
$\{v_i, v_j\}$	Aresta que conecta o vértice v_i ao vértice v_j.
$x \in V$	Indica que o vértice x pertence ao conjunto de vértices V do grafo.
$V \setminus S$	Diferença entre os conjuntos V e S, correspondendo ao conjunto de vértices em V que não pertencem a S.
$\Delta(G)$	Grau máximo do grafo G.
$\sigma(G)$	O menor número de vértices que precisam ser removidos para que G se torne um grafo split.
$m(d)$	Quantidade mínima de vértices de grau d que devem ser removidos para que o grafo possa ser transformado em um grafo split.
C	Fórmula booleana em CNF.
ℓ_n	Literal na fórmula em CNF.
$\overline{\ell_n}$	Negação do literal ℓ_n.
(X, C)	Instância do problema SAT, onde X é o conjunto de variáveis booleanas e C a fórmula em CNF.
x_i	Variável booleana de índice i no conjunto X.
$\overline{x}_i$	Negação da variável booleana x_i.
(k, l)	Partição de um grafo em k conjuntos independentes e l cliques.

$E_1 \subseteq E \subseteq E_2$ Condição do Problema do Grafo Sanduíche: o grafo solução deve conter todas as arestas de E_1 e estar contido em E_2.

$E_2 \setminus E_1$ Arestas que estão em E_2 mas não em E_1.

$E_1 \subseteq E$ Indica que todas as arestas obrigatórias estão no grafo solução.

$E \cap E_3 = \emptyset$ Indica que o grafo solução não contém arestas proibidas.

(V, E_1, E_3) Instância do Problema do Grafo Sanduíche Generalizado.

SUMÁRIO

1 INTRODUÇÃO

A Teoria dos Grafos é um ramo da Matemática que estuda estruturas formadas por vértices e arestas, permitindo representar e analisar relações entre elementos de um conjunto. Seu surgimento remonta ao famoso Problema das Pontes de Königsberg, proposto por Leonhard Euler em 1736, considerado o marco inicial da área. Desde então, a teoria evoluiu significativamente e tornou-se uma ferramenta essencial na resolução de problemas em diversas áreas, como Computação, Engenharia, Biologia, Logística e Ciências Sociais.

Entre os diversos problemas estudados nessa área, destaca-se o Problema do Grafo Sanduíche, que pode ser visto como uma generalização dos problemas de reconhecimento de classes de grafos. Nesse contexto, além das arestas obrigatórias, também é fornecido um conjunto de arestas opcionais, permitindo verificar se existe um grafo intermediário que satisfaça determinadas propriedades estruturais. Formalmente, um grafo $G(V, E)$ é chamado de grafo sanduíche dos grafos $G_1(V, E_1)$ e $G_2(V, E_2)$ se, e somente se, seu conjunto de arestas satisfaz $E_1 \subseteq E \subseteq E_2$. Em outras palavras, G possui o mesmo conjunto de vértices V dos grafos G_1 e G_2, contém obrigatoriamente todas as arestas de G_1 e pode incluir algumas arestas opcionais de G_2 que não pertencem a G_1. Além disso, é possível considerar também um conjunto E_3 de arestas proibidas, que não podem estar presentes no grafo solução, tornando o problema ainda mais geral e desafiador.

Dessa forma, a questão central do Problema do Grafo Sanduíche pode ser formulada da seguinte maneira: dado um conjunto de vértices V, um conjunto de arestas obrigatórias E_1 e um conjunto de arestas proibidas E_3, pergunta-se se existe um grafo $G(V, E)$ tal que $E_1 \subseteq E$ e $E \cap E_3 = \emptyset$, e que apresente uma propriedade bem definida que caracteriza a classe de grafos desejada. Em outras palavras, busca-se construir um grafo intermediário que respeite todas as restrições impostas e atenda ao critério estrutural estabelecido para a solução.

Os problemas sanduíche surgiram a partir de aplicações práticas na Biologia Computacional, sendo o mapeamento físico de DNA uma das principais motivações para sua definição formal. Nesse processo, a molécula de DNA é dividida em fragmentos menores porque seu comprimento original torna inviável a análise direta em laboratório. A fragmentação permite que cada parte seja examinada com mais precisão e que se identifiquem sobreposições entre os fragmentos. A partir dessas informações, os pesquisadores tentam reconstruir a sequência completa do DNA, organizando os fragmentos na ordem correta. No entanto, como os dados experimentais nem sempre são completos ou exatos, surgem incertezas sobre a relação entre alguns fragmentos, o que leva à formulação do problema como um caso típico de grafo sanduíche.

Nesse contexto, o problema pode ser representado como um grafo sanduíche, no qual os vértices correspondem aos fragmentos de DNA e as arestas representam as relações de sobreposição entre eles. As arestas obrigatórias indicam fragmentos que com certeza se sobrepõem; as arestas proibidas representam fragmentos que não podem se sobrepor; e as arestas

incertas refletem situações em que os dados experimentais não permitem confirmar ou negar a relação entre dois fragmentos. Por exemplo, se os fragmentos F_1 e F_2 se sobrepõem de forma comprovada, essa ligação será obrigatória; se F_3 e F_4 não se sobrepõem, será proibida; e, se não houver dados confiáveis entre F_2 e F_3, a ligação permanecerá incerta. O desafio está em decidir quais dessas arestas incertas devem ser incluídas para que a estrutura final represente corretamente a sequência original, evidenciando a importância e a complexidade do Problema do Grafo Sanduíche.

Outra aplicação importante do Problema do Grafo Sanduíche ocorre no raciocínio temporal, que lida com a organização lógica de um conjunto de eventos e das relações existentes entre eles ao longo do tempo. Para cada par de eventos, pode-se ter a informação de que são disjuntos (não ocorrem simultaneamente), que compartilham um ponto no tempo (um termina exatamente quando o outro começa) ou que ambas as situações são possíveis. Essas informações podem ser modeladas por um grafo, em que cada evento é representado por um vértice e as conexões entre eles são determinadas pelas restrições temporais conhecidas.

Por exemplo, suponha três eventos A, B e C. Se A e B devem ocorrer um após o outro, essa ligação será uma aresta obrigatória. Se A e C não podem acontecer ao mesmo tempo, trata-se de uma aresta proibida. Já se a relação entre B e C ainda não é conhecida, teremos uma aresta incerta. O desafio está em decidir quais arestas incertas devem ser incluídas para que seja possível atribuir intervalos de tempo coerentes a todos os eventos, respeitando as restrições conhecidas. Assim como no mapeamento físico de DNA, essa aplicação também se enquadra no contexto do grafo sanduíche e evidencia a dificuldade de reconstruir estruturas consistentes a partir de informações incompletas.

O estudo do Problema do Grafo Sanduíche é motivado pela necessidade de lidar com situações em que as informações sobre as conexões entre elementos são parciais ou incertas. Em muitos cenários reais, parte das relações é conhecida com precisão, enquanto outras permanecem indefinidas ou restritas por condições específicas. Nesses casos, torna-se necessário completar ou ajustar as informações disponíveis de forma a obter uma estrutura coerente, que respeite todas as restrições impostas e satisfaça as propriedades desejadas para a classe de grafos considerada.

Neste trabalho, a complexidade do Problema do Grafo Sanduíche foi revisada com base na partição (k,l), que permite dividir o conjunto de vértices em k conjuntos independentes e l cliques. A escolha dessa abordagem se justifica porque a partição (k,l) modela diferentes formas de organização dos vértices, tornando possível representar estruturas simples e complexas dentro de um mesmo quadro teórico. Por exemplo, em um cenário de mapeamento de DNA, alguns fragmentos podem estar fortemente conectados entre si (formando cliques), enquanto outros permanecem isolados ou com poucas conexões (formando conjuntos independentes). Ao variar k e l, é possível observar como essas estruturas influenciam diretamente a dificuldade de encontrar uma solução.

Quando k e l assumem valores baixos, como $(1,1)$ ou $(2,0)$, o problema pode ser resolvido em tempo polinomial, permitindo algoritmos eficientes. Por outro lado, quando os valores

de k e l aumentam, a estrutura do grafo torna-se mais complexa e o problema atinge a classe NP-completo, tornando-se difícil de resolver computacionalmente. Para casos $(k,l)\Delta-$Limitado, o problema é polinomial quando $k \leq 2$ ou $\Delta \leq 3$ tornando-se NP-completo nos demais casos, evidenciando o impacto direto do grau máximo na complexidade. Assim, a partição (k,l) fornece uma ferramenta teórica poderosa para entender por que alguns casos do Problema do Grafo Sanduíche são tratáveis e outros não, além de delimitar fronteiras de complexidade.

Dentro desse contexto, o Capítulo 2 apresenta a fundamentação teórica que sustenta o desenvolvimento deste trabalho. Essa etapa é essencial para compreender os conceitos, propriedades e estruturas que servirão de base para a análise da complexidade do Problema do Grafo Sanduíche a partir da partição (k,l), permitindo estabelecer a relação entre diferentes classes de grafos e seus respectivos níveis de dificuldade computacional.

Na Seção 2.1, são introduzidos os conceitos básicos da Teoria dos Grafos, abordando definições, propriedades e estruturas fundamentais que servem de suporte para os capítulos seguintes. Em seguida, na Seção 2.2, discute-se a complexidade computacional, destacando as classes P e NP-completo, que permitem entender por que certos casos do problema são tratáveis enquanto outros se tornam computacionalmente difíceis.

Por fim, na Seção 2.3, apresenta-se o problema de partição em grafos, que está diretamente relacionado à estratégia utilizada neste trabalho para analisar o Problema do Grafo Sanduíche. Essa seção mostra como a partição (k,l) permite organizar diferentes estruturas de grafos e identificar fronteiras de complexidade. Assim, este capítulo fornece a base conceitual necessária para as discussões e resultados apresentados nos capítulos seguintes.

No Capítulo 3, apresenta-se de forma detalhada o Problema do Grafo Sanduíche, abordando sua formulação, principais características e variações relacionadas à complexidade computacional. O objetivo deste capítulo é fornecer uma visão organizada do problema, destacando como diferentes classes de grafos influenciam diretamente a dificuldade de resolução.

Na Seção 3.1, são apresentadas as definições preliminares do problema, descrevendo formalmente seus elementos e condições. Em seguida, nas Seções 3.2 a 3.5, são analisadas quatro variações associadas a partições específicas (k,l): grafos split $(1,1)$, grafos bipartidos $(2,0)$, grafos com partição $(2,1)$ e grafos com partição $(2,2)$. Por fim, na Seção 3.6, discute-se o Problema do Grafo Sanduíche em $(k,l)\Delta$-Limitado, que permite generalizar e investigar a influência do grau máximo na complexidade do problema.

Por fim, no Capítulo 4, apresentam-se as considerações finais deste trabalho. São retomados os principais conceitos e resultados discutidos ao longo do texto, destacando a importância do Problema do Grafo Sanduíche, suas aplicações práticas e a análise da complexidade com base na partição (k,l).

2 FUNDAMENTAÇÃO TEÓRICA

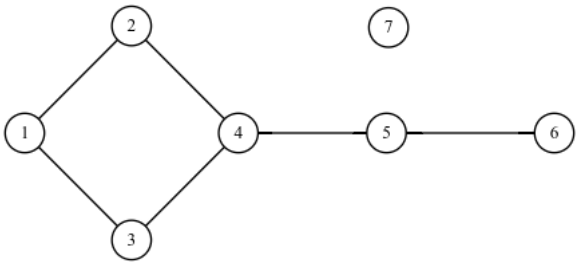
Neste capítulo, apresentam-se os conceitos e fundamentos teóricos que servem de base para o desenvolvimento deste trabalho. Inicialmente, são abordados os princípios da teoria dos grafos, incluindo definições, propriedades e estruturas fundamentais. Em seguida, discute-se a complexidade computacional, com ênfase em problemas solucionáveis em tempo polinomial (classe P) e problemas NP-completo, que permitem compreender a dificuldade associada à resolução de determinados problemas computacionais. Além disso, aborda-se o problema de partição em grafos, tema relevante para a análise e decomposição de estruturas complexas e para diversas aplicações práticas. Essa fundamentação teórica fornece os elementos necessários para o entendimento dos métodos e análises que serão desenvolvidos no capítulo a seguir.

2.1 Conceitos Básicos da Teoria de Grafos

Um grafo constitui uma estrutura matemática amplamente empregada para modelar relações entre objetos, em que os objetos são representados por vértices (ou nós) e as relações entre eles por arestas. Formalmente, um grafo é definido como um par ordenado $G(V, E)$, em que V representa o conjunto de vértices, $V = \{v_1, v_2, \ldots, v_n\}$, e E corresponde ao conjunto de arestas, formado por pares de vértices $\{u, v\}$, que indicam as conexões existentes entre os elementos (FEOFILOFF; KOHAYAKAWA; WAKABAYASHI, 2004).

Conforme exemplificado na Figura 1, apresenta-se a representação gráfica do grafo $G(V,E)$, em que o conjunto de vértices é $V = \{1,2,3,4,5,6,7\}$ e o conjunto de arestas é $E = \{(1,2),(1,3),(2,4),(3,4),(4,5),(5,6)\}$.

Figura 1 – Representação gráfica do grafo $G(V,E)$.

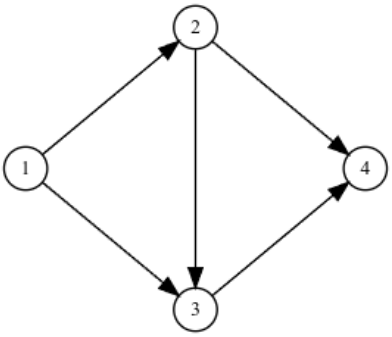


Fonte: Modificado de Feofiloff, Kohayakawa e Wakabayashi (2004).

Os grafos podem ser classificados, principalmente, em dois tipos: direcionados e não direcionados. Nos grafos direcionados, as arestas possuem uma orientação específica, geralmente representada por setas que indicam o sentido da relação entre os vértices. Já nos grafos não direcionados, as arestas não apresentam orientação, sendo representadas apenas como linhas entre os vértices. Cada tipo de grafo possui implicações distintas quanto à interpretação e ao comportamento das relações modeladas (RECUERO, 2014).

Como mostrado na Figura 1, tem-se um grafo não direcionado. Já a Figura 2 apresenta um grafo direcionado $G(V,E)$, em que o conjunto de vértices é $V = \{1,2,3,4\}$ e o conjunto de arestas é $E = \{(1,2),(1,3),(2,3),(2,4),(3,4)\}$.

Figura 2 – Representação de um grafo $G(V,E)$ direcionado.



Fonte: Modificado de Macambira et al. (2022).

Em um grafo não direcionado, um caminho é definido como uma sequência de vértices $(v_1, v_2, \ldots, v_p)$, em que p representa a quantidade de vértices que compõem o caminho. Para cada $1 \leq i < p$, existe uma aresta que conecta v_i a v_{i+1}. Nesse caminho, não há repetição de vértices. O comprimento do caminho é dado por $p - 1$, que corresponde ao número de arestas percorridas. Como os grafos não direcionados não possuem orientação, basta que exista uma aresta entre cada par de vértices consecutivos na sequência, independentemente do sentido (FREITAG, 2025).

Um ciclo em um grafo corresponde a um caminho fechado no qual o primeiro e o último vértice coincidem ($v_1 = v_p$), com $p \geq 3$, e todos os demais vértices são distintos entre si. Em outras palavras, trata-se de uma sequência de vértices que retorna ao ponto de partida sem repetir vértices, exceto o vértice inicial/final (FREITAG, 2025).

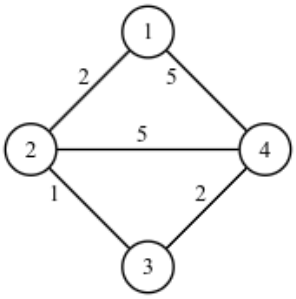
Conforme ilustrado na Figura 1, é possível identificar um caminho entre os vértices 1 e 6, por exemplo: $1 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 6$, com comprimento igual a 4. Além disso, o conjunto $(1,2,4,3,1)$ forma um ciclo, pois retorna ao vértice inicial e todos os vértices intermediários são distintos entre si. Já na Figura 2, na página 5, observa-se um caminho direcionado de $1 \rightarrow 2 \rightarrow 4$, com comprimento 2. Não há ciclo neste grafo, uma vez que não existe uma sequência que retorne ao vértice inicial seguindo a orientação das setas.

Considera-se um grafo rotulado aquele que possui valores (ou rótulos) atribuídos aos seus vértices ou arestas. Esses rótulos podem ser números ou, em algumas abordagens, cores. Quando tanto os vértices quanto as arestas apresentam rótulos, o grafo é denominado totalmente rotulado. A Figura 3 ilustra um exemplo desse tipo de grafo, no qual possuem rótulos numéricos associados. (AIRES, 2015).

Denomina-se complemento de um grafo $G(V, E)$, o grafo $\bar{G}(V, \bar{E})$, que possui o mesmo conjunto de vértices V. Para todo par de vértices distintos $\{u, v\}$, a aresta $\{u, v\}$ pertence a $\bar{E}$ se, e somente se, essa aresta não pertence a E (MACAMBIRA et al., 2022).

Por exemplo, a Figura 4 ilustra essa relação: a Figura 4b apresenta o complemento do

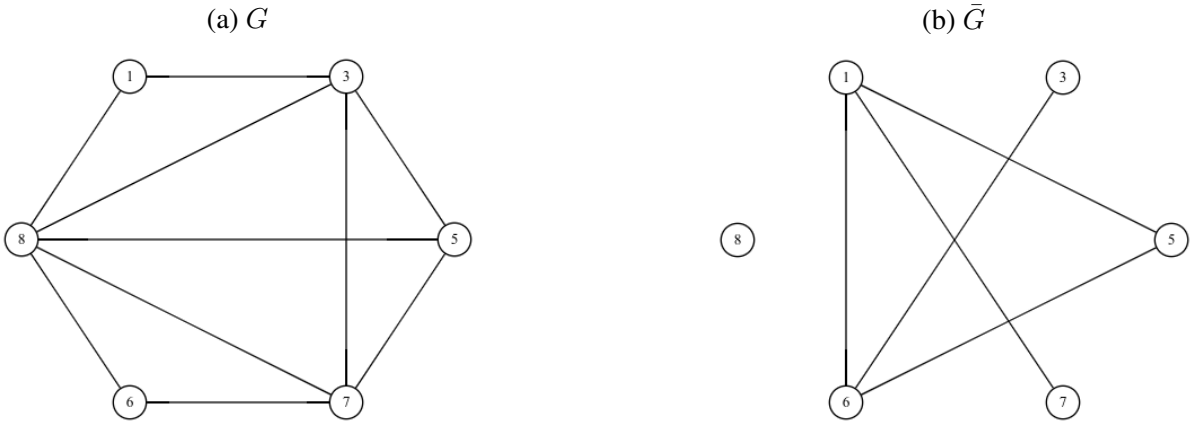
Figura 3 – Representação de um grafo rotulado.



Fonte: Modificado de Aires (2015).

grafo mostrado na Figura 4a.

Figura 4 – Representação de um grafo e de seu complemento.



Fonte: Modificado de Macambira et al. (2022).

Quando uma aresta está ligada a um vértice, diz-se que ela é incidente ou adjacente a esse vértice. O número de arestas incidentes determina o grau do vértice, representado por d. Por exemplo, no grafo da Figura 4a, o vértice 1 possui grau 2, ou seja, $d = 2$ (AIRES, 2015).

No estudo de grafos, destacam-se dois tipos importantes de grau: o grau mínimo e o grau máximo. O grau mínimo de um grafo G, denotado por $\delta(G)$, corresponde ao menor número de arestas incidentes a qualquer vértice do grafo. Por sua vez, o grau máximo, representado por $\Delta(G)$, é o maior número de arestas incidentes a algum vértice de G. No grafo da Figura 4b, observa-se que o vértice 8 não possui nenhuma aresta conectada, ou seja, seu grau é zero. Dessa forma, o grau mínimo do grafo G é $\delta(G) = 0$. Por outro lado, o vértice 1 apresenta três arestas incidentes, sendo, portanto, o vértice de maior grau no grafo. Assim, o grau máximo de G é $\Delta(G) = 3$ (FEOFILOFF; KOHAYAKAWA; WAKABAYASHI, 2004).

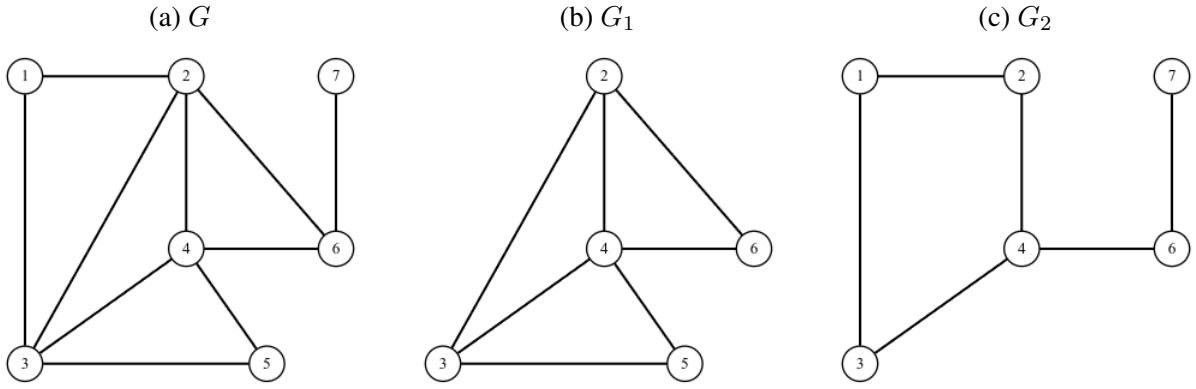
Diz-se que $G_1(V_1, E_1)$ é subgrafo de $G(V, E)$ se, e somente se, $V_1 \subseteq V$ e $E_1 \subseteq E$. Quando G_1 é um subconjunto próprio de G (isto é, $G_1 \neq G$), diz-se que G_1 é um subgrafo próprio, denotado por $G_1 \subset G$ (FREIRE, 2024).

Dado um subconjunto de vértices $V_1 \subseteq V$ de um grafo $G(V, E)$, o subgrafo induzido por V_1 é o grafo $G_1(V_1, E_1)$, cujo conjunto de arestas E_1 é formado por todas as arestas de E que possuem ambos os extremos em V_1. Formalmente, $E_1 = \{(u,v) \in E \mid u,v \in V_1\}$ (MACAMBIRA et al., 2022).

A Figura 5a ilustra o grafo original G e dois subgrafos derivados: G_1 e G_2. O grafo G_1, apresentado na Figura 5b, é um subgrafo induzido, pois contém um subconjunto de vértices de G, neste caso, $\{2, 3, 4, 5, 6\}$, e todas as arestas do grafo original que conectam pares desses vértices. Em outras palavras, nenhuma conexão entre os vértices selecionados foi omitida, o que caracteriza um subgrafo induzido por vértices.

Por outro lado, o grafo G_2, mostrado na Figura 5c, é um subgrafo próprio de G, mas não é um subgrafo induzido. Isso ocorre porque, por exemplo, os vértices 2 e 3 pertencem a G_2 e $(2,3)$ é uma aresta de G, mas não de G_2.

Figura 5 – Exemplos de subgrafo induzido e subgrafo próprio.



Fonte: Modificado de Macambira et al. (2022).

Além disso, um subgrafo $G_1(V_1, E_1)$ é denominado subgrafo gerador de um grafo $G(V, E)$ se $V_1 = V$, isto é, se contém todos os vértices de G. Nessa configuração, o subgrafo preserva a estrutura de vértices do grafo original, podendo, entretanto, apresentar um subconjunto das arestas (FREIRE, 2024).

Na Figura 6, o grafo G mostrado na Figura 6a representa o grafo original, enquanto o grafo G_1, apresentado na Figura 6b, é um subgrafo gerador de G. Observa-se que G_1 contém todos os vértices de G, porém apenas um subconjunto das arestas, satisfazendo assim a definição de subgrafo gerador.

Denomina-se clique de um grafo G um subgrafo completo, ou seja, um conjunto de vértices no qual todos os pares distintos de vértices estão conectados por uma aresta. Formalmente, um conjunto $L_n \subseteq V$ é uma clique se o subgrafo induzido por L_n, denotado por $G[L_n]$, é um grafo não atribuido. Nesse contexto, a notação L_n representa um grafo completo com n vértices, ou seja, um subgrafo em que todos os vértices estão mutuamente conectados. O tamanho de uma clique é definido pelo número de vértices que a compõem (MACAMBIRA et al., 2022).

Figura 6 – Exemplo de subgrafo gerador.

Fonte: Modificado de Freire (2024).

Uma clique maximal é uma clique que não pode ser expandida, isto é, não existe vértice fora do conjunto que possa ser adicionado a ela sem que ela deixe de ser um subgrafo completo. Em outras palavras, uma clique L_n é maximal se não existe um conjunto estritamente maior que contenha L_n e que também seja uma clique. Vale destacar que toda clique maximal é uma clique, porém nem toda clique é maximal (LOZADA, 1996).

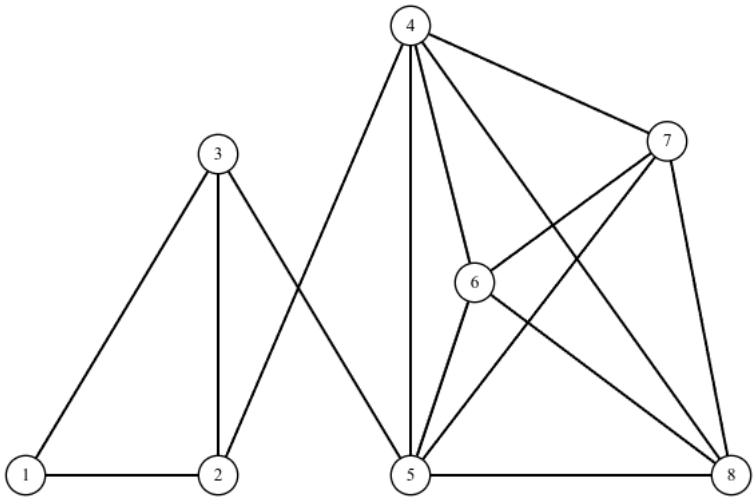
A clique máxima de um grafo G é a clique de maior tamanho possível em G. O tamanho da clique máxima é denotado por $\omega(G)$, que representa o maior número de vértices que formam uma clique em G. É importante destacar que pode haver mais de uma clique máxima em um grafo. Isso significa que podem existir diferentes conjuntos de vértices, cada um formando uma clique com o tamanho máximo, porém compostos por vértices distintos. Em outras palavras, a clique máxima não é necessariamente única, mas todas as cliques máximas possuem o mesmo número de vértices (LOZADA, 1996).

Conforme ilustrado na Figura 7, considera-se o grafo $G(V, E)$, com $V = \{1,2,3,4,5,6,7,8\}$. O conjunto $\{1,2,3\}$ forma uma clique maximal de tamanho 3, pois não pode ser expandida sem perder a completude. Já o conjunto $\{4,5,6,7,8\}$ é uma clique máxima, pois possui o maior número de vértices conectados entre si. Portanto, neste grafo, a maior clique identificada tem tamanho 5, definindo $\omega(G) = 5$. Dentro dessa clique maior, existem outras cliques menores, uma vez que qualquer subconjunto de vértices adjacentes também constitui um subgrafo completo. Esse exemplo demonstra que um grafo pode conter várias cliques maximais e cliques de diferentes tamanhos.

Denomina-se conjunto independente (ou conjunto estável) de um grafo G qualquer subconjunto de vértices $S \subseteq V$ tal que não existam dois vértices adjacentes contidos em S. Em outras palavras, o subgrafo induzido por S é sem arestas. A cardinalidade de um conjunto independente corresponde ao número de vértices que ele contém (MACAMBIRA et al., 2022).

Um conjunto independente maximal é um conjunto independente que não pode ser ampliado pela inclusão de outros vértices sem perder a propriedade de independência. Em outras palavras, trata-se de um conjunto que não está contido em nenhum outro conjunto independente estritamente maior (ISERNHAGEN; BARBOSA, 2006).

Figura 7 – Um grafo que possui cliques de diferentes tamanhos.

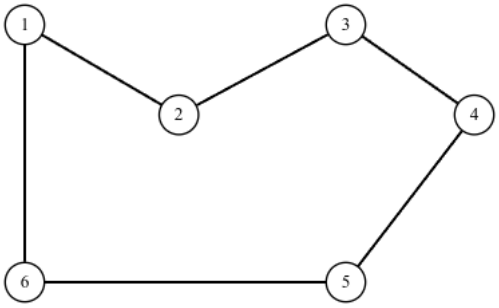


Fonte: Modificado de Pinto e Silva (2021).

Já um conjunto independente máximo, representado por $\alpha(G)$, é o conjunto independente de maior cardinalidade entre todos os conjuntos independentes possíveis no grafo G (ISERNHAGEN; BARBOSA, 2006).

Conforme ilustrado na Figura 8, onde o conjunto de vértices é $V = \{1,2,3,4,5,6\}$ e o conjunto de arestas é $E = \{(1,2),(2,3),(3,4),(4,5),(5,6),(6,1)\}$. Nesse grafo, o conjunto $\{1,3,5\}$ forma um conjunto independente máximo, pois contém o maior número de vértices não adjacentes entre si, definindo $\alpha(G) = 3$. Outros conjuntos independentes também podem ser identificados, como $\{2,4,6\}$. Além disso, pares como $\{1,4\}$, $\{2,5\}$ e $\{3,6\}$ são independentes maximais.

Figura 8 – Um grafo com exemplos de conjuntos independentes.



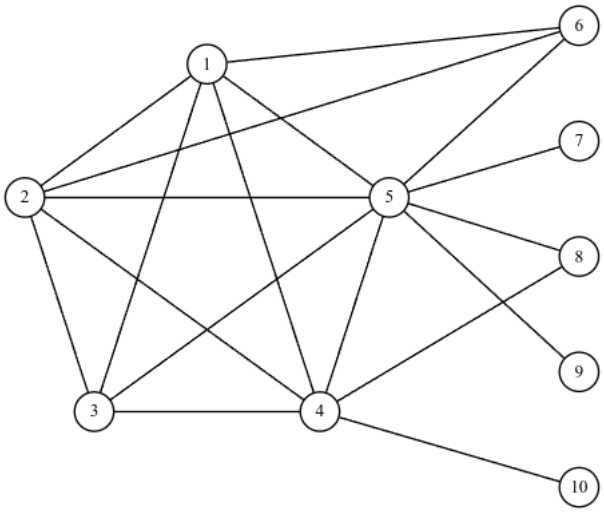
Fonte: Modificado de Cordeiro (2015).

Alguns conceitos importantes em teoria dos grafos envolvem as noções de clique e conjunto independente, destacando-se, nesse contexto, os grafos split e os grafos bipartidos. Um grafo split é aquele cujos vértices podem ser particionados em dois conjuntos: um que forma uma clique e outro que forma um conjunto independente (HAMMER; SIMEONE, 1981).

Conforme o grafo apresentado na Figura 9, é possível identificar a divisão entre os dois subconjuntos de vértices. À esquerda, observa-se uma clique formada pelos vértices

$\{1, 2, 3, 4, 5\}$, onde todos os vértices estão conectados entre si. À direita, encontram-se os vértices $\{6, 7, 8, 9, 10\}$, que formam um conjunto independente, ou seja, não há arestas ligando qualquer par desses vértices.

Figura 9 – Um grafo split.

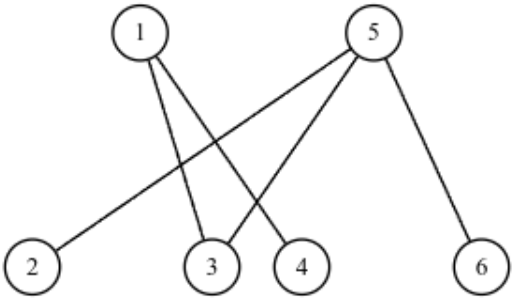


Fonte: Modificado de Freire (2024).

Já os grafos bipartidos são aqueles cujo conjunto de vértices pode ser particionado em dois conjuntos independentes, de modo que não existam arestas entre vértices pertencentes ao mesmo conjunto (FREIRE, 2024).

Na Figura 10, tem-se um exemplo de grafo bipartido. Os vértices estão particionados em dois grupos: o primeiro com os vértices $\{1, 5\}$ e o segundo com os vértices $\{2, 3, 4, 6\}$. As arestas ocorrem apenas entre vértices de grupos distintos, sem conexões internas em cada grupo, o que caracteriza a presença de dois conjuntos independentes e a ausência de cliques.

Figura 10 – Um grafo bipartido.



Fonte: Modificado de Aires (2015).

2.2 Complexidade Computacional

Nesta seção, são apresentados exemplos de problemas cuja complexidade computacional é conhecida, com ênfase nas classes P e NP-completo, destacando casos representativos de

problemas considerados tratáveis e intratáveis.

A complexidade computacional é o ramo da ciência da computação que investiga as razões pelas quais certos problemas são difíceis de serem resolvidos por computadores. Ela classifica os problemas de acordo com o tempo necessário para sua solução por algoritmos determinísticos, isto é, algoritmos que, para uma mesma entrada, sempre produzem a mesma saída e seguem uma sequência fixa de execução. A classe P engloba os problemas que podem ser resolvidos em tempo polinomial, sendo, portanto, considerados tratáveis na prática. Já a classe NP reúne problemas para os quais não se conhece um algoritmo determinístico eficiente de resolução, mas cuja verificação de uma solução candidata pode ser feita em tempo polinomial. A classe NP-completo abrange os problemas mais difíceis dentro de NP. Esses problemas têm uma propriedade importante: a resolução de qualquer um deles em tempo polinomial implicaria a resolução de todos os problemas de NP em tempo polinomial. Por essa razão, os problemas NP-completo são considerados os mais desafiadores da classe NP (AIRES, 2015).

2.2.1 Problemas de Complexidade Polinomial

Nesta seção, são apresentadas algumas classes de problemas pertencentes à classe P, ou seja, problemas que podem ser solucionados por algoritmos determinísticos que executam em tempo polinomial.

2.2.1.1 Busca em Largura (BFS)

A determinação das componentes conexas de um grafo pode ser realizada em tempo polinomial. Para essa tarefa, emprega-se a Busca em Largura (BFS), que percorre o grafo por níveis, iniciando a exploração a partir de um vértice escolhido. A BFS visita todos os vizinhos imediatos antes de avançar para os vértices dos níveis seguintes, utilizando uma fila para manter a ordem de exploração (FARIA, 2024).

A BFS explora as arestas de forma sistemática até identificar todos os vértices alcançáveis a partir da fonte. O algoritmo determina a menor distância, em número de arestas, da fonte até cada vértice e constrói uma árvore de busca em largura enraizada nesse vértice inicial. Cada vértice recebe uma cor para indicar seu estado de descoberta: branco para vértices ainda não descobertos, cinza para vértices descobertos que ainda possuem vizinhos a serem explorados e preto para vértices totalmente explorados. A estrutura de fila é essencial para o funcionamento do algoritmo e, garantindo que os vértices sejam processados por níveis. Durante a execução, todos os vértices a uma distância d do vértice fonte são explorados antes dos de distância $d + 1$, garantindo que o BFS encontre o menor caminho em número de arestas a partir de um vértice fixo. O algoritmo funciona em grafos direcionados ou não direcionados e possui complexidade $O(n + m)$ (CORMEN et al., 2001).

O Algoritmo 1 apresenta o procedimento da BFS para percorrer um grafo a partir de um vértice fonte s, escolhido arbitrariamente entre os vértices de V. Na etapa de inicialização, o algoritmo percorre todos os vértices do grafo, marca cada um deles como branco, atribui a eles

uma distância infinita e seus predecessores nulos. Para o vértice escolhido como fonte, a cor é alterada para cinza, a distância é definida como zero e o predecessor é definido como nulo. Em seguida, esse vértice é inserido na fila, dando início ao processo de exploração do grafo. Durante a execução, a variável u representa o vértice que está sendo processado no momento, ou seja, aquele que acabou de ser removido da fila, enquanto a variável v representa cada vizinho de u que será analisado (CORMEN et al., 2001).

Em cada iteração, o primeiro elemento da fila é removido pela operação DESENFILEIRA e armazenado na variável u, que corresponde ao vértice atualmente em processamento. Em seguida, são analisados todos os vértices v adjacentes a u. Se um vértice v estiver branco, ele é marcado com cinza, sua distância é atualizada para um nível acima de u, e v é inserido no final da fila por meio da operação ENFILEIRA. Quando todos os vizinhos de u são processados, u é marcado como preto, sinalizando que sua exploração foi concluída (CORMEN et al., 2001).

Algoritmo 1: Busca em Largura (BFS)

```
Entrada: G(V,E), vértice fonte s ∈ V
Saída: d e π a partir de s
1  início
2      para u ∈ V, u ≠ s faça
3          cor[u] ← BRANCO
4          d[u] ← ∞
5          π[u] ← NULL
6      fim
7      cor[s] ← CINZA
8      d[s] ← 0
9      π[s] ← NULL
10     Q ← ∅
11     ENFILEIRA(Q, s)
12     enquanto Q ≠ ∅ faça
13         u ← DESENFILEIRA(Q)
14         para v ∈ Adj[u] faça
15             se cor[v] = BRANCO então
16                 cor[v] ← CINZA
17                 d[v] ← d[u] + 1
18                 π[v] ← u
19                 ENFILEIRA(Q, v)
20             fim
21         fim
22         cor[u] ← PRETO
23     fim
24     retorna d, π
25 fim
```

Conforme apresentado na Tabela 1, baseada na Figura 10, na página 10, a BFS pode ser exemplificada considerando o vértice 1 como fonte. Na primeira tabela, antes da seta, o vértice 1 é marcado como cinza, com distância igual a zero e sem predecessor, enquanto os demais

vértices permanecem brancos, com distância infinita e predecessores nulos. Nesse momento, a fila contém apenas o vértice 1. Durante a execução, cada vértice é removido da fila, marcado como preto e seus vizinhos brancos são descobertos, passando a ter cor cinza, distância atualizada e predecessor correspondente, sendo então enfileirados. Esse processo se repete até que todos os vértices alcançáveis sejam explorados. Na segunda tabela, ao lado da primeira e após a seta, observa-se o estado final da execução. Ao final, todos os vértices apresentam cor preta, distância mínima e predecessores definidos, e a fila encontra-se vazia.

Tabela 1 – Execução do BFS.

	cor	d	π			cor	d	π
1	CINZA	0	NULL		1	PRETO	0	NULL
2	BRANCO	∞	NULL		2	PRETO	3	5
3	BRANCO	∞	NULL	$\rightarrow$	3	PRETO	1	1
4	BRANCO	∞	NULL		4	PRETO	1	1
5	BRANCO	∞	NULL		5	PRETO	2	3
6	BRANCO	∞	NULL		6	PRETO	3	5
Q	1				Q	~~1,3,4,5,2,6~~		

Fonte: Autoria própria.

2.2.1.2 Busca em Profundidade (DFS)

Mais um exemplo de problema resolvido em tempo polinomial é a identificação das componentes conexas de um grafo por meio da busca em profundidade (DFS). Nesse método, o algoritmo explora sistematicamente os vértices, avançando o máximo possível a partir de cada ponto antes de retroceder. Esse processo é implementado de forma natural por meio da recursão: sempre que um vértice não possui mais arestas a serem percorridas, a execução retorna ao vértice descoberto mais recentemente que ainda possui caminhos não explorados.Dessa forma, todos os vértices alcançáveis a partir de uma origem são visitados, caracterizando uma componente conexa (FREITAG, 2025).

Assim como na busca em largura, os vértices são coloridos durante a busca para indicar seu estado. Cada vértice é inicialmente branco, passa a cinza quando é descoberto e torna-se preto quando sua lista de adjacências é totalmente examinada. Essa estratégia garante que cada vértice tenha seu predecessor definido e que a ordem de exploração seja corretamente registrada. Além disso, a busca em profundidade também registra o tempo de descoberta e finalização de cada vértice. A complexidade desse algoritmo é $O(n + m)$ (CORMEN et al., 2001).

O Algoritmo 2 percorre todos os vértices do grafo, inicializando cada um como branco e com predecessor nulo. Em seguida, o tempo é zerado. Para cada vértice u, que representa o vértice atualmente analisado no laço principal, verifica-se a sua cor. Caso u esteja branco, significa que ainda não foi visitado, e então é chamado o procedimento DFS-VISITA (Algoritmo 3) para iniciar a exploração a partir desse vértice (CORMEN et al., 2001).

Algoritmo 2: Busca de Profundidade (DFS)

```
   Entrada: G(V,E)
   Saída: π, d, f
 1 início
 2     para u ∈ V faça
 3         cor[u] ← BRANCO
 4         π[u] ← NULL
 5     fim
 6     tempo ← 0
 7     para u ∈ V faça
 8         se cor[u] = BRANCO então
 9             DFS-VISITA(u)
10         fim
11     fim
12 fim
```

O Algoritmo 3, realiza a exploração recursiva de um vértice. Inicialmente, ele marca o vértice como cinza, atualiza o tempo e registra o instante de descoberta $d[u]$. Em seguida, para cada vértice adjacente a u, verifica se ele ainda está branco. Caso esteja, define u como predecessor desse vértice e chama recursivamente o DFS-VISITA para continuar a exploração a partir dele. Após explorar todos os vizinhos brancos, o vértice u é marcado como preto, o tempo é novamente incrementado e o instante de finalização $f[u]$ é registrado. Dessa forma, cada chamada do DFS-VISITA contribui para determinar a ordem de descoberta e finalização dos vértices, bem como a relação de precedência entre eles (CORMEN et al., 2001).

Algoritmo 3: DFS-VISITA

```
   Entrada: G(V,E), u
   Saída: π, d, f
 1 início
 2     tempo ← tempo + 1
 3     d[u] ← tempo
 4     cor[u] ← CINZA
 5     para v ∈ Adj[u] faça
 6         se cor[v] = BRANCO então
 7             π[v] ← u
 8             DFS-VISITA(v)
 9         fim
10     fim
11     cor[u] ← PRETO
12     tempo ← tempo + 1
13     f[u] ← tempo
14 fim
```

Conforme apresentado na Tabela 2 e ilustrado na Figura 10, na página 10, na primeira tabela, antes da seta, no DFS todos os vértices começam brancos, com predecessores $\pi[u] =$

NULL e tempo inicial igual a zero. Qualquer vértice branco pode ser escolhido para iniciar a busca; neste exemplo, foi selecionado o vértice 1. Durante a execução, cada vértice é visitado pela procedimento DFS-VISITA, sendo marcando como cinza, registrando o tempo de descoberta $d[u]$ e explorando recursivamente seus adjacentes brancos. Na segunda tabela, ao lado da primeira e após a seta, observa-se o estado final da execução. Ao final da exploração, o vértice é marcado como preto, o tempo de finalização $f[u]$ é registrado e todos os predecessores $\pi[v]$ são definidos. Quando a busca termina, todos os vértices apresentam cor preta, tempos de descoberta e finalização determinados, além de predecessores atribuídos.

Tabela 2 – Execução do DFS.

	cor	d	f	π			cor	d	f	π
1	BRANCO	-	-	NULL		1	PRETO	1	12	NULL
2	BRANCO	-	-	NULL		2	PRETO	4	5	5
3	BRANCO	-	-	NULL	$\rightarrow$	3	PRETO	2	9	1
4	BRANCO	-	-	NULL		4	PRETO	10	11	1
5	BRANCO	-	-	NULL		5	PRETO	3	8	3
6	BRANCO	-	-	NULL		6	PRETO	6	7	5
tempo = 0						tempo = 12				

Fonte: Autoria própria.

2.2.1.3 2-SAT

Entre os problemas que podem ser resolvidos em tempo polinomial, destaca-se o 2-SAT, que corresponde a uma restrição do problema de Satisfatibilidade Booleana (SAT). O SAT é um problema de decisão no qual a instância consiste em uma expressão booleana composta por variáveis, parênteses e operadores lógicos AND ($\wedge$), OR ($\vee$) e NOT ($^{-}$). O objetivo é determinar se existe alguma atribuição de valores verdadeiros ou falsos para as variáveis de forma que toda a expressão verdadeira (FREITAG, 2025).

No caso específico do 2-SAT (ou 2-satisfatibilidade), cada cláusula da fórmula contém no máximo dois literais, isto é, duas variáveis ou negações dessas variáveis. Essa restrição estrutural possibilita a aplicação de algoritmos especializados capazes de resolver o problema de forma eficiente. Como resultado, o 2-SAT pode ser solucionado em tempo polinomial, com complexidade $O(n + m)$ (FREITAG, 2025).

Uma abordagem eficiente para resolver o problema 2-SAT foi apresentada por Aspvall, Plass e Tarjan (1979), utilizando o algoritmo de Tarjan. Esse algoritmo identifica componentes fortemente conexas (SCCs) em grafos direcionados, permitindo detectar ciclos que indicam se a fórmula é satisfatível ou não. O método baseia-se na transformação da expressão booleana em um grafo de implicação, no qual cada variável e sua negação correspondem a vértices, enquanto as cláusulas são representadas por arestas direcionadas.

Para a resolução do problema 2-SAT, conforme apresentado por Aspvall, Plass e Tarjan (1979), é necessário que a fórmula esteja representada na forma 2-CNF (Forma Normal

Conjuntiva, com cláusulas contendo até dois literais). Uma fórmula está na forma 2-CNF quando é expressa como uma conjunção (operador lógico AND) de cláusulas, em que cada cláusula contém, no máximo, dois literais. Essa estrutura padronizada permite a aplicação direta do método de construção do grafo de implicação, utilizado para determinar a satisfatibilidade da fórmula.

Formalmente, uma Fórmula 2-CNF pode ser escrita como: $(\ell_1 \vee \ell_2) \wedge (\ell_3 \vee \ell_4) \wedge \ldots$, em que cada ℓ_n representa um literal. Por exemplo, considere a Fórmula C:

$$C = (x_1 \vee \overline{x}_2) \wedge (\overline{x}_1 \vee x_2) \wedge (\overline{x}_1 \vee \overline{x}_2) \wedge (x_1 \vee \overline{x}_3).$$

Neste caso, cada cláusula contém exatamente dois literais, que podem ser variáveis booleanas ou suas negações (ASPVALL; PLASS; TARJAN, 1979).

Para a resolução do problema 2-SAT, é fundamental transformar cada cláusula da fórmula em implicações lógicas, conforme apresentado por Aspvall, Plass e Tarjan (1979). Dada uma cláusula da forma $(l_1 \vee l_2)$, podem-se derivar duas implicações logicamente equivalentes: $\bar{l}_1 \rightarrow l_2$ e $\bar{l}_2 \rightarrow l_1$. Assim, para cada cláusula são geradas duas implicações. Na primeira, o primeiro literal é considerado falso e o segundo permanece inalterado, resultando em uma implicação em que o primeiro aparece negado. Na segunda, ocorre o inverso: o segundo literal é considerado falso e o primeiro permanece inalterado, gerando uma implicação em que o segundo aparece negado. Caso o literal original já esteja negado, o valor lógico se inverte na implicação, tornando-o positivo. Essa transformação bidirecional é o que permite representar cada cláusula como duas implicações no grafo de implicação.

No exemplo da Fórmula C, a Tabela 3 apresenta as implicações lógicas geradas a partir de cada cláusula, evidenciando como cada uma delas é convertida em duas implicações direcionais no grafo de implicação.

Tabela 3 – Implicações do Problema 2-SAT.

Cláusula	Implicação 1	Implicação 2
$(x_1 \vee \overline{x}_2)$	$\overline{x}_1 \rightarrow \overline{x}_2$	$x_2 \rightarrow x_1$
$(\overline{x}_1 \vee x_2)$	$x_1 \rightarrow x_2$	$\overline{x}_2 \rightarrow \overline{x}_1$
$(\overline{x}_1 \vee \overline{x}_2)$	$x_1 \rightarrow \overline{x}_2$	$x_2 \rightarrow \overline{x}_1$
$(x_1 \vee \overline{x}_3)$	$\overline{x}_1 \rightarrow \overline{x}_3$	$x_3 \rightarrow x_1$

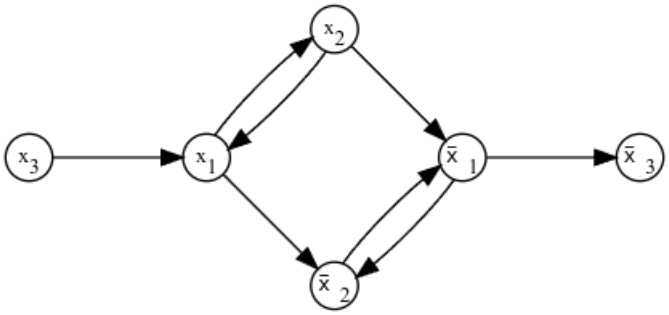
Fonte: Autoria própria.

Após a transformação das cláusulas em implicações lógicas, o próximo passo na resolução do problema 2-SAT consiste na construção do grafo de implicação. Nesse grafo, cada literal e sua negação correspondem a vértices distintos. Assim, para n variáveis booleanas, são criados $2n$ vértices, representando $l_1, \bar{l}_1, \quad l_2, \bar{l}_2, \quad \ldots, \quad l_n, \bar{l}_n$. As arestas direcionadas desse grafo representam às implicações derivadas de cada cláusula. Especificamente, cada implicação da forma $l_1 \rightarrow l_2$ é representada por uma aresta que parte do vértice l_1 e aponta para o vértice l_2 . Dessa forma, o grafo de implicação concentra todas as restrições lógicas da fórmula, permitindo

a análise estrutural necessária para determinar a satisfatibilidade (ASPVALL; PLASS; TARJAN, 1979).

No exemplo da Fórmula C, a Figura 11 ilustra o grafo de implicações construído a partir das cláusulas correspondentes. Nesse grafo, cada vértice representa um literal ou sua negação, enquanto cada aresta direcionada corresponde a uma implicação lógica derivada da transformação da fórmula. Essa representação gráfica concentra todas as restrições impostas pelas cláusulas e constitui a base para a análise de satisfatibilidade no problema 2-SAT.

Figura 11 – Grafo de implicações do problema 2-SAT.



Fonte: Autoria própria.

Após a construção do grafo de implicações, é necessário identificar as componentes fortemente conexas (SCCs), que correspondem a conjuntos de vértices nos quais cada vértice pode alcançar todos os outros por meio de caminhos direcionados. Essa etapa permite analisar a estrutura do grafo e verificar condições de satisfatibilidade. A identificação das SCCs pode ser realizada de forma eficiente por meio do algoritmo de Tarjan (ASPVALL; PLASS; TARJAN, 1979).

O algoritmo de Tarjan, proposto por Tarjan (1972), identifica as componentes fortemente conexas (SCCs) de um grafo direcionado em uma única busca em profundidade. A complexidade do algoritmo é $O(n+m)$. Cada vértice recebe um número de descoberta $d[u]$ e um valor mínimo $low[u]$, que representa o menor número de descoberta alcançável a partir dele. O algoritmo utiliza uma pilha para armazenar os vértices que ainda não foram atribuídos a uma SCC, garantindo que todos os vértices de um mesmo componente permaneçam agrupados até sua identificação. Durante a execução, para cada vértice que ainda não foi visitado, a busca em profundidade localiza todos os vértices que pertencem ao mesmo componente fortemente conexo. Quando esse componente é completamente identificado, todos os seus vértices são removidos da pilha e registrados como uma nova SCC.

O Algoritmo 4 apresenta esse procedimento. No início da execução, a variável que representa o contador de vértices visitados é inicializada com zero e a pilha M é criada vazia e a estrutura SCCs também começa vazia. Para cada vértice u, o valor de $d[u]$ e $low[u]$ são definidos como infinito, indicando que o vértice não foi descoberto. Além disso, a variável onStack recebe falso para todos os vértices, mostrando que nenhum deles está na pilha no início. Em seguida, para cada vértice u no grafo, se $d[u]$ seja igual a infinito, a função TARJAN-VISITA, apresentada

no Algoritmo 5, é chamada para iniciar a exploração a partir desse vértice, passando também a estrutura SCCs. A escolha do vértice inicial pode ser qualquer um, já que o algoritmo garantirá que todos os vértices do grafo sejam visitados no final da execução.

Algoritmo 4: Tarjan

```
   Entrada: G(V,E)
   Saída: SCCs
 1 início
 2 |  visitado ← 0
 3 |  M ← ∅
 4 |  SCCs ← ∅
 5 |  para u ∈ V faça
 6 |  |  d[u] ← ∞
 7 |  |  low[u] ← ∞
 8 |  |  onStack[u] ← FALSO
 9 |  fim
10 |  para u ∈ V faça
11 |  |  se d[u] = ∞ então
12 |  |  |  TARJAN-VISITA(u, SCCs)
13 |  |  fim
14 |  fim
15 fim
```

O Algoritmo 5, correspondente ao TARJAN-VISITA, inicia atribuindo ao vértice atual o número armazenado em visitado. Tanto $d[u]$ quanto $low[u]$ recebem esse valor, e em seguida visitado é incrementado em uma unidade. O vértice u é empilhado na pilha M e marcado como ativo, com onStack[u] definido como verdadeiro. Em seguida, cada vértice adjacente v de u é analisado. Quando $d[v]$ está infinito, a função TARJAN-VISITA é chamada recursivamente para v. Ao retornar da chamada, $low[u]$, é atualizado com o menor valor entre $low[u]$ e $low[v]$. Caso contrário, quando v já foi visitado e ainda está na pilha, $low[u]$ é atualizado com o menor valor entre $low[u]$ e $d[v]$. Depois que todos os vértices adjacentes são processados, se $d[u]$ e $low[u]$ tiverem o mesmo valor, significa que u é a raiz de uma componente fortemente conexa. Nesse momento, é criada uma estrutura chamada componente, inicialmente vazia. Os vértices começam a ser removidos da pilha um por um e armazenados nessa estrutura, até que u seja desempilhado. O conjunto resultante corresponde a uma SCC completa, que é então inserida na estrutura SCCs. Esse procedimento se repete para todos os vértices do grafo, permitindo identificar todas as componentes fortemente conexas (TARJAN, 1972).

Conforme apresentado na Tabela 4, a execução do algoritmo de Tarjan para o grafo de implicação da Figura 11, na página 17. Na primeira tabela, antes da seta inicia-se com o contador visitado igual a zero, a pilha M vazia, todos os vértices com $d[u] = \infty$, $low[u] = \infty$, onStack definido como FALSO e a estrutura SCCs também vazia. A função TARJAN-VISITA é então chamada para iniciar a exploração. O vértice x_3 é escolhido como ponto inicial, recebendo $d[x_3] = low[x_3] = 0$, sendo marcado com onStack igual a VERDADEIRO e empilhado em

Algoritmo 5: TARJAN-VISITA

```
Entrada: G(V,E), u, SCCs
Saída: d, low, SCCs
1  início
2      d[u] ← visitado
3      low[u] ← visitado
4      visitado ← visitado + 1
5      Empilha(M, u)
6      onStack[u] ← VERDADEIRO
7      para v ∈ Adj[u] faça
8          se d[v] = ∞ então
9              TARJAN-VISITA(v, SCCs)
10             low[u] ← min(low[u], low[v])
11         fim
12         senão se onStack[v] = VERDADEIRO então
13             low[u] ← min(low[u], d[v])
14         fim
15     fim
16     se d[u] = low[u] então
17         componente ← ∅
18         repita
19             w ← Desempilha(M)
20             onStack[w] ← FALSO
21             Adiciona(componente, w)
22         até w = u;
23         Adiciona(SCCs, componente)
24     fim
25 fim
```

L. Durante a execução, a função TARJAN-VISITA é chamada recursivamente para os vértices adjacentes ainda não visitados (x_1, x_2, $\overline{x_1}$, $\overline{x_2}$ e $\overline{x_3}$), atualizando os valores de $d[u]$ e $low[u]$, marcando onStack como VERDADEIRO e empilhando cada vértice. Sempre que $d[u]$ e $low[u]$ assumem o mesmo valor, ocorre o desempilhamento de vértices até alcançar o vértice atual, formando assim uma nova componente fortemente conexa (SCC). Na segunda tabela, ao lado da primeira e após a seta, observa-se o estado final da execução. Ao final da execução, a pilha M está novamente vazia, onStack retorna para FALSO em todos os vértices e são identificadas as seguintes SCCs:$\{\overline{x}_3\}$, $\{\overline{x}_1, \overline{x}_2\}$,$\{x_1, x_2\}$, e $\{x_3\}$.

A Figura 12 apresenta o grafo das componentes fortemente conexas (SCCs) após a execução do algoritmo de Tarjan. Dessa forma, o algoritmo identifica de maneira sistemática todas as SCCs do grafo, permitindo a separação precisa dos vértices de acordo com suas interdependências. Caso uma variável e sua negação pertençam à mesma componente fortemente conexa, a fórmula é considerada insatisfatível, pois isso representa uma contradição lógica. Caso contrário, a fórmula é satisfatível, existindo ao menos uma atribuição de valores que a torna verdadeira (ASPVALL; PLASS; TARJAN, 1979).

Tabela 4 – Execução do Tarjan.

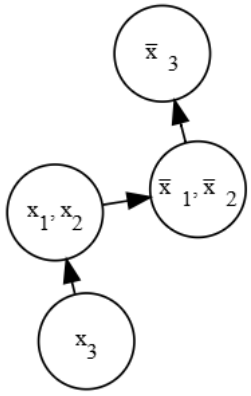
	d	low	onStack
x_1	∞	∞	FALSO
x_2	∞	∞	FALSO
x_3	∞	∞	FALSO
$\overline{x}_1$	∞	∞	FALSO
$\overline{x}_2$	∞	∞	FALSO
$\overline{x}_3$	∞	∞	FALSO
visitado = 0			
M = $\emptyset$			
SCCs = $\emptyset$			

$\rightarrow$

	d	low	onStack
x_1	1	1	FALSO
x_2	2	1	FALSO
x_3	0	0	FALSO
$\overline{x}_1$	3	3	FALSO
$\overline{x}_2$	5	3	FALSO
$\overline{x}_3$	4	4	FALSO
visitado = 6			
M = ~~$x_3, x_1, x_2, \overline{x}_1, \overline{x}_3, \overline{x}_2$~~			
SCCs = $\{\overline{x}_3\}$, $\{\overline{x}_1, \overline{x}_2\}$,$\{x_1, x_2\}$, e $\{x_3\}$			

Fonte: Autoria própria.

Figura 12 – Ordenação topológica das componentes fortemente conexas.



Fonte: Autoria própria.

Uma vez identificadas as componentes, elas são organizadas em ordem topológica, ou seja, em uma sequência na qual, se existir uma aresta de uma componente contendo l_1 para outra contendo l_2, então l_1 aparece antes de l_2 na ordenação. Essa ordenação pode ser obtida utilizando o Algoritmo 2, na página 14, aproveitando os tempos de finalização $f[u]$ que o próprio DFS calcula. Para isso, basta ordenar os vértices de acordo com o tempo de finalização, do maior para o menor (ASPVALL; PLASS; TARJAN, 1979).

O Algoritmo 6 apresenta um exemplo de como realizar a ordenação topológica. Primeiro, executa-se o DFS para obter os tempos de finalização $f[u]$ de cada vértice. Em seguida, os vértices são inseridos em uma lista e ordenados em ordem decrescente com base nesses tempos. Após essa ordenação, obtém-se a sequência topológica dos vértices utilizando o DFS (ASPVALL; PLASS; TARJAN, 1979).

Com base na Fórmula C e em suas componentes fortemente conexas (SCCs), apresentadas no grafo da Figura 12, na página 20, foi determinada a seguinte ordem topológica: $\{\overline{x}_3\} = 4$, $\{\overline{x}_1, \overline{x}_2\} = 3$,$\{x_1, x_2\} = 2$, e $\{x_3\} = 1$.

Por último, para realizar a atribuição dos valores lógicos às variáveis do 2-SAT, o Algoritmo 7 compara a posição na ordenação topológica da componente fortemente conexa

Algoritmo 6: Ordenação Topológica.

Entrada: $G(V,E)$
Saída: Ordem topológica
início
  Executar DFS(G) para obter $f[u]$ de cada $u \in V$;
  Lista $\leftarrow V$;
  ordenar(Lista, por $f[u]$ em ordem decrescente);
  retorna Lista;
fim

associada a cada variável l_n com o número da componente correspondente à sua negação $\overline{l_n}$. Relembrando que, caso uma variável e sua negação pertençam à mesma componente fortemente conexa, a fórmula é considerada insatisfatível. A regra de atribuição estabelece que, se a posição topológica de l_n é menor que a posição topológica de $\overline{l_n}$, a variável l_n recebe o valor falso; caso contrário, l_n recebe o valor verdadeiro (ASPVALL; PLASS; TARJAN, 1979).

Algoritmo 7: Atribuição de valores a l_n.

Entrada: Ordem Topológicas de l_n e Ordem Topológicas de $\bar{l}_n$
Saída: Valor lógico atribuído a l_n
início
  se *Ordem Topológica de* $(l_n) <$ *Ordem Topológica de* $(\overline{l_n})$ **então**
    $l_n \leftarrow$ *Falso*
  fim
  senão
    $l_n \leftarrow$ *Verdadeiro*
  fim
fim

Com base na Fórmula C apresentada, os valores lógicos das variáveis são determinados comparando a componente fortemente conexa de cada literal com a de sua negação. Para x_1, x_2 e x_3, verifica-se que Ordem Topológica(x_i) é menor que Ordem Topológica$(\overline{x_i})$, o que leva à atribuição do valor Falso para todas elas. Substituindo esses valores na fórmula original e considerando que $\overline{Falso} = Verdadeiro$, todas as cláusulas são satisfeitas. Assim, $C = (Verdadeiro) \wedge (Verdadeiro) \wedge (Verdadeiro) \wedge (Verdadeiro) = Verdadeiro$, confirmando que a atribuição obtida satisfaz toda a fórmula.

2.2.1.4 Grafo Split

Outra classe importante é a dos grafos split. Hammer e Simeone (1981) demonstram que esses grafos podem ser reconhecidos em tempo polinomial, com complexidade $O(n + m)$, por meio de uma fórmula baseada na sequência de graus dos vértices, utilizando o conceito de splittance. A splittance indica o número mínimo de arestas que precisam ser adicionadas ou

removidas para transformar um grafo qualquer em um grafo split. Quando o grafo já pertence a essa classe, o valor da splittance é igual a zero.

Hammer e Simeone (1981) propuseram que, dado um grafo $G(V, E)$ com n vértices, o índice $m(d)$ pode ser definido a partir da sequência de graus $d = (d_1, d_2, \ldots, d_n)$, ordenada do maior para o menor grau. Cada d_k representa o grau do vértice na posição k da sequência, com k assumindo valores inteiros de 1 até n. Aqui, k é apenas um índice de posição na sequência de graus, e não está relacionado a outros conceitos que também utilizam a letra k mais adiante neste trabalho. O operador máximo ($\max$) seleciona o maior valor de k que satisfaz a condição $d_k \geq k - 1$. Assim, $m(d)$ corresponde ao maior índice k cujo grau d_k é pelo menos $k - 1$, determinando o tamanho da clique máxima que pode ser formada no grafo split derivado do grafo original. A fórmula é dada por:

$$m(d) = \max\{k \mid d_k \geq k - 1\}.$$

Dada a Figura 9, na página 10, que ilustra um grafo candidato à classe dos grafos split, aplica-se o Teorema de Hammer e Simeone (1981) para verificar se ele pertence a essa classe, utilizando o cálculo da splittance. Primeiramente, extrai-se a sequência de graus dos vértices: $d = (5, 5, 4, 6, 8, 3, 1, 2, 1, 1)$. Após ordenar em ordem decrescente, obtém-se: $d = (8, 6, 5, 5, 4, 3, 2, 1, 1, 1)$. Em seguida, calcula-se $m(d)$, que corresponde ao maior valor de k - lembrando que aqui k é apenas o índice de posição na sequência de graus - que satisfaz $d_k \geq k-1$. Esse valor de k indica quantos vértices iniciais possuem grau suficiente para formar a clique máxima do grafo split.

$$d_1 = 8 \geq 0, \quad d_2 = 6 \geq 1, \quad d_3 = 5 \geq 2, \quad d_4 = 5 \geq 3, \quad d_5 = 4 \geq 4, \quad d_6 = 3 \ngeq 5$$

A desigualdade deixa de ser satisfeita a partir de d_6. Assim, conclui-se que $m(d) = 5$. Isso significa que os cinco primeiros vértices da sequência têm grau suficiente para formar a clique máxima do grafo split derivado.

Depois de determinar $m(d)$, Hammer e Simeone (1981) mostram que é possível calcular a splittance $\sigma(G)$, que representa o número mínimo de arestas que precisam ser adicionadas ou removidas para transformar o grafo G em um grafo split. A fórmula é dada por:

$$\sigma(G) = \frac{1}{2}\left(\left(m(d)(m(d)-1) - \sum_{i=1}^{m(d)} d_i\right) + \sum_{i=m(d)+1}^{n} d_i\right).$$

A fórmula é composta por duas partes complementam. A primeira parte, $m(d)(m(d) - 1) - \sum_{i=1}^{m(d)} d_i$, indica o déficit de arestas entre os vértices que deveriam formar a clique. Esse termo mostra quantas arestas faltam dentro da clique e também quantas dessas arestas acabam aparecendo ligadas aos vértices que deveriam pertencer ao conjunto independente. Por esse motivo, esse valor costuma ser negativo, pois representa exatamente o excesso de ligações inadequadas dos vértices da clique com o restante do grafo. A segunda parte da expressão, $\sum_{i=m(d)+1}^{n} d_i$ corresponde às arestas incidentes nos vértices que deveriam compor o conjunto

independente. Todas as arestas desses vértices são somadas, incluindo aquelas entre eles e também aquelas que os conectam aos vértices da clique. Isso produz um valor positivo, pois representa o total de arestas que precisam ser removidas para que esses vértices se tornem realmente independentes. Como a soma dos graus conta cada aresta duas vezes quando ambas as extremidades estão nesse subconjunto, esse valor tende naturalmente a ser maior. O fator $\frac{1}{2}$ corrige essa contagem dupla e garante que o resultado final represente exatamente o número mínimo de operações necessárias para transformar o grafo em um grafo split. Quando o grafo já é split, a parte negativa e a parte positiva da expressão se compensam, resultando em $\sigma(G) = 0$ o que significa que nenhuma modificação é necessária (HAMMER; SIMEONE, 1981).

Com base no exemplo da Figura 9, na página 10, em que foi determinado que $m(d) = 5$, calcula-se a splittance:

$$\sigma(G) = \frac{1}{2}\Big(\Big(5(5-1) - \sum_{i=1}^{5} d_i\Big) + \sum_{i=6}^{10} d_i\Big)$$

Primeiro, calcula-se o número de arestas necessárias para a clique máxima: $5(5-1) = 20$. Em seguida, a soma dos graus dos cinco primeiros vértices: $\sum_{i=1}^{5} d_i = 8+6+5+5+4 = 28$. Depois, a soma dos graus dos vértices restantes: $\sum_{i=6}^{10} d_i = 3+2+1+1+1 = 8$. Substituindo esses valores na fórmula da splittance, tem-se: $\sigma(G) = \frac{1}{2}(20-28)+8 = 0$. Como $\sigma(G) = 0$, conclui-se que a splittance do grafo é nula. Isso significa que não são necessárias adições ou remoções de arestas para que o grafo satisfaça as condições de um grafo split. Assim, confirma-se que o grafo representado na Figura 9, na página 10, pertence à classe dos grafos split, conforme estabelecido pelo Teorema de Hammer e Simeone (1981).

Considerando o grafo apresentado na Figura 1, na página 4, cuja sequência de graus é $d = (3,2,2,2,2,1,0)$, temos $m(d) = 3$. Assim, o cálculo da splittance fica:

$$\sigma(G) = \frac{1}{2}\Big(\Big(3(3-1) - \sum_{i=1}^{3}(3+2+2\Big) + \sum_{i=4}^{7}(2+2+2+1+0)\Big)$$

Portanto, $\sigma(G) = 2$. Esse resultado mostra que o grafo não pertence à classe dos grafos split. Para que ele satisfaça as condições dessa classe, são necessárias exatamente duas modificações na sua estrutura. No caso em análise, ao remover as arestas $\{1,2\}$ e $\{1,3\}$ o grafo passa a atender às propriedades de um grafo split.

2.2.1.5 Grafo Bipartido

Pode-se citar ainda o grafo bipartido, cuja verificação pode ser feita em tempo polinomial por meio do Algoritmo 1, na página 12, que corresponde ao algoritmo de busca em largura. Durante a execução, para cada aresta entre dois vértices, verifica-se se a distância de um vértice é diferente da distância do seu adjacente. Se todas as arestas respeitarem essa condição, o grafo é bipartido. Caso algum par de vértices adjacentes apresente a mesma distância, o grafo não é bipartido. A complexidade temporal do BFS é $O(n+m)$ (CORMEN et al., 2001).

O Algoritmo 8 exemplifica esse procedimento. Inicialmente, todos os vértices recebem distância infinita. Em seguida, para cada vértice escolhido como fonte, caso sua distância seja infinita, o BFS é chamado para iniciar a exploração a partir dele. Depois disso, para cada aresta (u, v) do grafo, verifica-se se a distância de u é igual à distância de v. Se essa condição ocorrer em alguma aresta, o algoritmo retorna falso, indicando que o grafo não é bipartido. Caso contrário, se nenhuma aresta violar a condição, o grafo é considerado bipartido.

Algoritmo 8: Bipartição.

```
Entrada: G(V,E)
Saída: VERDADEIRO ou FALSO
1  início
2      para u ∈ V faça
3          d[u] ← ∞
4      fim
5      para s ∈ V faça
6          se d[s] = ∞ então
7              BFS(s)
8          fim
9      fim
10     para cada aresta (u,v) ∈ E faça
11         se d[u] = d[v] então
12             retorna FALSO
13         fim
14     fim
15     retorna VERDADEIRO
16 fim
```

Considerando a Figura 10, na página 10, que ilustra um grafo bipartido, aplica-se o algoritmo para verificar a bipartição. O vértice 1 foi escolhido como vértice fonte, resultando nas seguintes distâncias: $d[1] = 0$, $d[2] = 3$, $d[3] = 1$, $d[4] = 1$, $d[5] = 2$ e $d[6] = 3$.

No exemplo da Figura 7, na página 9, foi aplicada a verificação da bipartição iniciando pelo vértice 1 como fonte. As distâncias obtidas foram $d[1] = 0$, $d[2] = 1$, $d[3] = 1$, $d[4] = 2$, $d[5] = 2$, $d[6] = 3$, $d[7] = 3$ e $d[8] = 3$. Na análise do vértice 1 com seus adjacentes (vértices 2 e 3), observou-se que as distâncias são diferentes, não havendo conflito. No entanto, ao verificar o vértice 2 com seu adjacente, que é o vértice 3, constatou-se que ambos têm a mesma distância. Isso já é suficiente para concluir que o grafo não é bipartido.

2.2.2 Problemas de Complexidade NP-Completo

Nesta seção, são apresentadas os problemas NP-Completo, que representam alguns dos maiores desafios da ciência da computação. Esses problemas não possuem algoritmos conhecidos que os resolvam de forma eficiente e, por isso, são fundamentais para o estudo da complexidade computacional.

2.2.2.1 SAT

O problema SAT (Satisfatibilidade Booleana) tem importância histórica por ter sido o primeiro problema identificado como NP-completo, conforme demonstrado por Cook (1971). Diferentemente do 2-SAT, citado na Subseção 2.2.1.3, na página 15, em que cada cláusula possui no máximo dois literais permitindo a construção de um grafo de implicações em que a presença de uma variável e sua negação na mesma SCC indica insatisfatibilidade e garante resolução em tempo polinomial, o SAT geral admite cláusulas com três ou mais literais. Essa característica impede uma transformação simples para grafo de implicações, fazendo com que a quantidade de combinações cresça rapidamente, o que torna o problema NP-completo.

O SAT pertence à classe NP, ou seja, é um problema para o qual, dada uma possível solução, é possível verificar sua correção em tempo polinomial. No caso do SAT, isso significa que, ao receber uma atribuição de valores para as variáveis, pode-se verificar rapidamente se a fórmula é satisfeita. Por exemplo, considere a fórmula $C = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x}_1 \vee x_4) \wedge (x_2 \vee \overline{x}_3)$, com a atribuição $x_1 = F$, $x_2 = V$, $x_3 = F$ e $x_4 = F$. Todas as cláusulas são satisfeitas, confirmando que a fórmula é verdadeira para essa atribuição. Isso mostra que, uma vez fornecida uma solução, sua validade pode ser verificada de forma eficiente, o que caracteriza os problemas pertencentes à classe NP (COOK, 1971).

O problema SAT também é considerado NP-difícil. Isso significa que ele é pelo menos tão difícil quanto qualquer outro problema pertencente à classe NP. Essa propriedade foi demonstrada por Cook (1971), que provou que qualquer problema da classe NP pode ser transformado em uma instância de SAT em tempo polinomial, ou seja, de forma eficiente.

Em termos mais práticos, Cook (1971) mostrou que é possível construir uma expressão booleana que simula o comportamento de qualquer máquina de Turing não determinística, um modelo teórico de computador capaz de executar todos os cálculos computáveis, considerando todas as possibilidades de execução ao mesmo tempo. A fórmula resultante é satisfatível se, e somente se, a máquina aceitar a entrada. Essa transformação é feita de modo eficiente, em tempo polinomial.

Portanto, se alguém encontrasse um algoritmo capaz de resolver o SAT em tempo polinomial, todos os problemas da classe NP também poderiam ser resolvidos em tempo polinomial. Por essa razão, o SAT é simultaneamente NP (pois é possível verificar uma solução em tempo polinomial) e NP-difícil (porque todos os problemas de NP podem ser reduzidos a ele), sendo assim classificado como NP-completo (COOK, 1971).

Como consequência, o SAT serve como problema de referência: muitos outros problemas são reduzidos a ele, o que permite demonstrar que esses problemas também pertencem à classe NP-completos (COOK, 1971).

2.2.2.2 3-SAT

Entre os problemas resolvidos em tempo não determinístico e pertencentes à classe dos NP-completos, destaca-se o 3-SAT. Assim como o 2-SAT, apresentado na Subseção 2.2.1.3, na página 15, o 3-SAT é uma variação do problema SAT geral. O problema SAT geral citado na Subseção 2.2.2.1, na página 25, recebe como entrada qualquer fórmula booleana expressa em forma normal conjuntiva (CNF), isto é, uma conjunção de cláusulas, onde cada cláusula é uma disjunção de literais (OLIVEIRA, 2004).

A diferença entre eles está na quantidade de literais por cláusula: no 2-SAT, cada cláusula contém duas literais, enquanto no 3-SAT, cada cláusula contém três literais. Essa diferença é fundamental, pois permite que o 2-SAT seja resolvido em tempo polinomial, enquanto o 3-SAT é NP-completo, ou seja, não existe algoritmo conhecido capaz de resolvê-lo de forma eficiente em tempo polinomial (OLIVEIRA, 2004).

Formalmente, uma fórmula em 3-CNF pode ser escrita como $(\ell_1 \vee \ell_2 \vee \ell_3) \wedge (\ell_4 \vee \ell_5 \vee \ell_6) \wedge \ldots$ em que cada ℓ_n representa uma literal, isto é, uma variável booleana ou a negação de uma variável. (OLIVEIRA, 2004).

Karp (1972) demonstrou que o problema 3-SAT também é NP-completo.Para isso, mostrou que qualquer instância do problema SAT geral pode ser transformada, em tempo polinomial, em uma instância equivalente do 3-SAT, isto é, uma fórmula em que cada cláusula contém exatamente três literais.

O processo consiste em ajustar as cláusulas do SAT para que todas tenham exatamente três literais. Quando uma cláusula possui mais de três literais, ela é dividida em várias cláusulas de três literais por meio da introdução de variáveis auxiliares. Já as cláusulas com menos de três literais são completadas com literais repetidos ou auxiliares até atingirem o formato necessário (KARP, 1972).

Por exemplo, a cláusula $(x_1 \vee x_2 \vee x_3 \vee x_4)$ pode ser transformada em duas cláusulas de três literais: $(x_1 \vee x_2 \vee y) \wedge (\overline{y} \vee x_3 \vee x_4)$, em que y é uma variável auxiliar. A nova fórmula em 3-SAT é satisfatível se, e somente se, a fórmula original também é satisfatível. A variável auxiliar y não altera a lógica da expressão, servindo apenas para permitir a divisão de cláusulas maiores em partes com três literais (KARP, 1972).

Uma atribuição de valores pode ilustrar essa equivalência. Quando $x_1 = F$, $x_2 = F$, $x_3 = V$ e $x_4 = V$, a cláusula original é verdadeira, e na versão 3-SAT ela também pode ser satisfeita com $y = V$. Já quando $x_1 = F$, $x_2 = F$, $x_3 = F$ e $x_4 = F$, a cláusula original é falsa e a versão 3-SAT também permanece insatisfatível, independentemente do valor atribuído a y. Dessa forma, a transformação mantém a lógica da fórmula e mostra que resolver o 3-SAT tem a mesma dificuldade que resolver o SAT geral, confirmando que o 3-SAT é NP-completo (KARP, 1972).

2.2.2.3 Problema do Clique

Outro problema considerado NP-completo é o Clique. Ele pertence à classe NP porque, dado um grafo G e um número t, é possível verificar em tempo polinomial se existe uma clique de tamanho t em G. Isso significa que, ao receber uma solução candidata, ou seja, um conjunto de vértices, é possível verificar rapidamente se todos os vértices desse conjunto são mutuamente adjacentes, formando uma clique. Essa capacidade de verificação eficiente é o que caracteriza o problema como pertencente à classe NP (SZWARCFITER, 2018).

Para provar que o problema Clique é NP-completo, considera-se uma redução do problema SAT, citado na Subseção 2.2.2.1, na página 25, para o problema da Clique. Parte-se de uma instância de SAT, composta por uma expressão booleana em forma normal conjuntiva (CNF) com p cláusulas, sendo p o número total de cláusulas da fórmula. Como exemplo, pode-se usar a fórmula $C = (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3})$ (SZWARCFITER, 2018).

A ideia da redução consiste em construir, a partir da expressão booleana, um grafo G de modo que G possua uma clique de tamanho p se, e somente se, a fórmula original é satisfatível. Cada cláusula da fórmula é representada por um conjunto de vértices, e cada literal, que pode ser uma variável ou sua negação, corresponde a um vértice no grafo. Os vértices pertencentes a cláusulas diferentes são conectados sempre que seus literais não forem contraditórios, isto é, não representarem simultaneamente uma variável e a sua negação (SZWARCFITER, 2018).

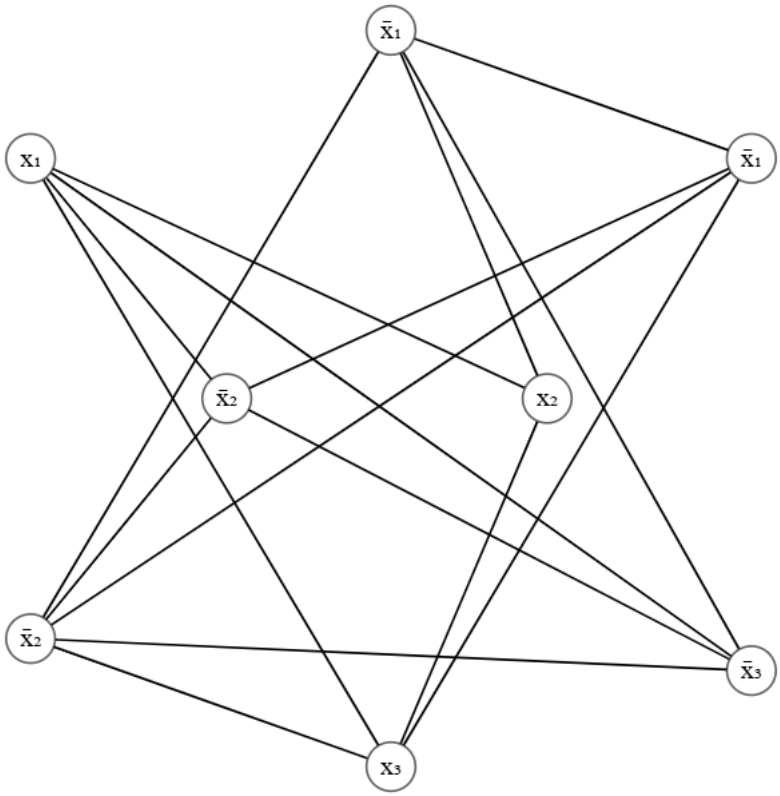
Na Figura 13, cada vértice representa um literal de uma cláusula da fórmula apresentada como exemplo. As arestas conectam literais que podem coexistir em uma mesma solução satisfatória. Assim, identificar uma clique de tamanho p no grafo equivale a selecionar um literal de cada cláusula de forma que todas sejam satisfeitas ao mesmo tempo, mostrando visualmente como a fórmula SAT é convertida em um problema de clique.

Assim, uma clique de tamanho p no grafo construído representa a escolha de um literal de cada cláusula que pode ser verdadeiro ao mesmo tempo, satisfazendo toda a fórmula booleana. No exemplo da Figura 13, a fórmula possui três cláusulas, logo $p = 3$. Cada vértice do grafo corresponde a uma literal, e as arestas conectam literais que não se contradizem, isto é, que não representam simultaneamente uma variável e a sua negação (SZWARCFITER, 2018).

No exemplo apresentado, ao atribuir $x_1 = V$, $x_2 = V$ e $x_3 = V$, todas as cláusulas da fórmula são satisfeitas. Essa atribuição corresponde a uma clique de tamanho $t = 3$, que coincide com o número de cláusulas p. Isso confirma que é possível selecionar um literal de cada cláusula de forma que não haja contradições, garantindo assim a satisfatibilidade da fórmula original (SZWARCFITER, 2018).

Dessa forma, a construção mostra que qualquer instância do SAT pode ser transformada, em tempo polinomial, em uma instância do problema Clique. Isso comprova que o Clique é NP-difícil. Como já foi demonstrado que o Clique pertence à classe NP, conclui-se que ele é NP-completo (SZWARCFITER, 2018).

Figura 13 – Grafo da redução SAT-Clique.



Fonte: Modificado de Szwarcfiter (2018).

2.2.2.4 Problema Conjunto Independente

Ainda dentro dos problemas NP-completos, encontra-se o problema do conjunto independente. A prova de que esse problema é NP-completo pode ser feita a partir do Problema do Clique, citado na Subseção 2.2.2.3, na página 27, utilizando uma redução baseada no grafo complementar $\bar{G}$ (SZWARCFITER, 2018).

Seja $G(V,E)$ uma instância do Problema do Clique, em que t representa o tamanho da clique. Constrói-se o grafo complementar $\bar{G}(V, \bar{E})$, no qual cada aresta presente em G é removida e cada par de vértices que não possuía aresta passa a estar conectado. Dessa forma, existe uma clique de tamanho t em G se, e somente se, existe um conjunto independente de tamanho t em $\bar{G}$. Isso ocorre porque um conjunto de t vértices totalmente conectados em G torna-se um conjunto de t vértices sem conexões entre si no grafo complementar, satisfazendo exatamente a definição de conjunto independente (SZWARCFITER, 2018).

Considere a Figura 14. A Figura 14a apresenta um grafo G com uma clique de tamanho $t = 3$. Ao construir o grafo complementar $\bar{G}$, apresentado na Figura 14b, com base no grafo da Figura 14a, os mesmos três vértices deixam de estar conectados entre si, formando um conjunto independente de tamanho 3. Como essa construção é realizada em tempo polinomial, a redução é válida. Dessa forma, como o problema Clique é NP-Completo, conclui-se que o problema Conjunto Independente também é NP-Completo (SZWARCFITER, 2018).

Figura 14 – Redução de clique para conjunto independente.

(a) Clique de tamanho 3. (b) Conjunto independente de tamanho 3.

Fonte: Autoria própria.

2.3 Problema de Partição em Grafo

Nesta seção, é abordado o problema de partição em grafos, definido a partir de um grafo $G(V, E)$, no qual busca dividir G em componentes menores que satisfaçam determinadas propriedades. Em especial, será tratado o problema de partição (k, l).

2.3.1 Grafos (k,l)

Um problema clássico na teoria dos grafos é o problema de partição (k, l). Um grafo G é dito possuir uma partição (k, l) quando seu conjunto de vértices pode ser dividido em k conjuntos independentes e l cliques (COUTO, 2016).

Na partição (k, l), cada vértice do grafo pertence exclusivamente a um único conjunto: ou integra uma clique ou faz parte de um conjunto independente, nunca ambos ao mesmo tempo. O objetivo é dividir todos os vértices do grafo em blocos bem definidos e sem sobreposição, buscando a menor combinação possível de k e l que represente a estrutura do grafo (COUTO, 2016).

Um bom exemplo dessa ideia aparece na Figura 7, na página 9. Nesse grafo, existem várias cliques de tamanhos diferentes, permitindo diferentes formas de particionamento. Uma possibilidade é selecionar duas cliques, uma de tamanho 3 e outra de tamanho 5, resultando em uma partição $(0, 2)$.Nesse caso, todos os vértices são alocados nessas duas cliques, sem conjuntos independentes. Outra forma possível de particionar o mesmo grafo seria escolher os vértices (1,4) para formar um conjunto independente, agrupar (2,3) em uma clique de tamanho 2 e (5,6,7,8) em uma clique de tamanho 4. Esse tipo de análise é mais complexo, pois exige encontrar combinações que cubram todos os vértices sem sobreposição e, ao mesmo tempo, minimizem a quantidade total de partições.

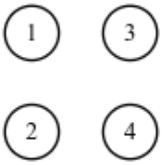
Um exemplo clássico de grafo que se enquadra no modelo de partição (k, l) é o grafo split, ilustrado na Figura 9, na página 10. Esse grafo representa uma partição $(1, 1)$, pois seus vértices podem ser divididos em uma única clique e um único conjunto independente, ou seja, $k = 1$ e $l = 1$. Essa configuração torna o grafo split um caso particular e muito utilizado para ilustrar o conceito de partição (k, l), por apresentar simultaneamente uma estrutura totalmente conectada e outra completamente desconectada. (FREIRE, 2024).

Outra configuração possível dentro do modelo de partição (k, l) é o grafo bipartido, ilustrado na Figura 10, na página 10. Esse tipo de grafo representa uma partição $(2, 0)$, pois seu conjunto de vértices é dividido em dois conjuntos independentes e não há presença de cliques, ou seja, $k = 2$ e $l = 0$ (FREIRE, 2024).

Conforme apresentado por Brandstädt (1996), os grafos split e bipartidos já são conhecidos por permitirem reconhecimento em tempo polinomial.

Outro exemplo de partição é a $(1, 0)$, ilustrado na Figura 15, em que o grafo apresenta apenas um conjunto independente e nenhuma clique. Nesse caso, todos os vértices estão desconectados entre si, sem a presença de arestas internas. Essa estrutura representa um grafo totalmente disperso, no qual $k = 1$ indica a existência de um único conjunto independente e $l = 0$ confirma a ausência de cliques.

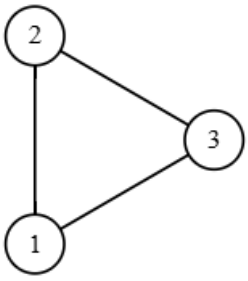
Figura 15 – Grafo partição $(1,0)$.



Fonte: Modificado de Alves (2019).

Também é possível observar outra configuração do modelo de partição (k, l) na Figura 16. Nesse tipo de grafo, todos os vértices estão conectados entre si, formando uma única clique, sem conjuntos independentes. Essa configuração na relação de partição (k, l) representa um caso em que $k = 0$ indica ausência de vértices isolados e $l = 1$ representa a existência de apenas uma clique que reúne todos os vértices.

Figura 16 – Grafo partição $(0,1)$.

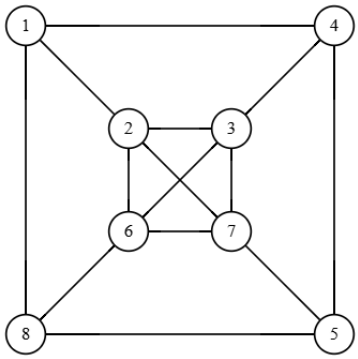


Fonte: Modificado de Szwarcfiter (2018).

Na Figura 17, observa-se um exemplo de partição $(2, 1)$. Nesse caso, o conjunto de vértices está dividido em dois conjuntos independentes e uma clique. Os vértices 1 e 5 formam o primeiro conjunto independente, enquanto os vértices 4 e 8 compõem o segundo conjunto independente. Já os vértices 2, 3, 6 e 7 estão agrupados em uma clique de tamanho 4 no qual todos estão conectados entre si. Essa configuração mostra de forma clara como a partição (k, l)

organiza a estrutura do grafo combinando regiões desconectadas internamente com uma região totalmente conectada.

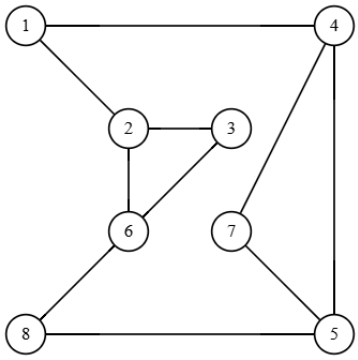
Figura 17 – Grafo partição $(2,1)$.



Fonte: Autoria própria.

Já na Figura 18, tem-se um exemplo de partição $(1, 2)$. Nesse caso, o grafo apresenta um conjunto independente e duas cliques. Os vértices 1 e 8 formam o conjunto independente, sem arestas entre si. A primeira clique é composta pelos vértices 2, 3 e 6, resultando em uma clique de tamanho 3, enquanto a segunda clique reúne os vértices 4, 5 e 7, também de tamanho 3. Essa configuração evidencia como uma única região independente pode coexistir com duas regiões densamente conectadas, caracterizando a partição $(1, 2)$.

Figura 18 – Grafo partição $(1,2)$.

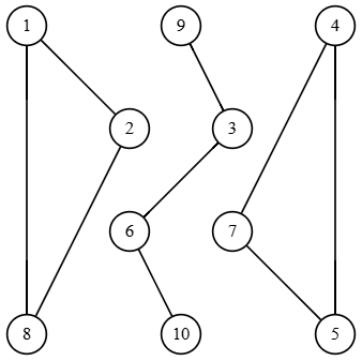


Fonte: Autoria própria.

Observa-se na Figura 19 um exemplo de partição $(2, 2)$. Nesse caso, o grafo apresenta duas cliques e dois conjuntos independentes. A primeira clique é formada pelos vértices 1, 2 e 8, com tamanho 3. A segunda clique reúne os vértices 4, 5 e 7, também com tamanho 3. Além disso, há dois conjuntos independentes: o primeiro formado pelos vértices 9 e 6, e o segundo pelos vértices 3 e 10.

Brandstädt (1996) estudou que o reconhecimento de grafos que admitem partições $(1,2)$, $(2,1)$ e $(2,2)$ pode ser realizado em tempo polinomial. Isso significa que há algoritmos determinísticos e eficientes capazes de identificar se um grafo pertence a essas classes de partição.

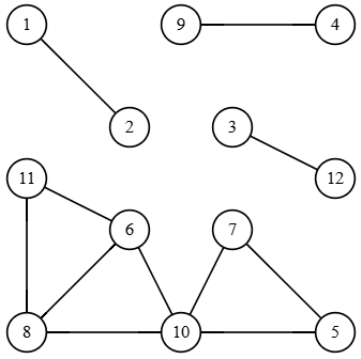
Figura 19 – Grafo partição $(2,2)$.



Fonte: Autoria própria.

Por último, a Figura 20 apresenta uma partição $(3, 3)$, que representa uma estrutura mais complexa e mais difícil de ser analisada. Nesse tipo de grafo, também seria possível encontrar outras combinações de partição com valores menores de (k, l), porém, a divisão em três cliques e três conjuntos independentes foi escolhida para demonstrar a viabilidade dessa configuração. As três cliques são formadas da seguinte forma: os vértices 1 e 2 compõem a primeira clique de tamanho 2, os vértices 3 e 12 formam a segunda clique de tamanho 2 e os vértices 9 e 4 constituem a terceira clique, também de tamanho 2. Já os conjuntos independentes são compostos pelos vértices 11 e 10 no primeiro conjunto, 6 e 7 no segundo e 8 e 5 no terceiro. Essa configuração evidencia a flexibilidade da partição (k, l), mostrando que um mesmo grafo pode ser dividido de diferentes formas, e que a representação adotada comprova a possibilidade de obter exatamente três cliques e três conjuntos independentes.

Figura 20 – Grafo partição $(3,3)$.



Fonte: Autoria própria.

A complexidade da partição (k, l) também foi apresentada por Brandstädt (1996), que mostrou que, para valores de $k \geq 3$, o problema de reconhecimento já é considerado NP-Completo. Para os demais valores de k e l, o problema é polinomial. Isso inclui os casos simples, como $(0,1)$, apresentado na Figura 16, e $(1,0)$, ilustrado na Figura 15, além dos exemplos clássicos: grafo split $(1,1)$ e grafo bipartido $(2,0)$, assim como $(0,2)$, $(1,2)$, $(2,1)$ e $(2,2)$. No caso da Figura 20, que representa uma partição $(3,3)$, observa-se uma estrutura mais complexa

e difícil de ser analisada, confirmando a elevação da dificuldade computacional. Dessa forma, conforme estabelecido, quando $k \geq 3$, o problema é NP-Completo.

3 O PROBLEMA SANDUÍCHE

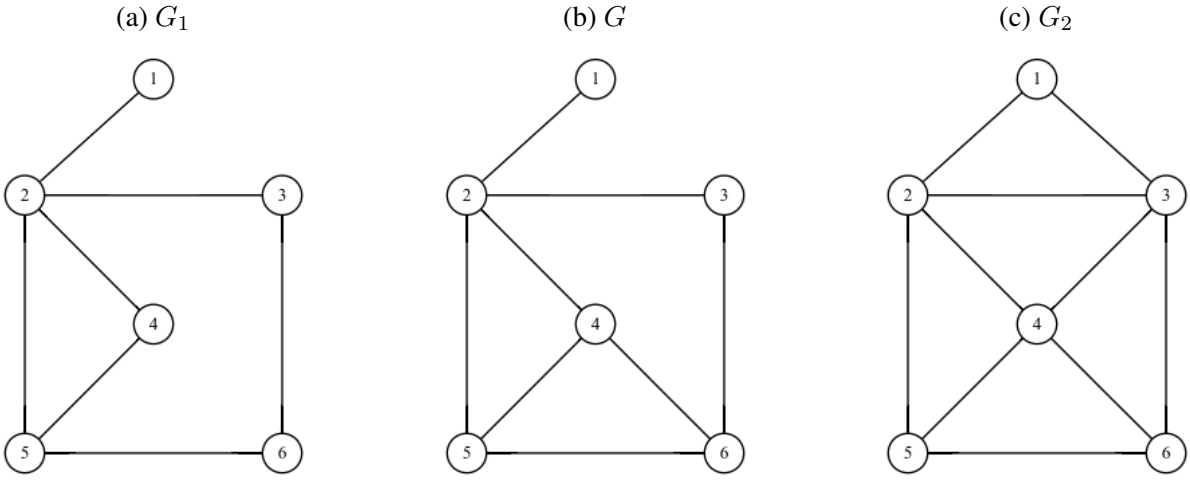
Neste capítulo, são apresentadas as definições preliminares do Problema Sanduíche, explicando sua formulação e principais características. Também são descritas cinco variações do problema relacionadas à sua complexidade computacional, considerando partições $((k,l)$ nos casos split $(1,1)$, bipartido $(2,0)$, $(2,1)$, $(2,2)$ e (k,l) Δ-Limitado. Duas dessas variações podem ser resolvidas em tempo polinomial, enquanto as outras três são classificadas como NP-completo.

3.1 Definições Preliminares do Problema

Um grafo $G(V,E)$ é dito ser grafo-sanduíche dos grafos $G_1(V,E_1)$ e $G_2(V,E_2)$ se, e somente se, $E_1 \subseteq E \subseteq E_2$. Em outras palavras, trata-se de um grafo que possui o mesmo conjunto de vértices V dos grafos G_1 e G_2. Além disso, tanto G quanto G_1 podem ser vistos como subgrafos geradores de G_2, pois compartilham o mesmo conjunto de vértices e possuem conjuntos de arestas contidos em E_2. O conjunto de arestas E inclui obrigatoriamente todas as arestas de G_1 e pode incluir algumas ou todas as arestas presentes em G_2 representadas por $E_2 \setminus E_1$ (COUTO, 2016).

Na Figura 21 são apresentados três grafos: G_1, G e G_2. O grafo G_1 representa as arestas obrigatórias que devem estar presentes em qualquer solução do problema. O grafo G_2 corresponde ao conjunto total de arestas possíveis que podem ser utilizadas. O grafo G é um exemplo de solução do problema sanduíche, pois contém todas as arestas obrigatórias de G_1 adicionais presentes em G_2. Neste caso, G inclui, além das arestas de G_1, apenas a aresta $(4,6)$ de G_2.

Figura 21 – Problema do Grafo Sanduíche.

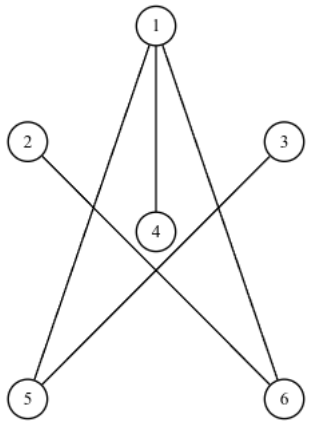


Fonte: Modificado de Cerioli et al. (1998).

Para simplificar a notação, adota-se a seguinte definição: E_3 representa o conjunto de arestas que não podem ser incluídas em nenhuma solução válida para o problema sanduíche. Dessa forma, um grafo sanduíche G para o par G_1 e G_2 deve satisfazer as condições $E_1 \subseteq E$ e $E \cap E_3 = \emptyset$. Nesse contexto, E_1 corresponde ao conjunto de arestas obrigatórias, $E_2 \setminus E_1$ representa as arestas opcionais e E_3 indica o conjunto de arestas proibidas (DANTAS; FIGUEIREDO; FARIA, 2002).

Com o objetivo de complementar a explicação da instância ilustrada na Figura 21, na página 34, foi incluída a Figura 22, que apresenta o grafo G_3, destacando as arestas proibidas. Ela tem a finalidade de tornar mais clara a compreensão das restrições impostas na construção de soluções válidas para o problema sanduíche. O grafo G_3 reúne todas as arestas que não pertencem ao conjunto G_2, isto é, aquelas que não podem ser incluídas em G.

Figura 22 – Grafo de arestas proibidas.



Fonte: Autoria própria.

Esse conceito faz com que o grafo sanduíche seja uma versão intermediária entre G_1 e G_2, garantindo que ele contenha todas as arestas de G_1 e, opcionalmente, algumas arestas de G_2. Além disso, nenhuma aresta pertencente a G_3, que representa as conexões do grafo completo que não estão em G_2, pode ser incluída (DANTAS; FIGUEIREDO; FARIA, 2002).

O Problema do Grafo Sanduíche consiste em determinar se existe um grafo G, com conjunto de vértices V e conjunto de arestas E, que esteja entre G_1 e G_2, e que respeite a propriedade especificada. Nesse contexto, essa propriedade é denominada genericamente de Propriedade II.

Conforme definido por Golumbic, Kaplan e Shamir (1995), o PROBLEMA GRAFO SANDUÍCHE PARA A PROPRIEDADE II é formalmente apresentado da seguinte maneira:

> PROBLEMA GRAFO SANDUÍCHE PARA A PROPRIEDADE II
>
> Instância: Conjunto de vértices V, conjunto de arestas obrigatórias E_1, conjunto de arestas proibidas E_3.
>
> Questão: Existe um grafo $G(V, E)$ tal que $E_1 \subseteq E$ e $E \cap E_3 = \emptyset$ que satisfaz a propriedade II?

Para responder a essa questão, as seções seguintes apresentarão diferentes propriedades que podem ser exploradas no Problema Grafo Sanduíche. Entre essas propriedades, destacam-se aquelas baseadas na partição (k, l). A utilização dessa partição não é arbitrária, pois ela define estruturalmente a propriedade a ser verificada no grafo solução. Como o Problema do Grafo Sanduíche consiste em determinar se existe um grafo G, com $E_1 \subseteq E \subseteq E_2$, que satisfaça uma determinada propriedade, a adoção de (k,ℓ) permite expressar essa propriedade em termos de partições estruturais, ou seja, dividindo os vértices em k conjuntos independentes e l cliques.

Além disso, há uma relação direta entre as partições (k,l) e (l,k), pois trocar a ordem desses parâmetros apenas inverte os papéis de cliques e conjuntos independentes, preservando a estrutura central do problema e, consequentemente, sua complexidade.

Essa abordagem também possibilita uma classificação mais clara da complexidade computacional, uma vez que determinados valores de k e l levam a problemas polinomiais, enquanto outros resultam em problemas NP-Completo, estabelecendo a fronteira entre casos tratáveis e intratáveis no Problema do Grafo Sanduíche.

Na Seção 3.2 é apresentado o caso $(1,1)$, que corresponde ao grafo split. Nessa seção são discutidas as principais características desse tipo de grafo, bem como sua relação com o Problema Grafo Sanduíche e a forma como a partição $(1,1)$ pode ser utilizada para verificar a existência de uma solução.

Na Seção 3.3 é abordado o caso $(2,0)$, correspondente aos grafos bipartidos. Essa seção apresenta as propriedades estruturais dos grafos bipartidos e destaca os resultados conhecidos sobre a complexidade do problema nesse tipo de partição.

As Seções 3.4 e 3.5 tratam, respectivamente, dos casos $(2,1)$ e $(2,2)$, que representam outras combinações entre conjuntos independentes e cliques. Nessas seções são discutidas as particularidades dessas partições e como elas impactam na análise do Problema Grafo Sanduíche.

Por fim, na Seção 3.6 é analisada a propriedade $\Delta(k, l)$-limitada, a qual impõe restrições adicionais ao grau dos vértices. Essa seção complementa a discussão, mostrando como limitações no grau podem influenciar diretamente na tratabilidade do problema.

3.2 Problema do Grafo Sanduíche em Grafos Split $(1,1)$

Nesta seção é analisada a complexidade do Problema Grafo Sanduíche para grafos split, tomando como base os resultados clássicos apresentados por Golumbic, Kaplan e Shamir (1995). Toda a análise desenvolvida aqui segue diretamente esse trabalho fundamental; por isso, as definições, propriedades e caracterizações discutidas a seguir são extraídas e adaptadas de sua formulação original.

Um grafo split é definido por uma partição do conjunto de vértices $V = K \cup L$, em que K forma um conjunto independente e L forma uma clique, não existindo vértices fora dessa divisão. No contexto do Problema Grafo Sanduíche, essa partição deve ser compatível com os grafos limites: o conjunto K precisa ser independente já em G_1, enquanto o conjunto L deve constituir uma clique já em G_2. Essas restrições garantem que o grafo solução G, com

$E_1 \subseteq E \subseteq E_2$ possa respeitar a estrutura imposta pela classe dos grafos split.

Dada uma entrada (V, E_1, E_3) do Problema Grafo Sanduíche Split, o objetivo é determinar, para cada vértice, se ele deve pertencer ao conjunto independente K ou à clique L, de modo que a estrutura resultante satisfaça as restrições impostas pelos conjuntos de arestas obrigatórias e proibidas.

> PROBLEMA GRAFO SANDUÍCHE PARA GRAFO SPLIT $(1,1)$
>
> Instância: Conjunto de vértices V, conjunto de arestas obrigatórias E_1, conjunto de arestas proibidas E_3.
>
> Questão: Existe um grafo $G(V, E)$ tal que $E_1 \subseteq E$ e $E \cap E_3 = \emptyset$, e G é um grafo split $(1,1)$?

A resolução pode ser alcançada por meio da formulação de um sistema de restrições booleanas que modela as condições impostas pelas arestas obrigatória E_1 e pelas arestas proibidas E_3. Esse sistema corresponde a uma instância do problema de 2-SAT, apresentado na Seção 2.2.1.3, na página 15, que pode ser solucionado em tempo polinomial. Dessa forma, o Problema Grafo Sanduíche para grafos split pode ser resolvido de maneira eficiente por meio da construção e solução dessa instância de 2-SAT.

As restrições associadas ao Problema Grafo Sanduíche Split decorrem diretamente da definição de grafo split e das propriedades dos conjuntos K e L. Nesta etapa do trabalho, será utilizada a notação $[x,y]$ para representar uma aresta entre os vértices x e y (GOLUMBIC; KAPLAN; SHAMIR, 1995).

Para as arestas obrigatórias (E_1) se uma aresta $[x,y]$ pertence a E_1, significa que ela deve necessariamente estar presente no grafo solução. Como o conjunto K é um conjunto independente, não é permitido que ambos os vértices x e y pertençam simultaneamente a K, pois isso violaria essa propriedade estrutural. Em termos lógicos, essa restrição é representada pela cláusula $(X \vee Y)$, uma vez que, se ambos os valores fossem falsos (isto é, se x e y estivessem em K), a cláusula não seria satisfeita. Assim, pelo menos um dos vértices deve necessariamente pertencer ao conjunto L.

Para as arestas proibidas (E_3), se uma aresta $[x,y]$ pertence a E_3, significa que ela não pode aparecer no grafo solução. Como, em um grafo split o conjunto L forma uma clique, todos os seus vértices devem estar conectados entre si. Portanto, não é permitido que ambos os vértices x e y sejam alocados em L, pois isso implicaria na presença da aresta proibida, violando a restrição imposta. Essa condição é modelada pela cláusula $(\overline{X} \vee \overline{Y})$, uma vez que, se ambos fossem verdadeiros (isto é, se x e y estivessem em L), a cláusula não seria satisfeita. Assim, pelo menos um dos vértices deve pertencer ao conjunto K.

Para formalizar, associa-se a cada vértice $x \in V$ uma variável booleana X, definida da

seguinte forma:

$$X = \begin{cases} \text{Verdadeiro}, & \text{se } x \in L \text{ (clique)} \\ \text{Falso}, & \text{se } x \in K \text{ (conjunto independente)} \end{cases}$$

Essa representação possibilita a aplicação de algoritmos de 2-SAT para verificar a existência de uma atribuição de valores que satisfaça todas as cláusulas geradas. Em outras palavras, permite determinar de forma eficiente se existe uma partição dos vértices entre K e L que satisfaça as restrições impostas pelo problema.

Considere que exista um grafo sanduíche split válido. Nesse caso, o conjunto de vértices V pode ser particionado em dois subconjuntos K e L, onde K é um conjunto independente e L forma uma clique. A partir dessa partição, define-se uma atribuição de verdade t para cada variável booleana X associada a um vértice x, de modo que $t(X) =$ VERDADEIRO se, e somente se, $x \in L$.

Essa atribuição garante que todas as cláusulas do sistema sejam satisfeitas, pois, para cada aresta obrigatória $[x,y] \in E_1$, pelo menos um dos vértices pertence a L, o que assegura a veracidade da cláusula $(X \vee Y)$. Além disso, para cada aresta proibida $[x,y] \in E_3$, pelo menos um dos vértices pertence a K, tornando verdadeira a cláusula $(\overline{X} \vee \overline{Y})$. Dessa forma, a existência de um sanduíche split implica que o sistema de equações booleanas é consistente.

Por outro lado, suponha que o sistema de equações booleanas seja consistente, isto é, que exista uma atribuição de verdade t que satisfaça todas as cláusulas. A partir dessa atribuição, define-se a partição dos vértices de modo que um vértice x pertence ao conjunto L se, e somente se, $t(X) =$ VERDADEIRO, e ao conjunto K caso contrário.

Com essa definição, constrói-se o grafo sanduíche $G(V, E)$, incluindo todas as arestas que formam a clique em L, nenhuma aresta entre vértices de K, todas as arestas obrigatórias de E_1 e, opcionalmente, qualquer subconjunto de arestas de E_2 que não viole as restrições impostas.

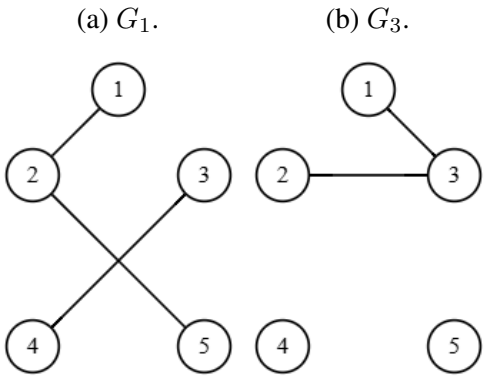
É necessário verificar que essa construção não viola as restrições originais do problema. Se houvesse uma aresta proibida $[x,y] \in E_3$ com ambos os extremos em L, teríamos $t(X) = t(Y) =$ VERDADEIRO, o que tornaria a cláusula $(\overline{X} \vee \overline{Y})$ falsa, entrando em contradição com a hipótese de consistência. De forma análoga, se houvesse uma aresta obrigatória $[x,y] \in E_1$ com ambos os vértices em K, teríamos $t(X) = t(Y) =$ FALSO, o que violaria a cláusula $(X \vee Y)$.

Dessa forma, para que a construção seja válida, todas as condições devem ser satisfeitas, garantindo que a partição obtida define um grafo sanduíche split válido.

Com isso, considere um exemplo do Problema Grafo Sanduíche para o caso split. Na Figura 23, tem-se o grafo G_1, mostrado na Figura 23a, que contém as arestas obrigatórias, e o grafo G_3, apresentando na Figura 23b, que contém as arestas proibidas. A partir dessa entrada, deseja-se determinar se existe um grafo sanduíche que seja um grafo split $(1,1)$.

Para cada uma dessas arestas, representamos a condição correspondente por meio de cláusulas com dois literais. Assim, para a aresta obrigatória $\{1,2\}$, obtemos a cláusula $(1 \vee 2)$; para a aresta $\{2,5\}$, a cláusula $(2 \vee 5)$; e, para a aresta obrigatória $\{3,4\}$, a cláusula $(3 \vee 4)$.

Figura 23 – Exemplo do caso Problema Split.



Fonte: Autoria própria.

Para as arestas proibidas, procedemos de forma análoga, mas representando a condição de não-adjacência. Assim, para a aresta proibida $\{1,3\}$, obtemos a cláusula $(\overline{1} \vee \overline{3})$, e para a aresta proibida $\{2,3\}$, a cláusula $(\overline{2} \vee \overline{3})$. Assim, a Fórmula C correspondente a este caso é dada por: $C = (1 \vee 2) \wedge (2 \vee 5) \wedge (3 \vee 4) \wedge (\overline{1} \vee \overline{3}) \wedge (\overline{2} \vee \overline{3})$.

Para cada cláusula, a Tabela 5 apresenta as implicações correspondentes para este caso. A partir dessas implicações, constrói-se o grafo de implicações, e, em seguida, realiza-se a ordenação topológica para verificar a consistência das atribuições.

Tabela 5 – Implicações do Problema Split.

Cláusula	Implicação 1	Implicação 2
$(1 \vee 2)$	$\overline{1} \to 2$	$\overline{2} \to 1$
$(2 \vee 5)$	$\overline{2} \to 5$	$\overline{5} \to 2$
$(3 \vee 4)$	$\overline{3} \to 4$	$\overline{4} \to 3$
$(\overline{1} \vee \overline{3})$	$1 \to \overline{3}$	$3 \to \overline{1}$
$(\overline{2} \vee \overline{3})$	$2 \to \overline{3}$	$3 \to \overline{2}$

Fonte: Autoria própria.

Após a ordenação topológica, é possível determinar quais vértices recebem os valores F ou V. A Tabela 6 apresenta o resultado final desse caso, indicando a atribuição correspondente para cada vértice.

Tabela 6 – Valor lógico Problema Split.

V	Valor lógico
1	F
2	V
3	F
4	V
5	V

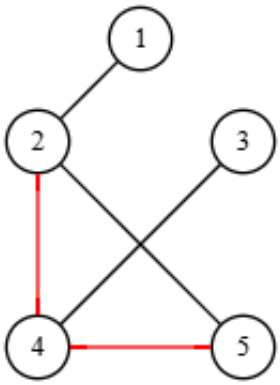
Fonte: Autoria própria.

Já conhecendo os valores lógicos, substituímos diretamente na Fórmula C, assim obtemos: $C = (F \vee V) \wedge (V \vee V) \wedge (F \vee V) \wedge (\overline{F} \vee \overline{F}) \wedge (\overline{V} \vee \overline{F})$. Sabendo que $\overline{F} = V$ e $\overline{V} = F$, a expressão torna-se: $C = V \wedge V \wedge V \wedge V \wedge V = V$. Como a Fórmula C é satisfeita, conclui-se que, nesse exemplo, existe um grafo sanduíche que é um grafo split.

Quem recebe valor V é atribuído ao grupo da clique L, enquanto os vértices com valor F são destinados ao conjunto independente K. Nesse exemplo, como os vértices 1 e 3 possuem valor F, eles formam um conjunto independente K, e os vértices 2, 4 e 5, com valor V, compõem a clique L.

Para construir o grafo sanduíche split, partimos do grafo contendo apenas as arestas obrigatórias e, em seguida, acrescentamos, se necessário, as arestas faltantes para que os vértices de L formem uma clique. Nesse caso, foi preciso adicionar as arestas $\{2,4\}$ e $\{4,5\}$. A Figura 24 destaca em vermelho as arestas acrescentadas e permite observar que nenhuma aresta proibida foi incluída. Dessa forma, obtem-se um grafo sanduíche que é split para as entradas consideradas.

Figura 24 – Grafo Sanduíche Split.



Fonte: Autoria própria.

Conclui-se que o Problema Grafo Sanduíche Split pode ser resolvido de maneira eficiente. A transformação inicial das restrições do grafo em um sistema de equações booleanas é realizada em tempo linear em relação ao tamanho da entrada, isto é, em $O(|E_1| + |E_3|)$. Uma vez formulado o sistema, sua resolução corresponde à verificação da satisfatibilidade de uma fórmula 2-SAT. Como o problema 2-SAT admite algoritmos polinomiais baseados na construção do grafo de implicações e na identificação das componentes fortemente conexas, a complexidade total permanece linear em relação ao número de vértices e ao número de restrições, resultando em uma solução com complexidade $O(|V| + |E_1| + |E_3|)$.

3.3 Problema do Grafo Sanduíche em Grafos Bipartido $(2,0)$

Nesta seção é revisada a complexidade do Problema Grafo Sanduíche aplicado a grafos bipartidos, conforme descrito por Souza (2002). Um grafo bipartido é definido como aquele cujo conjunto de vértices pode ser particionado em dois subconjuntos disjuntos, denominados

partições independentes, de modo que não existam arestas conectando vértices pertencentes a um mesmo subconjunto.

O problema de reconhecimento de grafos bipartidos consiste em determinar se alguma partição é possível. Na Seção 2.2.1.5, na página 23, é apresentada a complexidade associada a esse tipo de grafo, além de ser descrito como é possível verificar, de forma prática, se um grafo é bipartido. O algoritmo percorre os vértices do grafo a partir de uma busca em largura BFS, que também foi apresentada na Seção 2.2.1.1, na página 11. Durante a execução, para cada aresta $[u,v]$, verifica-se se os vértices u e v pertencem a níveis diferentes da busca. Se, em alguma aresta, $d[u] = d[v]$, significa que u e v estão no mesmo nível, o que caracteriza a presença de um ciclo ímpar, impossibilitando a bipartição e indicando que o grafo não é bipartido. Caso contrário, se nenhuma aresta violar essa condição, o grafo é considerado bipartido (FEOFILOFF; KOHAYAKAWA; WAKABAYASHI, 2004).

Por se tratar de um problema solucionável em tempo polinomial, o Problema Grafo Sanduíche considerando a propriedade bipartida, também denominada $(2,0)$, tem como objetivo verificar, dada uma instância formada por um conjunto de vértices, arestas obrigatórias e arestas proibidas, se existe algum grafo sanduíche que satisfaça essa propriedade (SOUZA, 2002).

> PROBLEMA GRAFO SANDUÍCHE PARA GRAFO BIPARTIDO $(2,0)$
>
> Instância: Conjunto de vértices V, conjunto de arestas obrigatórias E_1, conjunto de arestas proibidas E_3.
>
> Questão: Existe um grafo $G(V, E)$ tal que $E_1 \subseteq E$ e $E \cap E_3 = \emptyset$, e G é um grafo bipartido $(2,0)$?

Conforme apresentado por Souza (2002), quando a Propriedade II é hereditária, existe um grafo sanduíche para a instância (V, E_1, E_2) que satisfaz essa propriedade se, e somente se, o grafo $G_1(V, E_1)$, formado apenas pelas arestas obrigatórias, já possui tal característica. Em outras palavras, no caso de propriedades hereditárias, não é necessário explorar todas as possíveis adições de arestas permitidas: a verificação se restringe a avaliar o grafo obrigatório G_1.

A propriedade de ser bipartido é hereditária. Assim, no Problema Grafo Sanduíche $(2,0)$, basta verificar se G_1 já é bipartido. Caso essa condição seja satisfeita, existe pelo menos um grafo sanduíche que mantém a propriedade bipartida, desde que as arestas proibidas não sejam incluídas. Em outras palavras, não é necessário adicionar novas arestas para garantir a bipartição: o próprio grafo obrigatório assegura essa característica. Essa propriedade simplifica significativamente a resolução do problema (SOUZA, 2002).

A verificação da bipartição no grafo G_1 pode ser realizada utilizando o Algoritmo 8, na página 24, que possui complexidade linear, isto é, $O(n + m)$. Dessa forma, o Problema Grafo Sanduíche para grafos bipartidos $(2,0)$ é solucionado em tempo polinomial, garantindo eficiência computacional mesmo em instâncias de grande porte (SOUZA, 2002).

3.4 Problema do Grafo Sanduíche em Grafos com Partição $(2,1)$

Nesta seção é analisada a complexidade computacional do Problema Grafo Sanduíche quando aplicado à classe de grafos com partição $(2,1)$, conforme apresentado por Souza (2002). Toda a análise desenvolvida aqui segue diretamente esse trabalho fundamental; por esse motivo, as definições, propriedades e caracterizações discutidas a seguir são extraídas e adaptadas de sua formulação original. Um grafo com partição $(2,1)$ é definido como aquele cujos vértices podem ser divididos em dois conjuntos independentes e um conjunto que forma uma clique.

O Problema do Grafo Sanduíche com partição $(2,1)$ é NP-completo, por meio de uma redução a partir do Problema da 3-SAT, apresentado na Seção 2.2.2.2, na página 26, reconhecidamente NP-completo. Para isso, a autora definiu formalmente cada um dos problemas da seguinte maneira:

> 3-SATISFATIBILIDADE $(3-SAT)$
>
> Instância: Conjunto $X = \{x_1, \ldots, x_n\}$ de variáveis, coleção $C = \{c_1, \ldots, c_m\}$ de cláusulas sobre X tal que cada cláusula $c \in C$ tem $|c| = 3$ literais.
>
> Questão: Existe uma atribuição verdade para X tal que cada cláusula em C tem ao menos um literal verdadeiro?

> PROBLEMA GRAFO SANDUÍCHE $(2,1)$
>
> Instância: Conjunto de vértices V, conjunto de arestas obrigatórias E_1, conjunto de arestas proibidas E_3.
>
> Questão: Existe um grafo $G(V, E)$, tal que $E_1 \subseteq E$ e $E \cap E_3 = \emptyset$, e G é (2, 1)?

A partir dessas definições, considera-se a construção que transforma uma instância arbitrária de 3-SAT em uma instância equivalente do Problema Grafo Sanduíche $(2,1)$, demonstrando, assim, sua NP-completude. Para isso, a autora construiu uma instância específica (V, E_1, E_3) do Problema Grafo Sanduíche $(2,1)$ a partir de uma instância genérica (X, C) de 3-SAT, de modo que C é satisfatível se, e somente se, (V, E_1, E_3) admite um grafo sanduíche $G(V, E)$ pertencente à classe $(2,1)$.

Nessa construção, são definidos o conjunto de vértices, as arestas obrigatórias e as arestas proibidas, de modo que cada componente represente adequadamente as variáveis, os literais e as cláusulas da instância de 3-SAT. Essa correspondência garante que a fórmula seja satisfatível se, e somente se, o grafo resultante admitir uma partição $(2,1)$.

O conjunto de vértices V é formado por três grupos distintos. O primeiro grupo consiste em um conjunto auxiliar de vértices: $\{l_1, l_2, s_{11}, s_{12}, s_{21}, s_{22}\}$. Nesta construção os vértices identificados pela letra s correspondem ao conjunto independente. O segundo grupo corresponde às variáveis da instância de 3-SAT: para cada variável x_i, com $1 \leq i \leq n$, são definidos dois vértices que representam os literais x_i e $\overline{x}_i$, além de um vértice adicional p_i. Por fim, o terceiro grupo está relacionado às cláusulas: para cada cláusula $c_j = (\ell_1^j \vee \ell_2^j \vee \ell_3^j)$, com $1 \leq j \leq m$, são

introduzidos três vértices t_1^j, t_2^j, t_3^j, Cada um correspondente a um literal da cláusula, onde cada l_k^j representa um literal da forma x_i ou $\overline{x}_i$.

O conjunto de arestas obrigatórias E_1 é definido em três grupos principais. Nesta construção os vértices s ao conjunto independente, seguindo a convenção adotada nesta seção. O primeiro grupo contempla as arestas entre os vértices auxiliares:

$$\{l_1l_2,\, l_1s_{11},\, l_1s_{12},\, s_{11}s_{12},\, l_2s_{21},\, l_2s_{22},\, s_{21}s_{22}\}.$$

O segundo grupo refere-se às variáveis: para cada variável x_i, considera-se o conjunto

$$\{x_is_{11},\, \overline{x}_is_{12},\, x_ip_i,\, \overline{x}_ip_i\}.$$

O terceiro grupo corresponde às cláusulas: para cada cláusula c_j, define-se o conjunto de arestas

$$\{t_1^jt_2^j,\, t_1^jt_3^j,\, t_2^jt_3^j\}.$$

O conjunto de arestas proibidas E_3 também é dividido em três grupos principais. Nesta construção os vértices s correspondem ao conjunto independente, mantendo a mesma convenção adotada nesta seção. O primeiro grupo abrange as arestas entre os vértices auxiliares:

$$\{l_1s_{21},\, l_1s_{22},\, l_2s_{11},\, l_2s_{12},\, s_{11}s_{21},\, s_{11}s_{22},\, s_{12}s_{21},\, s_{12}s_{22}\}.$$

O segundo grupo refere-se às variáveis: para cada variável x_i, inclui-se o conjunto

$$\{x_i\overline{x}_i,\, p_il_2\}.$$

O terceiro grupo está associado às cláusulas: para cada cláusula $c_j = (\ell_1^j \vee \ell_2^j \vee \ell_3^j)$, considera-se o conjunto

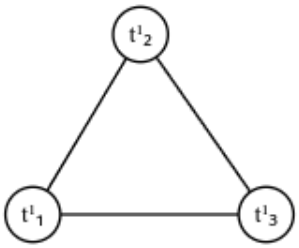
$$\{t_1^j\ell_1^j,\, t_2^j\ell_2^j,\, t_3^j\ell_3^j\}.$$

Conforme ilustrado na Figura 25, apresenta-se o grafo base com as arestas obrigatórias, representadas por linhas sólidas. As arestas proibidas, por sua vez, são indicadas por linhas pontilhadas.

A Figura 26 apresenta a estrutura associada a uma variável, evidenciando as arestas obrigatórias e proibidas correspondentes.

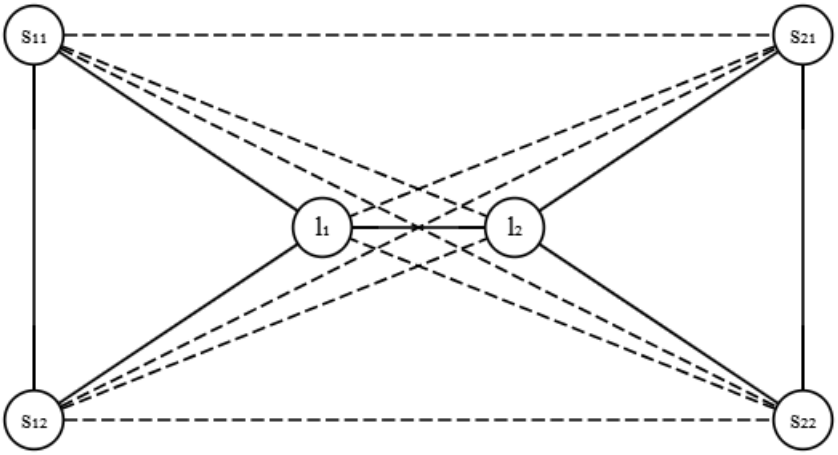
A Figura 27 apresenta a estrutura correspondente a uma cláusula, mostrando como os vértices t_1^l, t_2^l e t_3^l cada um associado a um literal da cláusula, são conectados por arestas obrigatórias que formam um triângulo. Essa construção garante que, em qualquer solução válida, pelo menos um dos literais da cláusula seja satisfeito.

Figura 27 – Cláusula no grafo $(2{,}1)$.



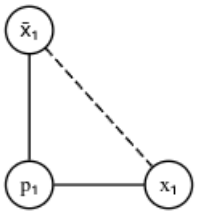
Fonte: Modificado de Souza (2002).

Figura 25 – Grafo base $(2,1)$.



Fonte: Modificado de Souza (2002).

Figura 26 – Variável no grafo $(2,1)$.



Fonte: Modificado de Souza (2002).

Considere a instância $I = (X, C)$, em que $X = \{x_1, x_2, x_3\}$ e

$$C = (\overline{x_1} \vee x_2 \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee x_2 \vee x_3),$$

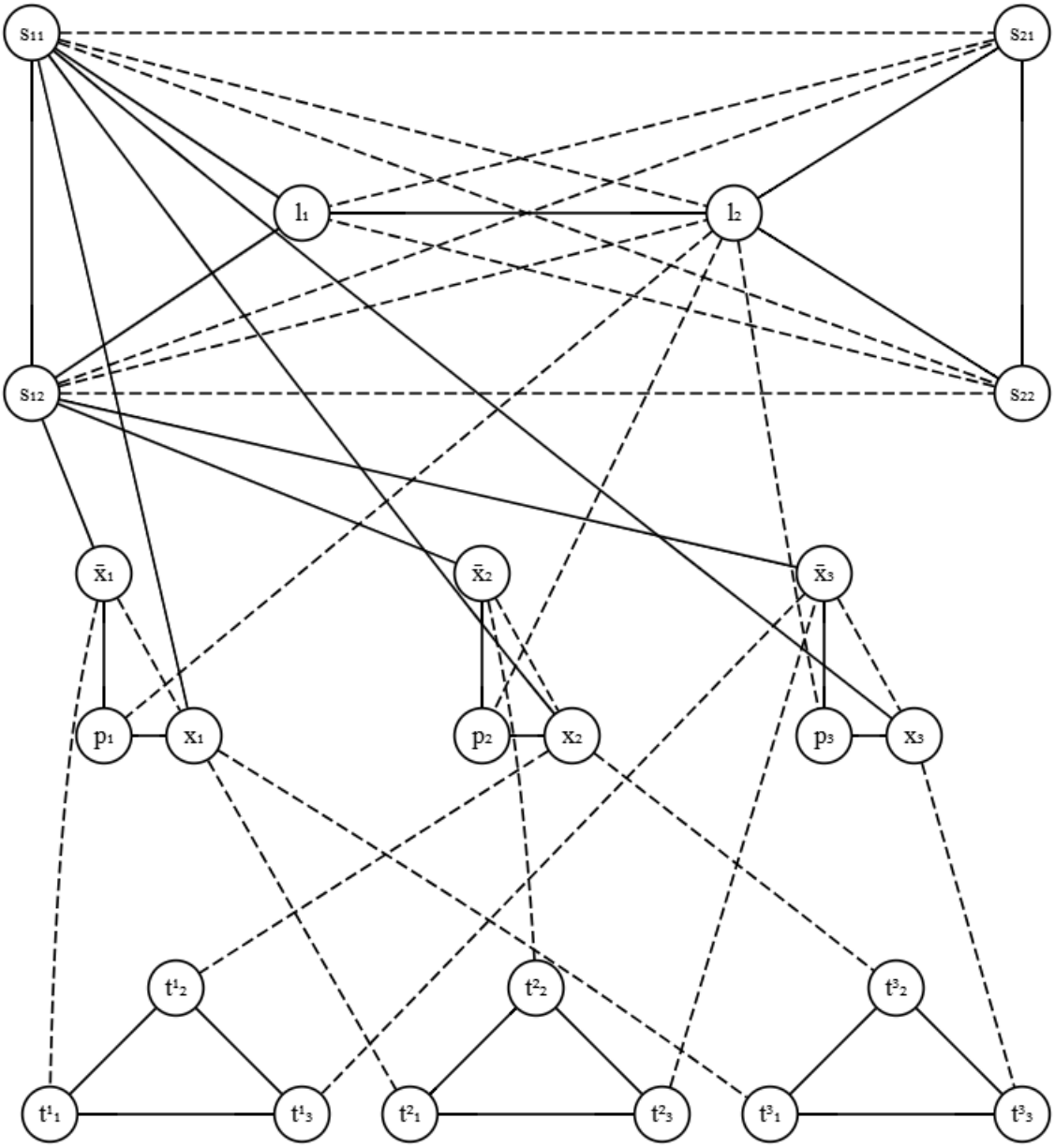
uma instância do problema 3-SAT. A Figura 28 apresenta a construção da instância particular (V, E_1, E_3) do Problema Grafo Sanduíche $(2,1)$ gerada a partir de I.

Considera-se dois lemas fundamentais para a comprovação da NP -completude do Problema Grafo Sanduíche $(2,1)$. Esses lemas estabelecem a equivalência entre a satisfatibilidade da instância original de 3-SAT e a existência de um grafo que contenha todas as arestas obrigatórias, exclua as arestas proibidas e admita uma partição $(2,1)$.

O Lema 1 a seguinte afirmação: se a instância particular (V, E_1, E_3) do Problema Grafo Sanduíche $(2,1)$, construída conforme descrito, admite um grafo $G(V, E)$ tal que $E_1 \subseteq E$ e $E \cap E_3 = \emptyset$ e G possui uma partição $(2,1)$, então existe uma atribuição de valores de verdade que satisfaz a instância (X, C) do Problema da 3-SAT.

A prova é conduzida assumindo a existência de um grafo sanduíche $(2,1)$, denotado por $G(V, E)$, em que (S_1, S_2, L) constitui uma partição $(2,1)$. Nessa partição, os conjuntos S_1 e S_2 são conjuntos independentes, enquanto L forma uma clique. Para a demonstração deste lema, a autora estabeleceu três afirmações que descrevem propriedades fundamentais dessa partição e sua relação com os vértices e arestas definidos na construção da instância.

Figura 28 – Construção da instância $(2,1)$.



Fonte: Modificado de Souza (2002).

Afirmação 1: $l_1, l_2 \in L$ e $s_{11}, s_{12}, s_{21}, s_{22} \in S_1 \cup S_2$.

Essa afirmação estabelece que os vértices auxiliares l_1 e l_2 pertencem necessariamente ao conjunto L, que corresponde à clique da partição. Já os vértices s_{11}, s_{12}, s_{21} e s_{22} devem estar alocados em um dos conjuntos independentes S_1 ou S_2. Essa distinção é fundamental, pois garante que a estrutura do grafo preserve as propriedades impostas pela partição $(2,1)$, evitando que esses vértices formem novas cliques ou gerem adjacências indevidas que possam violar as restrições de independência.

A prova da Afirmação 1 demonstra que, como $S_1 \cup S_2$ induz um subgrafo bipartido em G, qualquer triângulo presente no grafo formado pelas arestas obrigatórias, denominado G_1, deve conter pelo menos um vértice pertencente a L. Isso ocorre porque os conjuntos S_1 e S_2 são

independentes, ou seja, não podem conter pares de vértices conectados por arestas. Assim, se existir um triângulo, isto é, três vértices com todas as três conexões entre eles presentes em E_1, pelo menos um desses vértices precisa estar em L, que é a única parte da partição que admite conexões completas entre seus vértices.

Além disso, cada vértice do conjunto $\{s_{11}, s_{12}, s_{21}, s_{22}\}$ está conectado, por meio de arestas proibidas (E_3), a três vértices que formam um triângulo em G_1. Isso significa que cada um desses vértices possui conexões proibidas com um conjunto de três vértices fortemente conectados entre si pelas arestas obrigatórias. Caso algum vértice s_{ij} fosse colocado no conjunto L, para que o triângulo fosse mantido como clique, os demais vértices também precisariam pertencer a L. Como existem arestas proibidas ligando s_{ij} a esses vértices, essa condição violaria a restrição de que nenhuma aresta de E_3 pode estar presente em L. Assim, os vértices s_{11}, s_{12}, s_{21} e s_{22} não podem pertencer ao conjunto L. Consequentemente, os vértices auxiliares restantes l_1 e l_2 devem necessariamente pertencer a L.

Os pares de vértices $\{s_{11}, s_{12}\}$ e $\{s_{21}, s_{22}\}$ induzem arestas no grafo G_1, o que implica que, para cada $i = 1, 2$, pelo menos um vértice do par $\{s_{i1}, s_{i2}\}$ deve pertencer a cada um dos conjuntos independentes S_1 e S_2. Dessa forma, pode-se assumir, sem perda de generalidade, que s_{11} e s_{21} pertencem ao conjunto S_1, enquanto s_{12} e s_{22} pertencem ao conjunto S_2.

Observe que, para a instância (V, E_1, E_3) que admite um grafo sanduíche $(2,1)$, qualquer partição (L, S_1, S_2) dessa instância satisfaz $L \neq \emptyset$, $S_1 \neq \emptyset$ e $S_2 \neq \emptyset$. Ou seja, nenhum dos conjuntos da partição é vazio. Isso ocorre porque L deve conter os vértices que formam a clique, enquanto S_1 e S_2 correspondem aos conjuntos independentes que particionam os demais vértices. Tal condição é essencial para assegurar a correta estrutura da partição exigida pelo problema.

Afirmação 2: Para cada $i \in \{1, \ldots, n\}$, o vértice p_i pertence ao conjunto $S_1 \cup S_2$, o vértice x_i pertence ao conjunto $L \cup S_2$ e o vértice $\overline{x}_i$ pertence ao conjunto $L \cup S_1$.

A Afirmação 2 estabelece a distribuição dos vértices associados a cada variável x_i na partição (L, S_1, S_2). O vértice auxiliar p_i deve necessariamente estar em um dos conjuntos independentes S_1 ou S_2. Já os vértices correspondentes aos literais x_i e $\overline{x}_i$ podem pertencer tanto ao conjunto da clique L quanto a um dos conjuntos independentes, porém de forma complementar: x_i está em L ou em S_2, enquanto $\overline{x}_i$ está em L ou em S_1. Essa organização é fundamental para garantir que a estrutura da partição $(2,1)$ reflita corretamente a relação de complementaridade entre os literais da variável x_i.

A prova da Afirmação 2 fundamenta-se nas restrições impostas pelas arestas obrigatórias E_1 e proibidas E_3, bem como na posição dos vértices já estabelecida na partição. Como a aresta proibida $p_i l_2 \in E_3$ existe e sabe-se que $l_2 \in L$, o vértice p_i não pode pertencer ao conjunto L, pois isso violaria a condição de que nenhuma aresta proibida pode estar presente dentro da clique. Assim, conclui-se que p_i deve estar em S_1 ou S_2.

Além disso, as arestas obrigatórias $x_i s_{11} \in E_1$ e $\overline{x}_i s_{12} \in E_1$ indicam que o vértice x_i está conectado a $s_{11} \in S_1$ e que $\overline{x}_i$ está conectado a $s_{12} \in S_2$. Como S_1 e S_2 são conjuntos independentes, o vértice conectado a um deles deve pertencer ao conjunto oposto ou à clique L.

Assim, conclui-se que $x_i \in L \cup S_2$ e $\overline{x}_i \in L \cup S_1$.

Por fim, considerando que $x_i p_i \in E_1$ e $x_i \overline{x}_i \in E_3$, tem-se duas situações possíveis. Se $x_i \in L$, então $\overline{x}_i$ não pode estar em L devido à aresta proibida, de modo que $\overline{x}_i \in S_1$, o que implica que p_i deve pertencer em S_2 para manter as conexões corretas. Caso contrário, se $x_i \in S_2$, para preservar a estrutura sem arestas proibidas, p_i deve estar em S_1 e $\overline{x}_i$ em L. Essas relações garantem que os vértices associados a cada variável estejam corretamente distribuídos na partição $(2,1)$, conforme as restrições do problema.

Afirmação 3: Para cada $j \in \{1, \ldots, m\}$, ao menos um dos vértices $\{t_1^j, t_2^j, t_3^j\}$ deve pertencer ao conjunto L.

A Afirmação 3 estabelece que, para cada cláusula j, representada pelos vértices $\{t_1^j, t_2^j, t_3^j\}$, pelo menos um desses vértices deve pertencer ao conjunto L.

Como $S_1 \cup S_2$ induz um subgrafo bipartido em G, não é possível que todos os vértices de um triângulo pertençam exclusivamente a S_1 ou a S_2. Para cada $j \in \{1, \ldots, m\}$, os vértices $\{t_1^j, t_2^j, t_3^j\}$ formam um triângulo em G_1, pois estão conectados por arestas obrigatórias. Assim, para que a partição $(2,1)$ seja válida, pelo menos um desses vértices deve estar no conjunto L, que admite a formação de cliques.

Define-se a atribuição de verdade para o conjunto X da seguinte forma: para cada variável x_i, ela é considerada falsa se, e somente se, o vértice correspondente x_i pertencer ao conjunto L. Caso contrário, se x_i estiver em S_1 ou S_2, a variável é considerada verdadeira.

Agora, suponha que exista uma cláusula $c_j = (\ell_1^j \vee \ell_2^j \vee \ell_3^j)$ que seja falsa sob essa atribuição. Pela construção da instância (V, E_1, E_3), cada literal ℓ_k^j está conectado por uma aresta proibida ao vértice t_k^j. Se o literal ℓ_k^j seja falso, então o vértice correspondente está em L, o que impede que o vértice t_k^j também esteja em L.

Assim, caso todos os literais da cláusula sejam falsos, então todos os vértices $\{t_1^j, t_2^j, t_3^j\}$ estariam em $S_1 \cup S_2$. No entanto, esses três vértices formam um triângulo em G_1 e, pela definição da partição $(2,1)$, não é permitido que um triângulo esteja inteiramente contido em $S_1 \cup S_2$. Isso gera uma contradição.

Portanto, a hipótese de que a cláusula seja falsa leva a uma contradição, o que implica que a atribuição de verdade definida satisfaz todas as cláusulas do conjunto C. Com isso, conclui-se a prova do Lema 1.

Define-se o Lema 2 da seguinte forma: se existe uma atribuição de verdade que satisfaz (X, C), então a instância particular (V, E_1, E_3) do Problema Grafo Sanduíche $(2,1)$, construída anteriormente, admite um grafo $G(V, E)$ tal que $E_1 \subseteq E$, $E \cap E_3 = \emptyset$ e G é $(2,1)$, isto é, seus vértices podem ser particionados em dois conjuntos independentes e uma clique.

Em outras palavras, caso a fórmula booleana original seja satisfatível, então é possível construir um grafo sanduíche que satisfaça todas as restrições de arestas obrigatórias e proibidas e que, além disso, apresente a estrutura de partição exigida pelo problema.

A prova deste lema inicia-se com a suposição de que exista uma atribuição de verdade que satisfaça a instância (X, C) de 3-SAT. O objetivo é construir uma partição do conjunto V

nos subconjuntos S_1, S_2 e L, de modo a definir um grafo $G(V, E)$ que satisfaça as condições da instância (V, E_1, E_3) do Problema Grafo Sanduíche $(2,1)$. Em outras palavras, é necessário garantir que todas as arestas obrigatórias estejam em E, que nenhuma das arestas proibidas pertença a E, e que a partição (S_1, S_2, L) seja tal que G seja um grafo com partição $(2,1)$, ou seja, com S_1 e S_2 formando conjuntos independentes e L formando uma clique.

Os vértices auxiliares são posicionados da seguinte forma: l_1 e l_2 pertencem ao conjunto L; s_{11} e s_{21} pertencem ao conjunto S_1; e s_{12} e s_{22} pertencem ao conjunto S_2. Para cada $i \in \{1, \ldots, n\}$, caso a variável x_i seja falsa na atribuição de verdade considerada, então posiciona-se x_i em L, $\overline{x}_i$ em S_1 e p_i em S_2. Caso contrário, se x_i seja verdadeira, posiciona-se x_i em S_2, $\overline{x}_i$ em L e p_i em S_1.

Para cada $j \in \{1, \ldots, m\}$, seja $c_j = (\ell_1^j \vee \ell_2^j \vee \ell_3^j)$ uma cláusula da instância de 3-SAT. Os vértices correspondentes t_1^j, t_2^j, t_3^j devem ser posicionados da seguinte forma: para cada $k \in \{1, 2, 3\}$, caso o literal ℓ_k^j seja falso na atribuição de verdade considerada, então t_k^j é colocado em $S_1 \cup S_2$; caso contrário, se o literal seja verdadeiro, t_k^j é posicionado em L. Como a atribuição satisfaz todas as cláusulas de (X, C), ao menos uma literal em cada cláusula é verdadeiro, o que garante que, para cada j, no máximo dois vértices t_k^j estarão em $S_1 \cup S_2$. Nesses casos, se dois vértices estiverem em $S_1 \cup S_2$, eles devem ser distribuídos alternadamente, ou seja, um em S_1 e o outro em S_2, para preservar a independência dentro de cada conjunto.

Para demonstrar que o grafo $G(V, E)$ construído é, de fato, um grafo sanduíche $(2,1)$, é necessário verificar que as seguintes condições são satisfeitas: não existe nenhuma aresta obrigatória em E_1 cujos dois extremos pertençam simultaneamente ao conjunto S_1; não existe nenhuma aresta obrigatória cujos dois extremos pertençam ambos ao conjunto S_2; e não existe nenhuma aresta proibida em E_3 cujos dois extremos pertençam ao conjunto L.

Considerando o posicionamento dos vértices, tem-se que s_{11} e s_{21} pertencem a S_1, assim como os vértices $\overline{x}_i$, t_k^j e p_i podem estar em S_1. As únicas arestas obrigatórias possíveis entre esses vértices são $\overline{x}_i p_i$ e $t_k^j t_q^j$ (com $k \neq q$). No entanto, nenhuma dessas arestas possui ambos os extremos em S_1, o que garante que não existam arestas de E_1 totalmente contidas em S_1.

De forma análoga, s_{12} e s_{22} pertencem ao conjunto S_2, assim como os vértices x_i, t_k^j e p_i podem estar em S_2. As arestas obrigatórias entre esses vértices, como $x_i p_i$ e $t_k^j t_q^j$, também não possuem ambos os extremos em S_2, o que garante que não existam arestas de E_1 totalmente contidas em S_2.

Por fim, no conjunto L, que contém os vértices l_1, l_2 e os vértices x_i, $\overline{x}_i$ e t_k^j, as únicas arestas proibidas possíveis são $\overline{x}_i x_i$ e as que conectam x_i ou $\overline{x}_i$ a t_k^j. Entretanto, nenhuma dessas arestas possui ambos os extremos em L, em razão do posicionamento definido dos vértices. Dessa forma, garante-se que não existam arestas de E_3 totalmente contidas em L.

Assim, conclui-se que a partição (L, S_1, S_2) satisfaz todas as restrições impostas pelo Problema Grafo Sanduíche $(2,1)$, garantindo a inexistência de arestas obrigatórias inteiramente contidas em S_1 ou S_2, bem como a ausência de arestas proibidas totalmente contidas em L.

Conforme a fórmula apresentada anteriormente, foram atribuídos valores de verdade às variáveis de modo que toda a expressão seja satisfeita. Assim, definiu-se $x_1 =$ Falso, $x_2 =$ Verdadeiro e $x_3 =$ Falso. Cabe ressaltar que, de acordo com a lógica proposicional clássica, a negação inverte o valor lógico da variável: se a variável possui valor Falso, sua negação assume valor Verdadeiro; de modo análogo, se a variável possui valor Verdadeiro, sua negação assume valor Falso. Substituindo esses valores na fórmula, obtém-se:

$$(\overline{x}_1 \vee x_2 \vee \overline{x}_3) = (\text{Verdadeiro} \vee \text{Verdadeiro} \vee \text{Verdadeiro}) = \text{Verdadeiro},$$

$$(x_1 \vee \overline{x}_2 \vee \overline{x}_3) = (\text{Falso} \vee \text{Falso} \vee \text{Verdadeiro}) = \text{Verdadeiro},$$

$$(x_1 \vee x_2 \vee x_3) = (\text{Falso} \vee \text{Verdadeiro} \vee \text{Falso}) = \text{Verdadeiro}.$$

Portanto, toda a fórmula é satisfeita com essa atribuição de valores, resultando em uma avaliação verdadeira para a expressão completa.

Como demonstrado, a fórmula é satisfeita com a atribuição de valores definida. A seguir, foi analisada a forma como essa atribuição se reflete na partição dos vértices do grafo.

Sabe-se que os vértices auxiliares l_1 e l_2 pertencem ao conjunto L, que representa a clique do grafo. Os vértices s_{11} e s_{21} são posicionados no conjunto S_1, enquanto s_{12} e s_{22} pertencem ao conjunto S_2. Dessa forma, compõem a base estrutural da partição $(2,1)$, servindo como referência para a distribuição dos demais vértices do grafo.

Para as variáveis, aplica-se a seguinte regra: se uma variável x_i é avaliada como falsa, o vértice correspondente x_i é alocado no conjunto L; o vértice negado $\overline{x}_i$ é posicionado no conjunto S_1; e o vértice auxiliar p_i é colocado no conjunto S_2. Por outro lado, quando a variável x_i é verdadeira, o vértice x_i é posicionado em S_2, o vértice $\overline{x}_i$ em L e o vértice p_i em S_1. Essa regra assegura que a distribuição dos vértices respeite as restrições impostas pela partição $(2,1)$.

No exemplo considerado, como $x_1 =$ Falso, o vértice x_1 é alocado em L, $\overline{x}_1$ em S_1 e p_1 em S_2. Como $x_2 =$ Verdadeiro, x_2 é posicionado em S_2, $\overline{x}_2$ em L e p_2 em S_1. Finalmente, como $x_3 =$ Falso, tem-se x_3 em L, $\overline{x}_3$ em S_1 e p_3 em S_2. Dessa forma, os vértices estão distribuídos entre os conjuntos L, S_1 e S_2 de acordo com a atribuição de valores de verdade que satisfaz a fórmula.

Na cláusula 1, $(\overline{x}_1 \vee x_2 \vee \overline{x}_3)$, todos os literais assumem valor Verdadeiro com a atribuição adotada: $\overline{x}_1$ é verdadeiro, pois $x_1 =$ Falso, x_2 é verdadeiro e $\overline{x}_3$ também é verdadeiro, uma vez que $x_3 =$ Falso. Dessa forma, os vértices t_1^1, t_2^1 e t_3^1, que representam esses literais, são todos posicionados no conjunto L.

Na cláusula 2, $(x_1 \vee \overline{x}_2 \vee \overline{x}_3)$, apenas o literal $\overline{x}_3$ assume valor Verdadeiro, pois $x_1 =$ Falso, $\overline{x}_2 =$ Falso, uma vez que $x_2 =$ Verdadeiro e $\overline{x}_3 =$ Verdadeiro. Dessa forma, o vértice t_3^2, correspondente ao $\overline{x}_3$, é colocado em L, enquanto os vértices restantes dessa cláusula são posicionados nos conjuntos S_1 ou S_2, respeitando a estrutura da partição $(2,1)$.

Na cláusula 3, $(x_1 \vee x_2 \vee x_3)$, apenas o literal x_2 assume valor Verdadeiro, pois $x_1 =$ Falso, $x_2 =$ Verdadeiro e $x_3 =$ Falso. Dessa forma, o vértice t_2^3, correspondente ao literal

x_2, é alocado no conjunto L, enquanto os vértices restantes dessa cláusula são posicionados nos conjuntos S_1 ou S_2, conforme as restrições da partição $(2,1)$.

Dessa forma, para cada cláusula, pelo menos um vértice t_k^j é alocado no conjunto L, garantindo que cada cláusula seja satisfeita e que essa satisfatibilidade esteja refletida diretamente na partição do grafo.

A Tabela 7 mostra essa configuração, evidenciando a distribuição dos vértices entre os conjuntos L, S_1 e S_2, conforme os valores de verdade atribuídos às variáveis e a forma como cada cláusula é satisfeita no exemplo apresentado.

Tabela 7 – Partição $(2,1)$ nos conjuntos L, S_1 e S_2.

L	S_1	S_2
l_1	s_{11}	s_{12}
l_2	s_{21}	s_{22}
x_1	$\overline{x}_1$	x_2
x_3	$\overline{x}_3$	p_1
$\overline{x}_2$	p_2	p_3
t_1^1	t_1^2	t_2^2
t_2^1	t_1^3	t_3^3
t_3^1	-	-
t_2^3	-	-
t_3^2	-	-

Fonte: Autoria própria.

O Problema Grafo Sanduíche $(2,1)$ é NP-completo, conforme demonstrado por Souza (2002), que apresentou uma redução do Problema 3-SAT, reconhecidamente NP-completo, para o problema em questão. A redução consiste na construção de uma instância particular (V, E_1, E_3) do Problema Grafo Sanduíche a partir de uma instância arbitrária (X, C) de 3-SAT, de modo que a existência de uma atribuição de verdade que satisfaz (X, C) corresponde exatamente à existência de um grafo $G(V, E)$ que contém todas as arestas obrigatórias, exclui as arestas proibidas e admite uma partição $(2,1)$. Como há equivalência entre as soluções das duas instâncias e o 3-SAT é NP-completo, conclui-se que o Problema Grafo Sanduíche $(2,1)$ também é NP-completo.

3.5 Problema do Grafo Sanduíche em Grafos com Partição $(2,2)$

Nesta seção, revisa-se a complexidade computacional do Problema Grafo Sanduíche para a classe de grafos com partição $(2,2)$, conforme apresentado por Souza (2002). Toda a análise desenvolvida aqui segue diretamente esse trabalho fundamental; por esse motivo, as definições, propriedades e caracterizações discutidas a seguir são extraídas e adaptadas de sua formulação original. Um grafo com partição $(2,2)$ é definido como aquele cujos vértices podem ser particionados em dois conjuntos independentes e duas cliques, respeitando as restrições impostas por essa estrutura.

O Problema Grafo Sanduíche com partição $(2,2)$ é NP-completo por meio de uma redução a partir do Problema 3-SAT, reconhecido por sua NP-completude. De forma análoga ao caso $(2,1)$ apresentado na Seção 3.4, na página 42, a construção para o Problema Grafo Sanduíche $(2,2)$ é apresentada a seguir.

PROBLEMA GRAFO SANDUÍCHE $(2,2)$

Instância: Conjunto de vértices V, conjunto de arestas obrigatórias E_1, conjunto de arestas proibidas E_3.

Questão: Existe um grafo $G(V, E)$, tal que $E_1 \subseteq E$ e $E \cap E_3 = \emptyset$, e G é (2, 2)?

A construção apresenta forte semelhança com o caso $(2,1)$, descrito na Seção 3.4, na página 42. O conjunto de vértices V é formado por dois conjuntos auxiliares: $B_1 = \{l_1, l_2, s_{11}, s_{12}, s_{21}, s_{22}\}$ e $B_2 = \{l_3, l_4, s_{31}, s_{32}, s_{41}, s_{42}\}$.

Para cada variável x_i, são adicionados os vértices correspondentes aos literais x_i, $\overline{x_i}$, além de um vértice auxiliar p_i. Para cada cláusula $c_j = (\ell_1^j \vee \ell_2^j \vee \ell_3^j)$, são incluídos três vértices t_1^j, t_2^j, t_3^j cada um associado a um literal da cláusula.

O conjunto de arestas obrigatórias E_1 foi definido contemplando três grupos principais. Primeiramente, inclui as arestas entre os vértices auxiliares:

$$\{l_1l_2, l_3l_4, l_1s_{11}, l_1s_{12}, l_2s_{21}, l_2s_{22}, l_3s_{31}, l_3s_{32}, l_4s_{41}, l_4s_{42}, s_{11}s_{12}, s_{21}s_{22}, s_{31}s_{32}, s_{41}s_{42}\}.$$

Em seguida, para cada variável x_i, considera-se o conjunto;

$$\{x_is_{11}, \overline{x}_is_{12}, x_ip_i, \overline{x}_ip_i\}.$$

Por fim, para cada cláusula c_j, define-se o conjunto:

$$\{t_1^jt_2^j, t_1^jt_3^j, t_2^jt_3^j\}.$$

O conjunto de arestas proibidas E_3 foi definido considerando quatro grupos principais. Primeiramente, inclui as arestas entre os vértices auxiliares do conjunto B_1:

$$\{l_1s_{21}, l_1s_{22}, l_2s_{11}, l_2s_{12}, s_{11}s_{21}, s_{11}s_{22}, s_{12}s_{21}, s_{12}s_{22}\},$$

e entre os vértices auxiliares do conjunto B_2:

$$\{l_3s_{41}, l_3s_{42}, l_4s_{31}, l_4s_{32}, s_{31}s_{41}, s_{31}s_{42}, s_{32}s_{41}, s_{32}s_{42}\}.$$

Em seguida, incluem-se as arestas proibidas entre vértices de B_1 e B_2:

$$\{uv : u \in B_1 \text{ e } v \in B_2\},$$

e entre vértices de B_2 e os vértices fora de B_1:

$$\{uv : u \in B_2 \text{ e } v \in V \setminus B_1\}.$$

Além disso, para cada variável x_i, com $1 \leq i \leq n$, considera-se o conjunto:

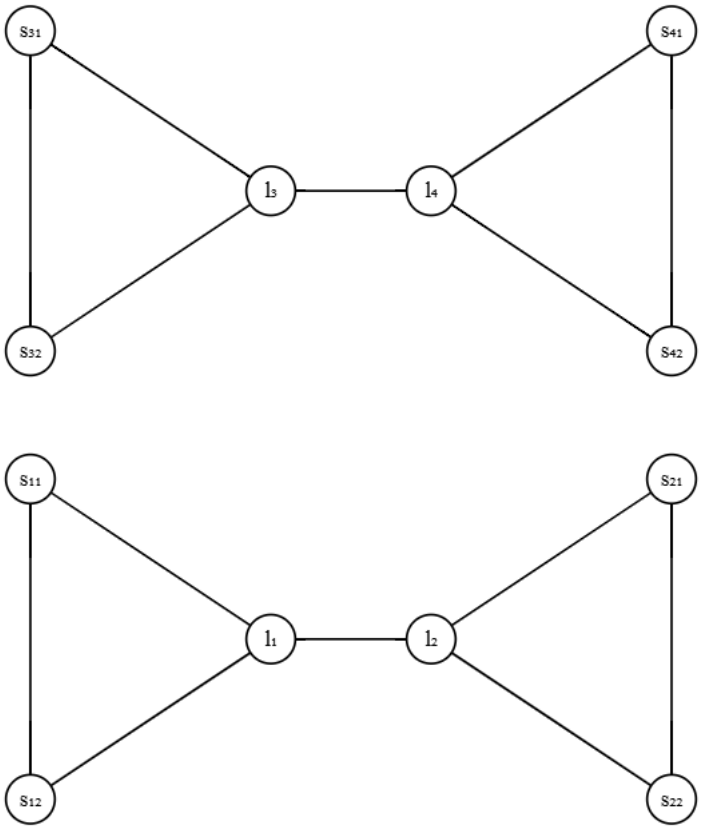
$$\{x_i\overline{x}_i, p_i l_2\}.$$

Por fim, para cada cláusula c_j, com $1 \leq j \leq m$, o conjunto de arestas proibidas é:

$$\{t_1^j t_1^{j'}, t_2^j t_2^{j'}, t_3^j t_3^{j'}\}.$$

Conforme ilustrado na Figura 29, apresenta-se o grafo base com as arestas obrigatórias definidas para a construção da instância do Problema Grafo Sanduíche $(2,2)$.

Figura 29 – Grafo base $(2,2)$.



Fonte: Modificado de Souza (2002)

Da mesma forma que no caso $(2,1)$, há dois tipos de estruturas: a Estrutura Variável, apresentada na Figura 26, na página 44, e a Estrutura Cláusula, ilustrada na Figura 27, na página 43.

Embora a ideia geral seja semelhante ao caso $(2,1)$, a construção para o Problema Grafo Sanduíche $(2,2)$ apresenta diferenças específicas na estrutura da partição dos vértices. Quando a instância especial (V, E_1, E_3) admite um grafo sanduíche $(2,2)$, ou seja, um grafo $G(V,E)$ com partição (L_1, L_2, S_1, S_2), todos os quatro conjuntos da partição são não vazios.

Os vértices auxiliares são organizados da seguinte forma: os vértices l_1 e l_2 pertencem ao conjunto L_1, os vértices l_3 e l_4 pertencem ao conjunto L_2, os vértices s_{11}, s_{21}, s_{31} e s_{41} são posicionados em S_1, e s_{12}, s_{22}, s_{32} e s_{42} pertencem a S_2.

Essa divisão garante que o grafo satisfaça as regras do problema $(2,2)$, no qual L_1 e L_2 formam duas cliques disjuntas, e S_1 e S_2 constituem dois conjuntos independentes.

Assim como o Problema Grafo Sanduíche com partição $(2,1)$ foi demonstrado ser NP-completo, o Problema Grafo Sanduíche com partição $(2,2)$ também é NP-completo. Isso se deve ao fato de que, embora a estrutura da partição seja mais elaborada, envolvendo dois conjuntos que formam cliques e dois conjuntos independentes, a dificuldade computacional do problema permanece equivalente. A prova da NP-completude do caso $(2,2)$ é estabelecida por meio de uma redução a partir do Problema 3-SAT, evidenciando que encontrar uma partição $(2,2)$ válida para o grafo é tão difícil quanto resolver um problema clássico de alta complexidade.

3.6 Problema do Grafo Sanduíche em (k, l) Δ-Limitado

Nesta seção, revisa-se a complexidade computacional do Problema Grafo Sanduíche para a classe de grafos em $(k, l)\Delta$-Limitado, considerando instâncias em que o grau máximo dos vértices é limitado em G_2, conforme apresentado por Souza (2002). Toda a análise desenvolvida aqui segue diretamente esse trabalho fundamental; por esse motivo, as definições, propriedades e caracterizações discutidas a seguir são extraídas e adaptadas de sua formulação original.

PROBLEMA GRAFO SANDUÍCHE $(k,l)\Delta$-LIMITADO

Instância: Conjunto de vértices V, conjunto de arestas obrigatórias E_1, conjunto de arestas proibidas E_3, onde G_2 é um grafo com grau máximo Δ.

Questão: Existe um grafo $G(V, E)$, tal que $E_1 \subseteq E$ e $E \cap E_3 = \emptyset$, o qual é um grafo (k,l)?

A classificação completa do Problema Grafo Sanduíche limitado por grau máximo, demonstrando que ele pode ser resolvido em tempo polinomial quando $k \leq 2$ ou $\Delta \leq 3$, e que é NP-completo nos demais casos. Para sustentar essa classificação, a autora formulou dois lemas que comprovam essas afirmações.

O primeiro lema estabelece que, se o Problema Grafo Sanduíche $(k,l)\Delta$-Limitado pode ser resolvido em tempo polinomial, então o Problema Grafo Sanduíche $(k,l+1)$ Δ-Limitado também é solucionável em tempo polinomial.

Para resolver o Problema Grafo Sanduíche $(k,l+1)$ Δ-Limitado, parte-se do pressuposto de que já existe um algoritmo polinomial para o caso $(k,l)\Delta$-Limitado. A ideia central baseia-se no fato de que, se existir um grafo sanduíche com $l + 1$ cliques, então pelo menos uma dessas cliques também está contida em G_2. Isso ocorre porque, por definição do problema, o grafo final G deve satisfazer $E_1 \subseteq E \subseteq E_2$, em que E_1 representa o conjunto de arestas obrigatórias e E_2 o conjunto de arestas permitidas. Assim, todas as arestas que compõem uma clique em G também estão presentes em G_2.

Nesse contexto, o conjunto S representa um subconjunto de vértices de G_2 que é candidato a ser a nova clique adicionada ao grafo final G. Como o grau máximo de G_2 é Δ,

o tamanho máximo de uma clique em G_2 é $\Delta + 1$. Dessa forma, o algoritmo analisa todos os subconjuntos S de tamanho menor ou igual a $\Delta + 1$. Para cada subconjunto S, verifica-se em G_2 se ele constitui uma clique, isto é, se todos os seus vértices estão mutuamente conectados. Caso S não forme uma clique, ele é descartado. Caso contrário, S é considerado como uma nova clique potencial para compor a solução.

Em seguida, remove-se S do conjunto de vértices V, restando $V \setminus S$, e aplica-se o algoritmo já existente para o caso (k,l) sobre os vértices remanescentes. Se o algoritmo encontrar uma solução, isso indica que existe um grafo sanduíche que utiliza S como clique adicional, combinada com a solução obtida para os demais vértices. Dessa forma, obtém-se um grafo sanduíche com $l + 1$ cliques.

Esse procedimento é eficiente porque o parâmetro Δ é fixo, o que limita o tamanho dos conjuntos S e, consequentemente, reduz a quantidade de subconjuntos que precisam ser testados. Com isso, o número de verificações seja polinomial, garantindo que a execução do algoritmo para $l + 1$ cliques também ocorra em tempo polinomial.

O segundo lema estabelece que, se $k \leq 2$, então o Problema Grafo Sanduíche $(k,l)\Delta$-Limitado pode ser resolvido em tempo polinomial.

Conforme apresentado na Seção 3.3, na página 40, o Problema Grafo Sanduíche para grafos bipartidos $(2,0)$ é solucionado em tempo polinomial. Como a propriedade de ser bipartido é hereditária e pode ser verificada diretamente sobre o grafo obrigatório G_1, conclui-se que o caso $(1,0)$ segue a mesma lógica, pois representa uma estrutura ainda mais restrita.Assim, ambos os casos podem ser resolvido em tempo polinomial. Dessa forma, para $k \leq 2$ o Problema Grafo Sanduíche Δ-Limitado também é solucionável em tempo polinomial.

Para $k \leq 2$ e $l \geq 0$, o Problema do Grafo Sanduíche $(k,l)\Delta$-Limitado é polinomial. Esse resultado decorre diretamente do lema apresentado anteriormente, que estabelece que, se o problema é solucionado em tempo polinomial para $(k,l)\Delta$-Limitado, então também o será para $(k,l+1)$ Δ-Limitado. Como para $k \leq 2$ o problema é polinomial no caso base, a aplicação iterativa do lema garante que ele permanece polinomial para qualquer valor de $l \geq 0$.

O Problema Grafo Sanduíche é polinomial nos casos em que $k \leq 2$ ou $\Delta \leq 3$ e torna-se NP-completo quando $k \geq 3$ e $\Delta \geq 4$. Isso ocorre porque, em grafos com vértices altamente conectados, o processo de reconhecimento e partição em conjuntos independentes e cliques torna-se computacionalmente mais complexo, não havendo algoritmos eficientes conhecidos para resolvê-lo em todos os casos. Dessa forma, a complexidade do problema aumenta consideravelmente quando o grau máximo ultrapassa 3, tornando sua solução prática inviável em grande parte das situações.

4 CONCLUSÃO

Neste trabalho, foi apresentada uma revisão bibliográfica sobre o Problema do Grafo Sanduíche, destacando sua formulação, aplicações práticas e comportamento em diferentes cenários de complexidade. Inicialmente, foram abordados os conceitos fundamentais da Teoria dos Grafos, da complexidade computacional e do problema de partição (k,l), que serviram como base teórica para a análise.

Após a apresentação dos conceitos teóricos fundamentais, foi discutido o Problema do Grafo Sanduíche, incluindo sua definição formal e principais características. Em seguida, analisaram-se cinco variações desse problema: split $(1,1)$, bipartido $(2,0)$, partição $(2,1)$, partição $(2,2)$ e $(k,l)\Delta$-Limitado. Cada uma dessas variações permitiu observar como a estrutura do grafo influencia diretamente a complexidade computacional associada à sua resolução.

Em relação aos resultados obtidos para cada variação, verificou-se que o caso split $(1,1)$ é polinomial, sendo possível comprovar esse resultado por meio da redução para o problema 2-SAT. Para o caso bipartido $(2,0)$, observou-se que se trata de uma classe hereditária, o que também permite uma solução em tempo polinomial. Já as partições $(2,1)$ e $(2,2)$ foram classificadas como NP-completo, com comprovação realizada através da redução para o problema 3-SAT, evidenciando o aumento da complexidade computacional. Por fim, no caso $(k,l)\Delta$-Limitado, constatou-se que o Problema do Grafo Sanduíche é polinomial quando $k \leq 2$ ou $\Delta \leq 3$ e torna-se NP-completo quando $k \geq 3$ e $\Delta \geq 4$, mostrando como o grau máximo dos vértices impacta diretamente a dificuldade do problema.

Os resultados discutidos ao longo deste trabalho reforçam a importância da análise de complexidade como ferramenta para compreender os limites entre casos tratáveis e intratáveis. Além de seu valor teórico, o Problema do Grafo Sanduíche apresenta aplicações práticas em áreas como biologia computacional e raciocínio temporal, evidenciando sua relevância interdisciplinar. Assim, este estudo contribui para uma compreensão mais clara das fronteiras de complexidade relacionadas à estrutura dos grafos e ao impacto de parâmetros como k,l e Δ na dificuldade de resolução do problema.

Como resultado adicional, este texto foi apresentado nos eventos integrados do IF Goiano: VII Integra IF Goiano, 15º Seminário de Avaliação dos Programas de Pós-graduação, 14º Congresso de Pesquisa e Pós-graduação, 7ª Maratona de Inovação da Diretoria de Extensão e 2ª Semana de Integração Acadêmica, realizados no Campus Rio Verde, reforçando sua contribuição acadêmica e científica no contexto institucional.

Como trabalhos futuros, pretende-se investigar novas classes de grafos (k,l) para o Problema do Grafo Sanduíche, com o objetivo de identificar novos casos pertencentes às classes P e NP-completo, ampliando o entendimento sobre as fronteiras de complexidade e as condições estruturais que tornam o problema tratável ou intratável.

Referências

AIRES, V. P. S. *Grafos Rotulados, Grafos Graciosos e Problemas de Coloração Especiais*. Manaus, AM, 2015. Orientadora: Rosiane de Freitas Rodrigues, D.Sc. Citado 4 vezes nas páginas 5, 6, 10 e 11.

ALVES, S. R. *Estudo da complexidade de grafos bem cobertos-(r, l): reconhecimento, problemas sanduíche e probe*. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, COPPE, Rio de Janeiro, RJ, Dezembro 2019. Orientadores: Sulamita Klein, Luerbio Faria e Fernanda Vieira Dias Couto. Citado na página 30.

ASPVALL, B.; PLASS, M. F.; TARJAN, R. E. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information processing letters*, Elsevier, v. 8, n. 3, p. 121–123, 1979. Citado 6 vezes nas páginas 15, 16, 17, 19, 20 e 21.

BRANDSTÄDT, A. Partitions of graphs into one or two independent sets and cliques. *Discrete Mathematics*, Elsevier, v. 152, n. 1-3, p. 47–54, 1996. Citado 3 vezes nas páginas 30, 31 e 32.

CERIOLI, M. R. et al. The homogeneous set sandwich problem. *Information Processing Letters*, Elsevier, v. 67, n. 1, p. 31–35, 1998. Citado na página 34.

COOK, S. A. The complexity of theorem-proving procedures. In: *Proceedings of the Third Annual ACM Symposium on Theory of Computing*. New York, NY, USA: ACM, 1971. p. 151–158. Citado na página 25.

CORDEIRO, A. V. V. *Grafos Perfeitos, Cliques e Colorações*. Manaus, AM, 2015. Orientadora: Rosiane de Freitas Rodrigues, D.Sc.; Organizado pela Fundação de Amparo à Pesquisa do Estado do Amazonas. Citado na página 9.

CORMEN, T. H. et al. *Introduction to Algorithms*. 2nd. ed. Cambridge, Massachusetts; Boston, MA: The MIT Press and McGraw-Hill Book Company, 2001. First edition 1990. ISBN 0-262-03293-7. Citado 5 vezes nas páginas 11, 12, 13, 14 e 23.

COUTO, F. V. D. *Complexidade dos problemas sanduíche e probe para subclasses de grafos-(k, l)*. Tese (Doutorado) — PhD thesis, Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ, 2016. Citado 2 vezes nas páginas 29 e 34.

DANTAS, S.; FIGUEIREDO, C. M. D.; FARIA, L. On the complexity of (k, l)-graph sandwich problems. In: SPRINGER. *Graph-Theoretic Concepts in Computer Science: 28th International Workshop, WG 2002 Český Krumlov, Czech Republic, June 13–15, 2002 Revised Papers 28*. [S.l.], 2002. p. 92–101. Citado na página 35.

FARIA, M. V. M. *Rotulagem automática de imagens da Web com busca em largura: um estudo de caso na competição Dog Breed Identification*. Serra, ES, 2024. Orientador: Prof. Dr. Francisco de Assis Boldt. Citado na página 11.

FEOFILOFF, P.; KOHAYAKAWA, Y.; WAKABAYASHI, Y. *Uma Introdução Sucinta à Teoria dos Grafos*. 2004. Material de minicurso na II Bienal da SBM. Acesso em: 2025-07-07. Disponível em: <http://www.ime.usp.br/~pf/teoriadosgrafos/>. Citado 3 vezes nas páginas 4, 6 e 41.

FREIRE, Y. K. F. *Um survey sobre resultados recentes em coloração backbone*. Crateús, CE, 2024. Orientador: Prof. Dr. Rennan Ferreira Dantas. Citado 6 vezes nas páginas 6, 7, 8, 10, 29 e 30.

FREITAG, F. R. d. S. *Estudo do problema 2-SAT: algoritmos eficientes e suas aplicações na programação competitiva*. Quixadá, CE, 2025. Orientador: Prof. Dr. Atílio Gomes Luiz; Coorientador: Prof. Dr. Paulo de Tarso Guerra Oliveira. Citado 3 vezes nas páginas 5, 13 e 15.

GOLUMBIC, M. C.; KAPLAN, H.; SHAMIR, R. Graph sandwich problems. *Journal of Algorithms*, Elsevier, v. 19, n. 3, p. 449–473, 1995. Citado 3 vezes nas páginas 35, 36 e 37.

HAMMER, P. L.; SIMEONE, B. The splittance of a graph. *Combinatorica*, Springer, v. 1, p. 275–284, 1981. Citado 4 vezes nas páginas 9, 21, 22 e 23.

ISERNHAGEN, M.; BARBOSA, R. M. Conjuntos independentes maximais em grafos: As classes m(t) e i(t). In: SOCIEDADE BRASILEIRA DE PESQUISA OPERACIONAL. *Anais do XXXVIII Simpósio Brasileiro de Pesquisa Operacional (SBPO)*. Goiânia, GO, 2006. p. 970–979. Citado 2 vezes nas páginas 8 e 9.

KARP, R. M. Reducibility among combinatorial problems. *Complexity of Computer Computations*, Springer, p. 85–103, 1972. Citado na página 26.

LOZADA, L. A. P. *Tópicos na classe dos grafos clique*. Dissertação (Mestrado) — Instituto de Matemática, Estatística e Ciência da Computação, UNICAMP, Campinas, SP, Abril 1996. Orientadora: Profa. Célia Picinin de Mello. Citado na página 8.

MACAMBIRA, A. F. U. et al. Tópicos em otimização inteira. *UFRJ*, 2022. Citado 4 vezes nas páginas 5, 6, 7 e 8.

OLIVEIRA, M. P. M. S. de. *Teias matemáticas: frentes na ciência e na sociedade*. [S.l.]: Imprensa da Universidade de Coimbra/Coimbra University Press, 2004. Citado na página 26.

PINTO, G. P.; SILVA, L. d. A. da. Algoritmos de clique máximo em redes complexas. *Anais do Congresso da Sociedade Brasileira de Computação (CSBC)*, Curitiba, Brasil, 2021. Citado na página 9.

RECUERO, R. Contribuições da análise de redes sociais para o estudo das redes sociais na internet: o caso da hashtag# tamojuntodilma e# calaabocadilma. *Revista Fronteiras*, v. 16, n. 2, 2014. Citado na página 4.

SOUZA, S. D. de. *Partições em Grafos: Caracterizações, Algoritmos e Complexidade*. VIII, 78 p. Tese (Tese de Doutorado) — Universidade Federal do Rio de Janeiro, COPPE, Rio de Janeiro, RJ - Brasil, junho 2002. Programa: Engenharia de Sistemas e Computação. Citado 9 vezes nas páginas 40, 41, 42, 43, 44, 45, 50, 52 e 53.

SZWARCFITER, J. L. *Teoria Computacional de Grafos: Os Algoritmos*. 1. ed. Rio de Janeiro, RJ: Elsevier Editora Ltda., 2018. Com programas em Python por Fabiano S. Oliveira e Paulo E. D. Pinto. Série SBC — Sociedade Brasileira de Computação. ISBN 978-85-352-8884-1. Citado 3 vezes nas páginas 27, 28 e 30.

TARJAN, R. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, v. 1, n. 2, p. 146–160, June 1972. Citado 2 vezes nas páginas 17 e 18.