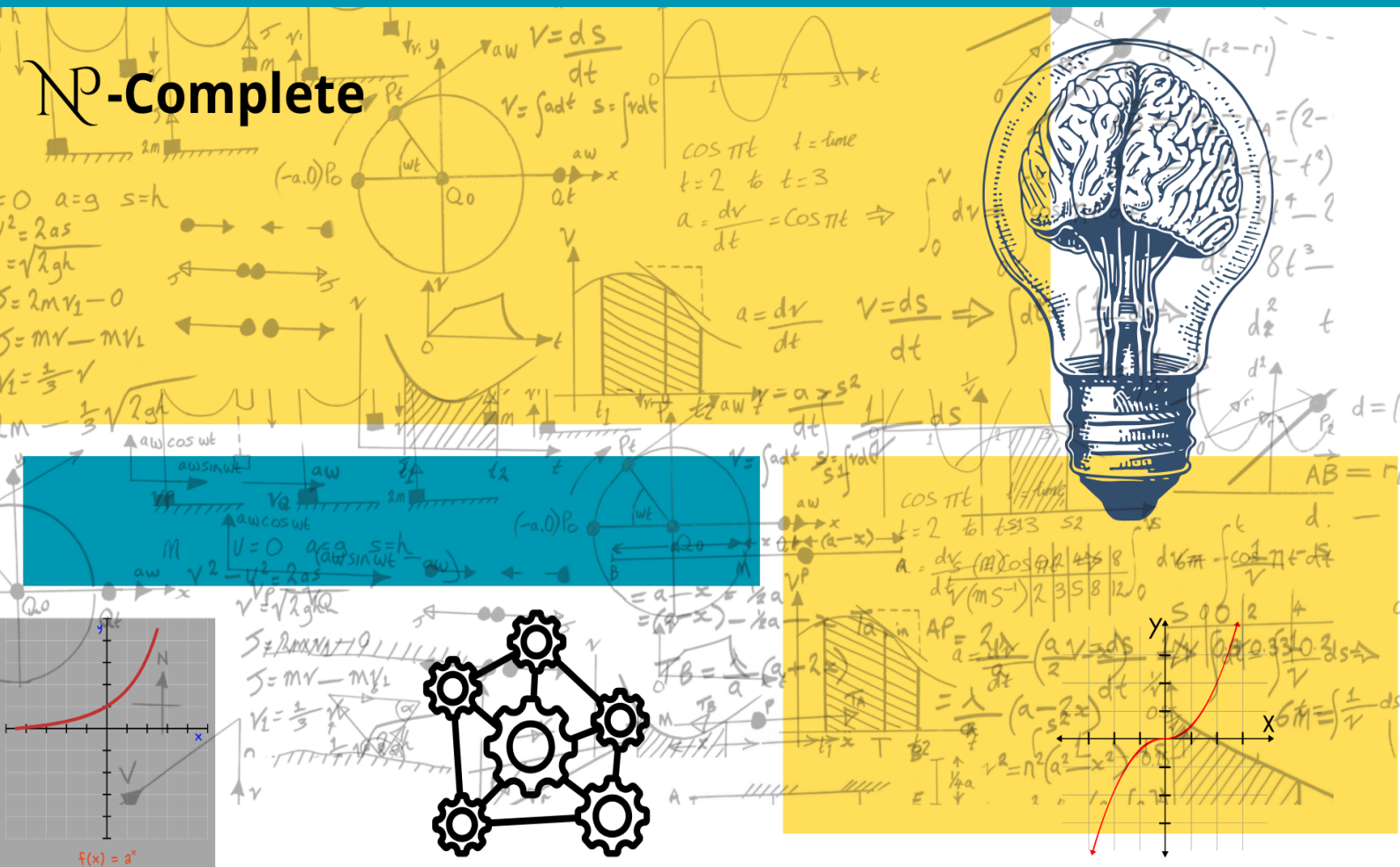
AJC
Academic Journal on Computing, Engineering and Applied Mathematics
EAM

Volume 7
Special Issue 2
2026

Academic Journal on Computing, Engineering and Applied Mathematics



UNIVERSIDADE FEDERAL DO TOCANTINS

ISSN: 2675-3588

Universidade Federal do Tocantins

Reitor
Profa. Dra. Maria Santana Ferreira dos Santos

Vice-Reitor
Prof. Dr. Marcelo Leineker Costa

Pró-Reitoria de Graduação
Profa. Dra. Valdirene Gomes dos Santos de Jesus

Pró-Reitoria de Pesquisa e Pós-Graduação
Profa. Dra. Flávia Lucila Tonani

Pró-Reitoria de Extensão e Cultura
Bruno Barreto Amorim Campos

Pró-Reitoria de Administração e Finanças
Me. Carlos Alberto Moreira de Araújo Júnior

Pró-Reitoria de Assuntos Estudantis e Comunitários
Prof. Dr. Kherlley Caxias Batista Barbosa

Pró-Reitoria de Avaliação e Planejamento
Prof. Dr. Eduardo Andrea Lemus Erasmo

Pró-reitoria de Gestão e Desenvolvimento de Pessoas
Dra. Michelle Matilde Semiguem Lima Trombini Duarte

Pró-Reitoria de Tecnologia da Informação e Comunicação
Olívia Tozzi Bittencourt

Direção do Campus de Palmas
Prof. Dr. Moisés de Souza Arantes Neto

Coordenação do Curso de Ciência da Computação
Prof. Dr. Ary Henrique Morais de Oliveira

ISSN: 2675-3588

Dados Internacionais de Catalogação na Publicação (CIP)

Academic Journal on Computing, Engineering and Applied Mathematics (AJCEAM) [recurso eletrônico] / Universidade Federal do Tocantins, Curso de Ciência da Computação. – vol. 07, n. 02 ([october/february], 2026) – Palmas - TO, UFT, 2026. ISSN nº 2675-3588.

Quadrimestral no primeiro ano de publicação 2020
Semestral.
Disponível em:
https://sistemas.uft.edu.br/periodicos/index.php/AJCEAM/index

1. Ciência da Computação - periódico. 2. Matemática Aplicada. 3. Computação Aplicada. 4. Engenharias. 5. Ciências Exatas. I. Universidade Federal do Tocantins.

CDD 22.ed. 004

Ficha Catalográfica elaborada por Edson de Sousa Oliveira – CRB/2 – 1069.

ISSN: 2675-3588

Expediente

Realização
Fundação Universidade Federal do Tocantins (UFT)
Quadra 109 Norte, Avenida NS-15, ALCNO-14 | Bloco III | sala 214 |Plano Diretor Norte | 77001-090 | Palmas / TO | Brasil

Periodicidade
Este periódico possui periodicidade semestral e utiliza a Licença Creative Commons 4.0 - CC BY-NC 4.0. Contudo, a publicação dos artigos em modalidade avançada ou ahead of print, ou seja, tão logo os manuscritos aprovados sejam editados para publicação, é possível. O AJCEAM não possui taxas de publicação, tanto pouco de submissão de manuscritos, sendo totalmente gratuita para autores e leitores.

Indexadores
Google Acadêmico, desde 9 de maio de 2020
International Standard Serial Number – ISSN, desde 28 de maio de 2020
Crossref, desde 7 de junho de 2020
Revistas de Livre Acesso – LivRe, desde 24 de junho de 2020
Diretório das revistas científicas eletrônicas brasileiras – Miguilim, desde novembro de 2022

ISSN: 2675-3588

Sumário

ISSN: 2675-3588

Editorial (Português)
Quando a Teoria Ensina: Grafos e Computação em Perspectiva Pedagógica

Tanilson Dias dos Santos (Organizador)[1]

[1] *Universidade Federal do Tocantins , tanilson.dias@uft.edu.br*

Resumo—*Esta edição especial da Revista AJCEAM reúne trabalhos oriundos das disciplinas de Teoria dos Grafos e Teoria da Computação, com forte caráter pedagógico e formativo. Os artigos apresentam abordagens didáticas, exemplos lúdicos e exposições auto-contidas de problemas clássicos da literatura, visando apoiar o aprendizado de alunos de graduação. Embora não tragam novos resultados científicos, os trabalhos contribuem para a melhor compreensão conceitual de temas tradicionalmente considerados complexos. A edição constitui, ainda, uma homenagem ao esforço acadêmico e à excelência demonstrada pelos estudantes de Ciência da Computação.*

Palavras-chave—*Contribuição Pedagógica. Problemas Computacionais. Teoria da Computação. Teoria dos Grafos.*

I. O QUE VOCÊ VAI ENCONTRAR NESTE ESCRITO?

Caríssimo leitor, esta edição especial da Revista AJCEAM reúne um conjunto de trabalhos oriundos de atividades desenvolvidas no âmbito das disciplinas de Teoria dos Grafos e Teoria da Computação, oferecendo uma coletânea cuidadosamente organizada com forte caráter didático, pedagógico e formativo. Os artigos aqui apresentados resultam de esforços acadêmicos que aliam rigor conceitual, criatividade e preocupação com a clareza na exposição de temas clássicos e fundamentais da Computação Teórica.

Os trabalhos desta edição não têm como objetivo a apresentação de novos resultados científicos. Em vez disso, lançam luz sobre problemas consagrados da literatura, frequentemente reconhecidos por sua complexidade conceitual e, por vezes, por dificuldades de assimilação por parte dos discentes. Nesse sentido, os autores propõem abordagens pedagógicas, exemplos lúdicos e discussões guiadas que favorecem uma compreensão mais acessível e aprofundada dos temas tratados, sem abrir mão da precisão teórica.

Uma característica marcante dos artigos que compõem esta edição especial é o seu caráter auto-contido: todos os conceitos, definições e fundamentos necessários à compreensão dos problemas abordados são apresentados nos próprios textos. Essa escolha editorial reforça a proposta de que os trabalhos possam ser utilizados como produtos de apoio ao ensino, servindo como material complementar para estudantes de graduação que desejem aprender, revisar ou se aprofundar em tópicos relevantes de Teoria dos Grafos e Teoria da Computação.

Os aspectos técnicos de cada problema são apresentados de forma intencionalmente superficial, priorizando a intuição, o entendimento conceitual e as ideias centrais envolvidas. Adicionalmente, cada trabalho traz reflexões sobre suas próprias contribuições, destacando pontos sutis que podem passar despercebidos em uma leitura apressada. Os trabalhos relacionados apresentados nos artigos contextualizam o leitor com resultados sólidos e recentes da literatura, enquanto comentários adicionais e propostas de trabalhos futuros aparecem como convite à continuidade do estudo e da pesquisa.

Por fim, esta edição especial pode ser entendida como uma verdadeira ode ao esforço acadêmico dos alunos do curso de Ciência da Computação, que conseguiram materializar, na forma de artigos

científicos, a excelência demonstrada ao longo das aulas teóricas e das atividades práticas. Trata-se de um testemunho do potencial formativo das disciplinas e do compromisso dos discentes com a construção, a comunicação e a reflexão crítica do conhecimento científico.

Desejamos ao leitor uma leitura proveitosa e inspiradora.

Prof. Dr. Tanilson Dias dos Santos
Organizador desta Edição Especial

ISSN: 2675-3588

MAX-2SAT: Contribuições Pedagógicas para o Aprendizado no Escopo da Teoria da Computação

MAX-2SAT: Reflections and Pedagogical Practices within the Scope of the Theory of Computation Course

Raphael Sales de Souza[1], Thiago Gonzaga dos Santos[1], Daniel Martins da Silva[1] e Tanilson Dias dos Santos[1]

sales.raphael@mail.uft.edu.br thiago.gonzaga@mail.uft.edu.br danielmartins@mail.uft.edu.br tanilson.dias@mail.uft.edu.br
[1] *Universidade Federal do Tocantins, Curso de Ciência da Computação, Tocantins, Brasil*

Data de recebimento do manuscrito: 28/11/2025
Data de aceitação do manuscrito: 27/01/2026
Data de publicação: 10/02/2026

Resumo—Este artigo apresenta uma demonstração formal e didática da $\mathcal{NP}$-completude do problema Maximum 2-Satisfiability (Max-2SAT) por meio de redução polinomial a partir do problema Clique. O Max-2SAT, variante de maximização do problema SAT em que cada cláusula contém no máximo dois literais, questiona se existe uma valoração booleana capaz de satisfazer pelo menos k cláusulas de uma fórmula em forma normal conjuntiva. Embora o problema 2-SAT seja resolvido em tempo polinomial, sua versão de maximização é $\mathcal{NP}$-completa. A demonstração utiliza uma construção com variável auxiliar que mapeia estruturas de grafos em fórmulas booleanas, estabelecendo correspondência biunívoca entre cliques e valorações satisfatórias. Como contribuições pedagógicas, o trabalho apresenta: (i) prova formal detalhada de $\mathcal{NP}$-pertinência e $\mathcal{NP}$-dificuldade; (ii) construção explícita da redução Clique $\leq_p$ Max-2SAT com figuras ilustrativas; (iii) exemplo completo comentado passo a passo; (iv) pseudocódigo do verificador polinomial; e (v) discussões sobre armadilhas comuns e estratégias de compreensão. O material produzido visa facilitar o aprendizado de reduções polinomiais e fortalecer a compreensão sobre a fronteira entre tratabilidade e intratabilidade computacional.

Palavras-chave—Max-2SAT, $\mathcal{NP}$-completude, Redução Polinomial, Clique, Teoria da Complexidade, Satisfatibilidade Booleana

Abstract—*This paper presents a formal and pedagogical demonstration of the $\mathcal{NP}$-completeness of the Maximum 2-Satisfiability (Max-2SAT) problem through polynomial reduction from the Clique problem. Max-2SAT, a maximization variant of the SAT problem where each clause contains at most two literals, asks whether there exists a Boolean assignment capable of satisfying at least k clauses of a formula in conjunctive normal form. Although the 2-SAT problem is solvable in polynomial time, its maximization version is $\mathcal{NP}$-complete. The demonstration employs a construction with an auxiliary variable that maps graph structures into Boolean formulas, establishing a bijective correspondence between cliques and satisfying assignments. As pedagogical contributions, this work presents: (i) detailed formal proof of $\mathcal{NP}$-membership and $\mathcal{NP}$-hardness; (ii) explicit construction of the Clique $\leq_p$ Max-2SAT reduction with illustrative figures; (iii) complete step-by-step annotated example; (iv) pseudocode for the polynomial verifier; and (v) discussions about common pitfalls and comprehension strategies. The material produced aims to facilitate the learning of polynomial reductions and strengthen understanding of the boundary between tractability and computational intractability.*

Keywords—*Max-2SAT, NP-Completeness, Polynomial Reduction, Clique, Complexity Theory, Boolean Satisfiability*

I. Introdução

A Teoria da Computação estabelece os fundamentos matemáticos para compreender os limites da computação, classificando problemas segundo sua complexidade computacional. Entre as classes de complexidade, a classe $\mathcal{NP}$ (*Nondeterministic Polynomial Time*) e, em especial, os problemas $\mathcal{NP}$-completos ocupam papel central, tanto do ponto de vista teórico quanto prático. Um problema é $\mathcal{NP}$-completo se pertence à classe $\mathcal{NP}$ e todo problema em $\mathcal{NP}$ pode ser reduzido a ele em tempo polinomial, caracterizando-o como um dos mais representativos quanto à dificuldade computacional.

O conceito de $\mathcal{NP}$-completude foi introduzido por Stephen Cook em 1971 [1], por meio do Teorema de Cook–Levin, que estabeleceu o problema SAT (*Satisfiability*)

Dados de contato: Raphael Sales de Souza, sales.raphael@mail.uft.edu.br

como o primeiro problema $\mathcal{NP}$-completo. Desde então, milhares de problemas foram classificados como $\mathcal{NP}$-completos através de reduções polinomiais, compondo uma ampla rede de equivalências que fundamenta a noção moderna de intratabilidade algorítmica.

Dentre os problemas derivados de SAT, o *Maximum 2-Satisfiability* (Max-2SAT) ocupa posição de destaque. Trata-se de uma variante de maximização em que cada cláusula contém no máximo dois literais, e o objetivo é determinar uma valoração que satisfaça o maior número possível de cláusulas. Na versão de decisão, dada uma fórmula em forma normal conjuntiva e um inteiro k, pergunta-se se existe uma atribuição que satisfaça pelo menos k cláusulas. Embora o problema 2-SAT seja resolvido em tempo polinomial [2], sua versão de maximização (Max-2SAT) é $\mathcal{NP}$-completa, conforme demonstrado por Garey, Johnson e Stockmeyer [3].

O Max-2SAT possui aplicações práticas relevantes, como otimização de circuitos eletrônicos, análise de dependências em sistemas de software, depuração de hardware, modelagem de redes biológicas e problemas de agendamento com restrições binárias. Além disso, algoritmos de aproximação e heurísticas para Max-2SAT são estudados de forma ampla na literatura, reforçando sua importância tanto teórica quanto aplicada.

O objetivo deste artigo é apresentar, de maneira didática, a demonstração da $\mathcal{NP}$-completude do Max-2SAT utilizando a redução polinomial *Clique* $\leq_p$ *Max-2SAT*. Essa redução não foi explorada em sala de aula e permite discutir a relação entre problemas de grafos e problemas de lógica proposicional.

É importante ressaltar que a principal contribuição deste manuscrito é de natureza *pedagógica*, consistindo na sistematização detalhada e acessível da demonstração da $\mathcal{NP}$-completude do Max-2SAT. Não são propostos novos resultados teóricos ou algorítmicos; a redução Clique $\leq_p$ Max-2SAT aqui apresentada é conhecida na literatura [4]. O valor do trabalho reside na exposição didática estruturada, com exemplos comentados, figuras ilustrativas e discussões sobre armadilhas conceituais, voltada para estudantes e docentes de disciplinas de Teoria da Computação.

As principais contribuições deste trabalho incluem a demonstração formal da $\mathcal{NP}$-pertinência e da $\mathcal{NP}$-dificuldade do Max-2SAT, a construção explícita e detalhada da redução polinomial Clique $\leq_p$ Max-2SAT, figuras ilustrativas destacando como cada parte do grafo é traduzida para cláusulas 2-SAT, um exemplo completo e comentado exibindo todas as etapas da transformação, o pseudocódigo do verificador polinomial para a versão de decisão do Max-2SAT, e discussões pedagógicas sobre armadilhas comuns e estratégias para compreender reduções entre problemas de grafos e fórmulas booleanas.

O restante deste artigo está organizado da seguinte forma: a Seção 2 – *Preliminares* – apresenta as preliminares necessárias, incluindo definições formais de classes de complexidade, reduções polinomiais e os problemas Clique e Max-2SAT; a Seção 3 – *Trabalhos Relacionados* – revisa trabalhos relacionados; a Seção 4 – *Descrição do Problema* – descreve em detalhes o problema Max-2SAT; a Seção 5 – *Demonstração e Contribuições* – apresenta a prova de $\mathcal{NP}$-completude por meio da redução Clique $\leq_p$ Max-2SAT; a Seção 6 – *Resultados e Reflexões* – discute resultados e reflexões.

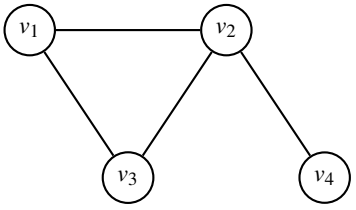


Figura 1: Grafo G1

II. PRELIMINARES

São definidos a seguir os conceitos fundamentais que embasam o restante deste trabalho, abrangendo classes de complexidade, reduções polinomiais, fórmulas em FNC e as especificações formais dos problemas *Clique* e *Max-2SAT*, que constituem, respectivamente, o problema Atacado e o problema Alvo da redução apresentada na Seção 5.

A seguir são apresentados os conceitos fundamentais de teoria dos grafos, que serão essenciais para compreender a redução Clique $\leq_p$ Max-2SAT. Um grafo é uma estrutura matemática que modela relações entre objetos. Formalmente, um grafo G é definido por um par ordenado $G = (V(G), E(G))$, onde $V(G)$ representa um conjunto finito e não vazio de vértices (também chamados de nós) e $E(G)$ representa um conjunto de arestas, sendo cada aresta um par de vértices (u, v) com $u, v \in V(G)$ e $u \neq v$. Quando existe uma aresta $(u, v) \in E(G)$, dizemos que os vértices u e v são adjacentes ou vizinhos.

O grau de um vértice $v \in V(G)$, denotado por $\deg(v)$, corresponde ao número de arestas incidentes a ele, ou equivalentemente, ao número de vizinhos que v possui no grafo. Um subgrafo de G é um grafo $G' = (V(G'), E(G'))$ tal que $V(G') \subseteq V(G)$ e $E(G') \subseteq E(G)$, onde todas as arestas de $E(G')$ conectam apenas vértices pertencentes a $V(G')$.

Para ilustrar esses conceitos, considera-se o grafo G1 da Figura 1 com quatro vértices $V(G) = \{v_1, v_2, v_3, v_4\}$ e quatro arestas $E(G) = \{(v_1, v_2), (v_1, v_3), (v_2, v_3), (v_2, v_4)\}$. Neste grafo, o vértice v_2 possui grau 3, pois está conectado a três outros vértices (v_1, v_3 e v_4); os vértices v_1 e v_3 possuem grau 2, cada um conectado a dois vizinhos; e o vértice v_4 possui grau 1, estando conectado apenas a v_2.

Uma clique é um subconjunto de vértices $C \subseteq V(G)$ tal que todo par de vértices distintos em C é adjacente. Formalmente, para quaisquer $u, v \in C$ com $u \neq v$, temos $(u, v) \in E(G)$. O tamanho de uma clique é o número de vértices que ela contém.

A Figura 2 ilustra o conceito de clique de forma detalhada. Nessa figura, o grafo possui quatro vértices $\{v_1, v_2, v_3, v_4\}$ e as arestas $\{(v_1, v_2), (v_1, v_3), (v_2, v_3), (v_2, v_4)\}$. O retângulo tracejado destaca o subconjunto $\{v_1, v_2, v_3\}$, que forma uma clique de tamanho 3. Para verificar que esse conjunto é de fato uma clique, observa-se que existem arestas conectando todos os pares possíveis dentro dele: a aresta (v_1, v_2) conecta v_1 a v_2, a aresta (v_1, v_3) conecta v_1 a v_3, e a aresta (v_2, v_3) conecta v_2 a v_3. Como cada par de vértices do conjunto está conectado por uma aresta, a condição de clique é satisfeita.

O vértice v_4, representado em cinza mais escuro na figura, não faz parte dessa clique. Embora v_4 esteja conectado a

 ISSN: 2675-3588

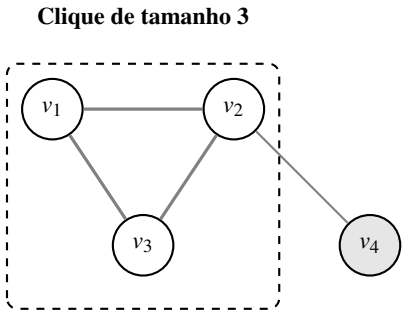


Figura 2: Grafo com clique de tamanho 3 formada por $\{v_1; v_2; v_3\}$.

v_2 pela aresta (v_2, v_4), ele não possui arestas com v_1 nem com v_3. Portanto, se fosse incluído v_4 no conjunto, os pares (v_1, v_4) e (v_3, v_4) não seriam adjacentes, violando a definição de clique. Esse exemplo ilustra por que uma clique exige conectividade total entre todos os seus membros, e não apenas conexões parciais.

A seguir são apresentados os conceitos de complexidade computacional, um problema de decisão é um problema cuja resposta é “sim” ou “não”.

Uma Máquina de Turing é um modelo matemático de computação que consiste em uma fita infinita dividida em células, um cabeçote de leitura/escrita que pode mover-se sobre a fita, um conjunto finito de estados, e uma função de transição que determina o comportamento da máquina. A cada passo, a máquina lê o símbolo da célula atual, escreve um novo símbolo (ou mantém o anterior), move o cabeçote para a esquerda ou direita, e muda de estado. Uma Máquina de Turing é *determinística* quando, para cada combinação de estado e símbolo lido, existe no máximo uma ação possível definida pela função de transição. Uma Máquina de Turing é *não determinística* quando podem existir múltiplas ações possíveis para uma mesma configuração, permitindo que a máquina "escolha"entre diferentes caminhos de computação. Esse modelo, proposto por Alan Turing em 1936, captura formalmente a noção intuitiva de algoritmo e constitui a base teórica para a definição de classes de complexidade computacional [5].

Um certificado para um problema de decisão é uma estrutura de dados que, quando fornecida junto com uma instância do problema, permite verificar em tempo polinomial se a resposta para aquela instância é “sim”. Formalmente, um problema L pertence à classe $\mathcal{NP}$ se existe um verificador polinomial V e uma constante c tal que, para toda instância x: $x \in L \iff \exists$ certificado y com $|y| \leq |x|^c$ tal que $V(x,y) =$ “aceita”. O certificado funciona como uma "prova"de que a instância tem resposta positiva. Por exemplo, para o problema Clique, um certificado seria um subconjunto específico de vértices; para um problema de satisfatibilidade booleana, seria uma valoração das variáveis. A existência de certificados verificáveis em tempo polinomial caracteriza a classe $\mathcal{NP}$ e distingue-a de outras classes de complexidade.

A classe de complexidade $\mathcal{P}$ consiste no conjunto de problemas decidíveis por uma Máquina de Turing determinística em tempo polinomial [5]. Formalmente,

$$\mathcal{P} = \{\, L \mid L \text{ é decidível por uma MT determinística em tempo polinomial} \,\}.$$

Por sua vez, a classe $\mathcal{NP}$ é definida de forma análoga, substituindo a Máquina de Turing determinística por uma não determinística:

$$\mathcal{NP} = \{\, L \mid L \text{ é decidível por uma MT não determinística em tempo polinomial} \,\}.$$

De modo equivalente, $\mathcal{NP}$ reúne os problemas cujas soluções podem ser verificadas em tempo polinomial mediante um certificado apropriado.

Um problema L é *$\mathcal{NP}$-difícil* quando todo problema em $\mathcal{NP}$ se reduz a L em tempo polinomial, e é *$\mathcal{NP}$-completo* quando, além disso, pertence à própria classe $\mathcal{NP}$.

Reduções polinomiais são o principal mecanismo para comparar a dificuldade de problemas. Diz-se que um problema de decisão A reduz-se polinomialmente a outro problema de decisão B, denotado pela notação $A \leq_p B$, quando existe uma função computável em tempo polinomial f tal que, para toda instância x, temos $x \in A \iff f(x) \in B$. O símbolo $\leq_p$ denota a relação de redução em tempo polinomial, indicando que o problema A não é mais difícil que o problema B do ponto de vista computacional. Nessa notação, A é denominado *problema atacado*, pois é o problema cuja complexidade já conhecemos, e B é denominado *problema alvo*, para o qual desejamos provar a complexidade. O uso dessas reduções permite demonstrar $\mathcal{NP}$-dificuldade e, quando combinado com a pertinência a $\mathcal{NP}$, também $\mathcal{NP}$-completude.

No contexto de reduções polinomiais, um *gadget* é uma construção auxiliar padronizada que traduz elementos estruturais do problema Alvo para o problema Atacado, preservando as propriedades essenciais da instância original. A técnica de construção por gadgets permite modularizar a redução, facilitando tanto a verificação de corretude quanto a análise de complexidade. Por exemplo, na redução Clique $\leq_p$ Max-2SAT apresentada neste trabalho, as cláusulas de seleção $(x_i \vee z)$ e $(x_i \vee \neg z)$ funcionam como gadgets que distinguem vértices selecionados de vértices não selecionados, enquanto as cláusulas de incompatibilidade $(\neg x_i \vee \neg x_j)$ atuam como gadgets que impedem a seleção simultânea de vértices não adjacentes.

No contexto de fórmulas booleanas, são definidos formalmente os conceitos fundamentais na ordem lógica de construção. Uma variável booleana é um símbolo que pode assumir um dentre dois valores possíveis, pertencentes ao conjunto $\{0,1\}$, onde 0 representa falso e 1 representa verdadeiro. Formalmente, dada uma variável x, uma valoração σ associa a x um valor em $\{0,1\}$, denotado por $\sigma(x)$. A partir de variáveis, construímos literais mediante a seguinte definição: um literal é uma variável x ou sua negação $\neg x$. Se $\sigma(x) = 1$, então o literal x é verdadeiro e o literal $\neg x$ é falso sob σ; de modo análogo, se $\sigma(x) = 0$, então o literal x é falso e o literal $\neg x$ é verdadeiro sob σ. Combinando literais mediante a operação lógica de disjunção, formamos cláusulas conforme a definição a seguir: uma cláusula é uma disjunção de literais, podendo ser representada formalmente como $C = (l_1 \vee l_2 \vee \cdots \vee l_r)$, onde cada l_i é um literal. Uma valoração σ satisfaz a cláusula C quando ao menos um de seus literais é verdadeiro sob σ, isto é, quando existe $i \in \{1, 2, \ldots, r\}$ tal que l_i é verdadeiro segundo σ. Finalmente, uma fórmula está em Forma Normal

TABELA 1: DEFINIÇÃO FORMAL DO PROBLEMA CLIQUE.

Clique
Entrada: Um grafo $G = (V(G), E(G))$ e um inteiro positivo k.
Questão: Existe um subconjunto $S \subseteq V(G)$ com $\|S\| \geq k$ tal que todo par de vértices de S é adjacente, isto é, para quaisquer $v_i, v_j \in S$ com $i \neq j$, tem-se $(v_i, v_j) \in E(G)$?

Clique: todos são amigos mútuos

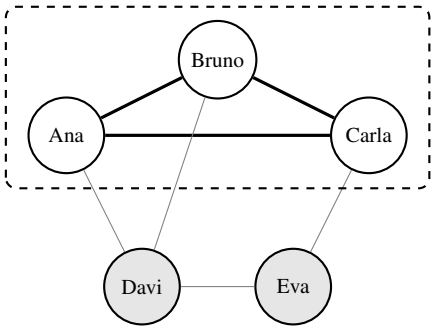


Figura 3: Rede social com clique {Ana, Bruno, Carla}.

Conjuntiva (FNC) quando é expressa como uma conjunção de cláusulas, isto é, $F = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, onde cada C_i é uma cláusula. Uma valoração σ satisfaz a fórmula F quando satisfaz simultaneamente todas as cláusulas C_i que compõem F.

A seguir são apresentados os dois problemas centrais desta redução: *Clique*, que atua como *problema atacado* ($\mathcal{NP}$-completo), e *Max-2SAT*, o *problema alvo* cuja $\mathcal{NP}$-dificuldade será estabelecida.

O problema Clique foi demonstrado como $\mathcal{NP}$-completo por Karp em 1972 [6]. Para tornar mais clara a estrutura do problema e facilitar a compreensão da redução apresentada posteriormente, é apresentado um exemplo cotidiano que ilustra o conceito de clique, conforme representado na Figura 3. Em uma rede social, cada pessoa é representada por um vértice do grafo, e uma aresta conecta duas pessoas que são amigas entre si. O problema Clique corresponde a encontrar um grupo de pelo menos k pessoas onde todos são mutuamente amigos, isto é, cada par de pessoas dentro desse grupo possui uma relação de amizade direta. Por exemplo, imagine que você deseja organizar um jantar e precisa convidar pelo menos três pessoas, mas com a restrição de que todos os convidados já se conheçam mutuamente para garantir um ambiente confortável e integrado. Determinar se tal grupo existe na sua rede de amizades é exatamente uma instância do problema Clique.

A Figura 3 ilustra esse cenário: Ana, Bruno e Carla formam uma clique de tamanho 3, pois cada par dentro desse grupo possui amizade direta (representada pelas arestas que conectam todos os três entre si). Davi e Eva, embora conectados a alguns membros do grupo, não fazem parte dessa clique pois não são amigos de todos os outros membros simultaneamente — por exemplo, Davi não possui aresta com Carla, e Eva não possui aresta com Ana nem com Bruno.

É importante observar a diferença fundamental entre o

TABELA 2: DEFINIÇÃO FORMAL DO PROBLEMA MAX-2SAT.

Max-2SAT
Entrada: Uma fórmula booleana F em forma normal conjuntiva (FNC), na qual cada cláusula contém no máximo dois literais, e um inteiro positivo k.
Questão: Existe uma valoração booleana das variáveis de F que satisfaça pelo menos k cláusulas?

problema 2-SAT e sua versão de maximização. O problema 2-SAT, que pergunta se existe uma valoração que satisfaz todas as cláusulas de uma fórmula onde cada cláusula tem no máximo dois literais, pode ser resolvido em tempo linear $O(n+m)$, onde n é o número de variáveis e m é o número de cláusulas, através de algoritmos baseados em grafos de implicação e componentes fortemente conexas [2]. Em contraste, a versão de maximização Max-2SAT foi demonstrada como $\mathcal{NP}$-completa por Garey, Johnson e Stockmeyer [3], que provou sua intratabilidade através de uma redução polinomial a partir do problema Vertex Cover. Essa diferença ilustra como alterações pequenas na especificação de um problema podem resultar em mudanças drásticas em sua complexidade computacional.

Demonstrar que Max-2SAT é $\mathcal{NP}$-completo requer exibir dois componentes: primeiro, um certificado verificável em tempo polinomial que comprove a pertinência à classe $\mathcal{NP}$; segundo, uma redução polinomial a partir de um problema já conhecido como $\mathcal{NP}$-completo. Neste trabalho, o problema Clique é utilizado como ponto de partida para a redução, estabelecendo uma correspondência entre estruturas altamente conectadas em grafos e fórmulas booleanas com alto grau de satisfatibilidade. Essa redução será desenvolvida em detalhes na Seção 5.

III. TRABALHOS RELACIONADOS

As referências clássicas sobre teoria da complexidade computacional e $\mathcal{NP}$-completude, como Cook [1], Karp [6], Garey e Johnson [4], Papadimitriou [7] e Sipser [5], constituem obras fundamentais para a compreensão geral de reduções polinomiais, classes de complexidade e taxonomia de problemas intratáveis, fornecendo o alicerce conceitual sobre o qual se desenvolvem estudos mais específicos.

No contexto específico de Max-2SAT e problemas de satisfatibilidade booleana, Goemans e Williamson [8] apresentam um algoritmo de aproximação baseado em programação semidefinida que alcança fator de aproximação de 0.878 para o problema Max-2SAT. O trabalho estabelece um marco importante ao conectar técnicas de otimização contínua com problemas combinatórios discretos, demonstrando que relaxações semidefinidas podem fornecer soluções de alta qualidade mesmo quando a solução ótima é computacionalmente intratável. Os autores utilizam arredondamento aleatório de variáveis baseado em vetores unitários, técnica que se tornou fundamental para o desenvolvimento de algoritmos de aproximação em problemas de satisfatibilidade.

Trevisan et al. [9] investigam a construção sistemática

ISSN: 2675-3588

de *gadgets* para reduções entre variantes de problemas de satisfação de restrições booleanas. O trabalho caracteriza quais propriedades estruturais devem ser preservadas ao transformar instâncias de um problema em outro, estabelecendo condições necessárias e suficientes para que uma redução mantenha a equivalência entre soluções ótimas. Os autores demonstram como gadgets bem projetados permitem controlar precisamente o número de cláusulas satisfeitas na fórmula resultante, técnica essencial para reduções que envolvem problemas de maximização como o Max-2SAT. Essa abordagem sistemática influenciou significativamente o desenvolvimento de novas reduções e a compreensão de limites de aproximabilidade.

Khanna et al. [10] estabelecem uma taxonomia completa de aproximabilidade para problemas de satisfação de restrições booleanas, incluindo Max-2SAT. O trabalho caracteriza formalmente quais variantes desses problemas admitem esquemas de aproximação em tempo polinomial (PTAS) e quais são $\mathcal{APX}$-completos, isto é, não admitem aproximação arbitrariamente boa sob a hipótese $\mathcal{P} \neq \mathcal{NP}$. Os autores demonstram que Max-2SAT pertence à classe $\mathcal{APX}$-completa, indicando que, embora existam algoritmos de aproximação com garantias constantes, não é possível obter esquemas que aproximem a solução ótima com erro arbitrariamente pequeno em tempo polinomial. Essa caracterização delimita precisamente as fronteiras entre o que é computacionalmente viável e o que permanece intratável mesmo sob relaxações de otimalidade.

No contexto pedagógico e didático, Lassance, Bianchini e Santos [11] apresentam um estudo fundamentado na experiência da disciplina de Teoria da Computação da Universidade Federal do Tocantins, evidenciando a importância de metodologias ativas baseadas em seminários acadêmicos para a aprendizagem de conceitos abstratos como decidibilidade, complexidade e $\mathcal{NP}$-completude. Os autores argumentam que a exposição pública, a análise crítica de demonstrações formais e a elaboração de apresentações estruturadas contribuem significativamente para o desenvolvimento de autonomia intelectual e domínio técnico por parte dos estudantes. A discussão mostra como abordagens dialogadas favorecem a consolidação de técnicas de redução polinomial e de formalização rigorosa, aspectos essenciais tanto para a compreensão de problemas intratáveis quanto para a construção de demonstrações corretas. Esse trabalho relaciona-se diretamente com a proposta pedagógica do presente artigo, que busca apresentar a demonstração de $\mathcal{NP}$-completude do Max-2SAT de forma didática e acessível a estudantes de graduação.

IV. Descrição do Problema

O problema *Maximum 2-Satisfiability* (Max-2SAT) é uma variante de maximização do problema clássico SAT, na qual cada cláusula contém no máximo dois literais. O objetivo é determinar uma valoração booleana que satisfaça o maior número possível de cláusulas. Na versão de decisão, investigada neste trabalho, pergunta-se se existe uma valoração capaz de satisfazer pelo menos k cláusulas de uma fórmula em forma normal conjuntiva (FNC).

Para tornar o problema Max-2SAT mais acessível, considere o seguinte cenário: um organizador de eventos precisa alocar n palestras em dois horários disponíveis, manhã e tarde. Cada palestra deve ocorrer em exatamente um dos dois períodos. Diversos pares de palestrantes expressam preferências conjuntas sobre os horários, representadas por restrições do tipo "pelo menos um de nós deve estar na manhã"ou "pelo menos um de nós deve estar na tarde".

Formalmente, cada palestra pode ser modelada i por uma variável booleana x_i, onde $x_i = 1$ significa que a palestra i está alocada no período da manhã, e $x_i = 0$ indica alocação no período da tarde. Uma preferência expressa por dois palestrantes i e j pode ser representada por uma cláusula booleana como $(x_i \vee x_j)$, que é satisfeita quando pelo menos uma das duas palestras ocorre na manhã, ou $(\neg x_i \vee x_j)$, indicando que se a palestra i for na manhã, então j também deve ser na manhã.

Em muitas situações práticas, as preferências dos palestrantes entram em conflito, tornando impossível satisfazer todas simultaneamente. Por exemplo, se três palestrantes A, B e C expressam as preferências "A ou B na manhã", "B ou C na tarde"e "A ou C em horários opostos", pode ser impossível atender todas ao mesmo tempo. Nesse contexto, o objetivo torna-se maximizar o número total de preferências atendidas, escolhendo uma alocação que satisfaça o maior número possível de restrições.

Esse cenário captura a essência do Max-2SAT: lidar com um sistema de restrições booleanas parcialmente conflitantes e buscar uma solução que maximize a consistência global, mesmo quando a satisfação total é inviável.

Formalmente, uma instância do Max-2SAT consiste em uma fórmula booleana F em forma normal conjuntiva (FNC), composta por cláusulas $C_1, C_2, \ldots, C_m$, cada uma contendo um ou dois literais, e por um inteiro positivo k. O problema consiste em determinar se existe uma valoração σ que satisfaça pelo menos k cláusulas de F, conforme definido na Tabela 2.

Embora o problema 2-SAT seja solucionável em tempo linear, sua versão de maximização (Max-2SAT) apresenta complexidade substancialmente maior. Essa diferença ilustra como pequenas alterações na formulação podem transformar um problema tratável em intratável. O Max-2SAT possui aplicações em gerenciamento de dependências de software, depuração de hardware, análise de redes biológicas e problemas de agendamento com restrições binárias.

Para ilustrar o comportamento do problema, considere a fórmula:

$$F = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_2 \vee \neg x_3).$$

Nenhuma valoração satisfaz simultaneamente as três cláusulas. Isso ocorre porque as duas últimas impõem condições opostas sobre a variável x_3: a cláusula $(\neg x_1 \vee x_3)$ força $x_3 = 1$ sempre que $x_1 = 1$, enquanto a cláusula $(\neg x_2 \vee \neg x_3)$ força $x_3 = 0$ sempre que $x_2 = 1$. Como a primeira cláusula $(x_1 \vee x_2)$ exige que pelo menos uma das duas variáveis seja verdadeira, inevitavelmente surge uma contradição. Se $x_1 = 1$, então x_3 deve ser 1, mas isso tende a violar a terceira cláusula. Se $x_2 = 1$, então x_3 deve ser 0, mas isso tende a violar a segunda cláusula. Assim, qualquer tentativa de satisfazer todas as três cláusulas força a violação de pelo menos uma delas.

Apesar disso, é possível satisfazer duas cláusulas. Por

TABELA 3: AVALIAÇÃO EXAUSTIVA DAS VALORAÇÕES PARA A FÓRMULA F.

x_1	x_2	C_1	C_2	C_3	C_4	Total
0	0	0	1	1	1	3
0	1	1	1	0	1	3
1	0	1	1	1	0	3
1	1	1	0	1	1	3

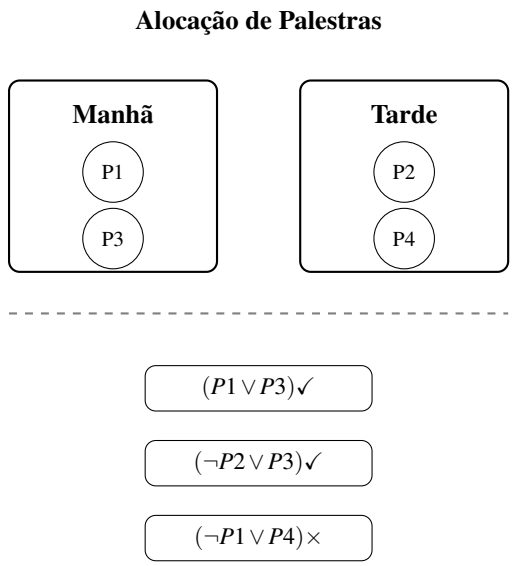


Figura 4: 2 de 3 preferências atendidas.

exemplo, a valoração $x_1 = 1$, $x_2 = 1$ e $x_3 = 0$ satisfaz a primeira e a terceira cláusulas, mas viola a segunda.

Esse comportamento evidencia o caráter de otimização do Max-2SAT: quando a estrutura das restrições contém conflitos inevitáveis, o objetivo deixa de ser satisfazer todas as cláusulas e passa a ser maximizar o número de cláusulas satisfeitas.

Para ilustrar de forma pedagógica situações em que não é possível satisfazer todas as cláusulas simultaneamente, considera-se a seguinte fórmula:

$$F = (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2).$$

Esta fórmula envolve duas variáveis booleanas, x_1 e x_2, de modo que existem exatamente quatro valorações possíveis. A Tabela 3 apresenta a avaliação sistemática de cada valoração, demonstrando que nenhuma satisfaz simultaneamente as quatro cláusulas.

Como evidenciado na tabela, cada valoração satisfaz exatamente três das quatro cláusulas, caracterizando um caso típico em que a formulação assume natureza de problema de maximização. Este exemplo evidencia a essência do Max-2SAT: quando a estrutura das restrições contém conflitos inevitáveis, o objetivo deixa de ser satisfazer todas as cláusulas e passa a ser maximizar o número de cláusulas satisfeitas.

Retornando à interpretação lúdica apresentada anteriormente, podemos visualizar o problema através do cenário de alocação de palestras. A Figura 4 ilustra de forma resumida a estrutura conceitual desse cenário, destacando os elementos centrais da formalização que será empregada na redução apresentada posteriormente.

Essa analogia capta a essência do Max-2SAT: resolver um sistema de restrições parcialmente conflitantes e maximizar sua consistência.

Do ponto de vista didático, o Max-2SAT é especialmente valioso por evidenciar de forma clara a diferença entre problemas de satisfação total e problemas de maximização. A análise desse tipo de fórmula permite ao estudante perceber como a impossibilidade de satisfazer todas as cláusulas conduz naturalmente a questões de otimização. O problema também mostra que a estrutura das cláusulas exerce influência direta sobre a complexidade computacional, deixando evidente que restrições simples como limitar cada cláusula a dois literais não garantem a existência de algoritmos polinomiais. Além disso, o Max-2SAT estabelece conexões importantes entre problemas booleanos e problemas em grafos, permitindo interpretar propriedades combinatórias por meio de fórmulas proposicionais. A construção de reduções por *gadgets*, como a utilizada na transformação Clique $\leq_p$ Max-2SAT apresentada na Seção 5, reforça técnicas fundamentais para demonstrações de $\mathcal{NP}$-completude. Uma compreensão precisa desse comportamento é essencial para acompanhar com rigor a prova apresentada.

V. DEMONSTRAÇÃO E CONTRIBUIÇÕES

Nesta seção é estabelecida a $\mathcal{NP}$-completude do problema Max-2SAT. A redução utilizada parte do problema Clique e emprega uma construção baseada em gadgets de seleção e incompatibilidade [4], detalhada passo a passo com a variável auxiliar z, as cláusulas de seleção e as cláusulas de exclusão para não-arestas.

Para provar que Max-2SAT é $\mathcal{NP}$-completo, é necessário demonstrar duas propriedades:

(i) Max-2SAT $\in \mathcal{NP}$;

(ii) existe um problema π sabidamente $\mathcal{NP}$-completo tal que $\pi \leq_p$ Max-2SAT.

Tome (I): Max-2SAT $\in \mathcal{NP}$. Para mostrar que Max-2SAT pertence à classe $\mathcal{NP}$, é suficiente exibir um certificado de tamanho polinomial e um algoritmo verificador que, dado esse certificado, decide em tempo polinomial se ele constitui uma solução válida para a instância. No caso do Max-2SAT, o certificado é uma valoração booleana $\sigma : \{x_1, x_2, \ldots, x_n\} \to \{0,1\}$ que atribui valores verdadeiro ou falso a todas as variáveis da fórmula F. Dado esse certificado, o Algoritmo 1 percorre cada cláusula da fórmula, avalia se ela é satisfeita pela valoração fornecida e conta o número total de cláusulas satisfeitas, verificando se esse total atinge pelo menos o limiar k especificado na instância.

Para verificar que Max-2SAT pertence à classe $\mathcal{NP}$, analisamos a complexidade do algoritmo verificador. O algoritmo percorre cada uma das m cláusulas uma única vez. Como cada cláusula contém no máximo dois literais, a avaliação de C_j sob σ é feita em tempo $O(1)$. Portanto, o tempo total de execução é $O(m)$, que é polinomial no tamanho da entrada. Logo, Max-2SAT $\in \mathcal{NP}$.

Tome (II): Max-2SAT é $\mathcal{NP}$-difícil via Clique $\leq_p$ Max-2SAT. Para demonstrar que Max-2SAT é $\mathcal{NP}$-difícil, seleciona-se o problema Clique, definido formalmente na Tabela 1 e conhecido por ser $\mathcal{NP}$-completo desde o trabalho

 ISSN: 2675-3588

Algorithm 1 Verificador Polinomial para Max-2SAT

Entrada: Fórmula $F = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ em 2-FNC, inteiro k, certificado σ
Saída: `ACEITA` se σ satisfaz pelo menos k cláusulas; `REJEITA` caso contrário

```
contador ← 0
for cada cláusula C_j ∈ F do
    Avalie C_j sob a valoração σ
    if C_j é satisfeita por σ then
        contador ← contador + 1
    end if
end for
if contador ≥ k then
    return ACEITA
else
    return REJEITA
end if
```

seminal de Karp [6], e constrói-se uma redução polinomial Clique $\leq_p$ Max-2SAT. Seguindo a terminologia adotada neste trabalho, Clique atua como *problema atacado* (o problema de partida, cuja $\mathcal{NP}$-completude já é conhecida) e Max-2SAT é o *problema alvo*, para o qual desejamos transferir a dificuldade computacional.

A escolha do problema Clique como ponto de partida para a redução é estratégica por diversas razões. Primeiro, a estrutura de Clique envolve a seleção de um subconjunto de vértices com propriedades específicas (adjacência mútua), o que mapeia naturalmente para variáveis booleanas indicando inclusão ou exclusão de elementos. Segundo, a condição de que todos os pares devem ser adjacentes traduz-se diretamente em cláusulas de incompatibilidade para pares não adjacentes. Terceiro, o parâmetro k (tamanho da clique) pode ser codificado no número de cláusulas satisfeitas, permitindo equivalência precisa entre os problemas. Por fim, a redução Clique $\leq_p$ Max-2SAT ilustra de forma didática a conexão entre problemas de grafos e problemas de lógica proposicional, tema central deste trabalho.

A estratégia geral consiste em criar uma variável booleana x_i para cada vértice, uma variável auxiliar z, e construir cláusulas que incentivam a seleção de vértices enquanto punem escolhas de pares não adjacentes. O parâmetro K' é ajustado para que a satisfação de exatamente K' cláusulas corresponda a uma clique de tamanho k.

A construção formal da fórmula procede da seguinte maneira. Dada uma instância (G,k) do problema *Clique*, onde $G = (V(G), E(G))$ é um grafo com conjunto de vértices $V(G)$ e conjunto de arestas $E(G)$, e k é um inteiro positivo representando o tamanho mínimo da clique procurada, a função de redução f produz uma instância (F', K') do problema *Max-2SAT*, transformando o grafo G em uma fórmula booleana F' em forma normal conjuntiva (onde cada cláusula contém no máximo dois literais) e o parâmetro k em um novo parâmetro K' que representa o número mínimo de cláusulas a serem satisfeitas. A seguir, descreve-se passo a passo como F' e K' são construídos a partir dos elementos de G e do valor k.

A construção de f pode ser realizada em tempo polinomial. A criação das variáveis booleanas requer $O(|V(G)|)$ operações para os n vértices, além de $O(1)$ para a variável auxiliar z. As cláusulas de seleção totalizam $2 \cdot |V(G)|$ cláusulas, cada uma construída em tempo constante, resultando em $O(|V(G)|)$. As cláusulas de incompatibilidade correspondem a uma cláusula para cada par de vértices não-adjacentes, o que no pior caso representa $|\overline{E}| = O(|V(G)|^2)$ cláusulas. Por fim, o cálculo de K' envolve apenas $O(1)$ operações aritméticas. Portanto, a complexidade total da redução é $O(|V(G)|^2)$, que é polinomial no tamanho da entrada.

A construção da fórmula F' envolve a criação de variáveis booleanas e três tipos de cláusulas que trabalham em conjunto para codificar a estrutura do grafo. Inicialmente, são definidas as variáveis que representarão os vértices do grafo. Para cada vértice $v_i \in V$, cria-se uma variável booleana x_i que indica se o vértice v_i faz parte da clique candidata. Além dessas variáveis, é introduzida uma variável auxiliar adicional z, cujo papel será explicado no contexto das cláusulas de seleção.

O primeiro tipo de cláusula são as cláusulas de seleção, que incentivam a escolha de vértices e permitem controlar o tamanho da clique. Para cada vértice v_i do grafo original, são inseridas duas cláusulas na fórmula: $(x_i \vee z)$ e $(x_i \vee \neg z)$. Essas cláusulas funcionam juntas para distinguir vértices selecionados de vértices não selecionados. Quando $x_i = 1$, indicando que o vértice v_i foi escolhido para compor a clique, ambas as cláusulas são satisfeitas, independentemente do valor atribuído à variável auxiliar z. Já quando $x_i = 0$, apenas uma das duas cláusulas pode ser satisfeita, dependendo do valor de z: se $z = 1$, a cláusula $(x_i \vee z)$ é satisfeita e $(x_i \vee \neg z)$ é violada; se $z = 0$, ocorre o inverso. Essa diferença de uma cláusula satisfeita entre vértices escolhidos e não escolhidos permite controlar o tamanho da clique por meio do parâmetro K', garantindo que apenas seleções com exatamente k vértices produzam o número exigido de cláusulas satisfeitas.

O segundo tipo de cláusula são as cláusulas de incompatibilidade, que garantem que apenas vértices mutuamente adjacentes sejam selecionados simultaneamente. Para cada par de vértices (v_i, v_j) que não são adjacentes no grafo original, isto é, para cada par onde $(v_i, v_j) \notin E$, adiciona-se à fórmula a cláusula $(\neg x_i \vee \neg x_j)$. Essa cláusula impõe uma restrição essencial: vértices não adjacentes não podem ser selecionados ao mesmo tempo para compor a clique. Se ambos x_i e x_j recebem o valor 1, a cláusula $(\neg x_i \vee \neg x_j)$ se torna falsa, penalizando essa escolha inválida. Por outro lado, se ao menos um dos vértices não for selecionado (isto é, se uma das variáveis for 0), a cláusula é satisfeita. Dessa forma, qualquer valoração que satisfaça um número elevado de cláusulas deve corresponder a um conjunto de vértices que forma uma clique no grafo original.

Finalmente, define-se o parâmetro K' que estabelece o número mínimo de cláusulas a serem satisfeitas. O parâmetro K', que estabelece o número mínimo de cláusulas a serem satisfeitas na instância de Max-2SAT, é definido de modo a manter equivalência exata com o problema Clique. O parâmetro é definido como $K' = |V| + k + |\overline{E}|$, onde $|V(G)|$ é o número total de vértices do grafo (e, portanto, o número de cláusulas do tipo $(x_i \vee z)$), k é o tamanho da clique procurada no problema original (refletindo o ganho adicional obtido nas cláusulas $(x_i \vee \neg z)$ quando k vértices são

TABELA 4: RESUMO DA REDUÇÃO CLIQUE $\leq_p$ MAX-2SAT.

Componente	Cláusulas	Função (Gadget)	Contrib. p/ K'
Var. de vértice	x_i, $v_i \in V$	Indica se v_i pertence à clique	—
Var. auxiliar	z	Controla contagem de cláusulas	—
Seleção positiva	$(x_i \vee z)$	Satisfeita se z=1; base fixa	$\lvert V\rvert$
Seleção negativa	$(x_i \vee \neg z)$	Satisfeita se x_i=1; mede seleção	k
Incompatib.	$(\neg x_i \vee \neg x_j)$, $(v_i, v_j) \notin E$	Impede seleção de não adjacentes	$\lvert\overline{E}\rvert$
		Total:	$\lvert V\rvert+k+\lvert\overline{E}\rvert$

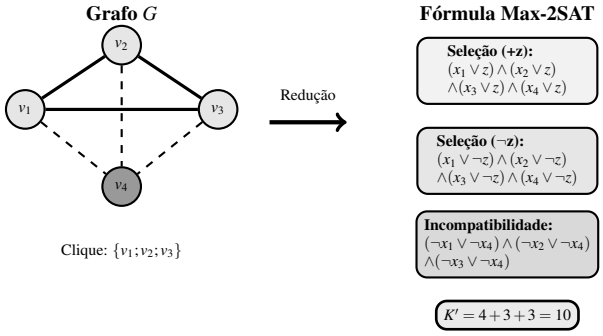


Figura 5: Transformação completa: grafo com clique $\{v_1, v_2, v_3\}$ e fórmula Max-2SAT resultante.

escolhidos), e $|\overline{E}|$ é o número de não-arestas, isto é, de pares de vértices não adjacentes (que correspondem às cláusulas $(\neg x_i \vee \neg x_j)$). Essa definição garante que a solução do Max-2SAT reproduza a condição do problema da clique.

Para facilitar a compreensão da construção, a Tabela 4 consolida os componentes da redução, explicitando o papel de cada tipo de cláusula, o significado dos gadgets empregados e a contribuição de cada bloco para o parâmetro K'.

A Figura 5 ilustra o processo de redução em três etapas, mostrando como cada elemento do grafo original é traduzido para componentes da fórmula Max-2SAT. Na primeira etapa, apresentamos o grafo de entrada com seus vértices e arestas. Na segunda etapa, mostramos os gadgets de seleção, que consistem nas cláusulas $(x_i \vee z)$ e $(x_i \vee \neg z)$ para cada vértice, representando o mecanismo que diferencia vértices escolhidos de não escolhidos. Na terceira etapa, exibimos os gadgets de incompatibilidade, que são as cláusulas $(\neg x_i \vee \neg x_j)$ geradas para cada par de vértices não adjacentes, impedindo a seleção simultânea de vértices que não formam aresta.

Para entender como esse valor opera, considere uma valoração que corresponde a uma clique válida de tamanho k. Todas as $|V(G)|$ cláusulas do tipo $(x_i \vee z)$ são satisfeitas, contribuindo $|V(G)|$ para a contagem. As k cláusulas $(x_i \vee \neg z)$ associadas aos vértices escolhidos também são satisfeitas, acrescentando k ao total. Além disso, todas as $|\overline{E}|$ cláusulas de incompatibilidade são satisfeitas, pois nenhum par de vértices não adjacentes foi selecionado ao mesmo tempo. Somando essas contribuições, obtém-se $K' = |V| + k + |\overline{E}|$. Já qualquer valoração que não represente uma clique de tamanho pelo menos k ou que viole alguma cláusula de incompatibilidade satisfará um número menor de cláusulas, o que garante a equivalência entre os dois problemas.

A Figura 5 apresenta uma visualização completa da transformação, mostrando lado a lado o grafo de entrada e a fórmula Max-2SAT resultante, com destaque para a correspondência entre cada elemento do grafo e as cláusulas geradas. Na parte esquerda da figura, observa-se o grafo G com a clique $\{v_1, v_2, v_3\}$ destacada, onde as linhas contínuas representam arestas e as linhas tracejadas representam não-arestas. Na parte direita, é exibida a fórmula completa organizada em três blocos: as cláusulas de seleção positivas $(x_i \vee z)$, as cláusulas de seleção negativas $(x_i \vee \neg z)$, e as cláusulas de incompatibilidade $(\neg x_i \vee \neg x_j)$. A organização visual evidencia como cada vértice do grafo origina suas cláusulas de seleção e como cada par de vértices não adjacentes (linhas tracejadas) gera sua respectiva cláusula de incompatibilidade.

Para ilustrar como o parâmetro K' é obtido, considere o grafo mostrado na Figura 5, com $V(G) = \{v_1; v_2; v_3; v_4\}$ e $E(G) = \{(v_1, v_2); (v_2, v_3); (v_1, v_3)\}$. Suponha que desejamos verificar se existe uma clique de tamanho $k = 3$ nesse grafo.

Primeiro, são identificados os componentes necessários para o cálculo:

O número de vértices é $|V(G)| = 4$, o que gera quatro cláusulas do tipo $(x_i \vee z)$, uma para cada vértice do grafo. Essas cláusulas sempre serão satisfeitas quando $z = 1$, independentemente de quais vértices forem escolhidos.

O tamanho da clique desejada é $k = 3$, que corresponde ao número de cláusulas adicionais do tipo $(x_i \vee \neg z)$ que esperamos satisfazer quando o número de vértices selecionados é três.

Para determinar o número de não-arestas $|\overline{E}|$, conta-se quantos pares de vértices não são adjacentes. Em um grafo com quatro vértices, existem $\binom{4}{2} = 6$ pares possíveis. Como o grafo possui $|E(G)| = 3$ arestas, o número de não-arestas é $|\overline{E}| = 6 - 3 = 3$. Os pares não adjacentes são: (v_1, v_4), (v_2, v_4) e (v_3, v_4). Para cada um desses pares, inclui-se uma cláusula de incompatibilidade.

Aplicando a fórmula $K' = |V(G)| + k + |\overline{E}|$, obtém-se:

$$K' = 4 + 3 + 3 = 10$$

Assim, a instância de Max-2SAT correspondente pergunta: "É possível satisfazer pelo menos 10 cláusulas da fórmula construída?"Uma resposta positiva equivale a afirmar que o grafo original possui uma clique de tamanho pelo menos 3.

Para verificar a construção, observe que o grafo contém uma clique de tamanho 3 formada pelos vértices $\{v_1, v_2, v_3\}$. Atribuindo $x_1 = x_2 = x_3 = 1$, $x_4 = 0$ e $z = 1$, temos:

As 4 cláusulas $(x_1 \vee z)$, $(x_2 \vee z)$, $(x_3 \vee z)$ e $(x_4 \vee z)$ são todas satisfeitas porque $z = 1$, contribuindo com 4 cláusulas.

Das 4 cláusulas do tipo $(x_i \vee \neg z)$, apenas as três correspondentes aos vértices selecionados são satisfeitas: $(x_1 \vee \neg z)$, $(x_2 \vee \neg z)$ e $(x_3 \vee \neg z)$, pois $x_1 = x_2 = x_3 = 1$. Isso contribui com 3 cláusulas adicionais.

As 3 cláusulas de incompatibilidade $(\neg x_1 \vee \neg x_4)$, $(\neg x_2 \vee \neg x_4)$ e $(\neg x_3 \vee \neg x_4)$ são todas satisfeitas porque $x_4 = 0$, contribuindo com 3 cláusulas.

O total é $4 + 3 + 3 = 10 = K'$, o que confirma que a valoração satisfaz exatamente o número exigido de cláusulas. Esse exemplo mostra que o parâmetro K' reflete de forma precisa a estrutura da clique por meio da contagem de cláusulas satisfeitas.

ISSN: 2675-3588

Estabelecida a construção, é necessário agora provar sua corretude demonstrando a equivalência entre as instâncias. Para estabelecer que a redução está correta, é preciso demonstrar que $(G,k) \in$ Clique se e somente se $(F',K') \in$ Max-2SAT. Cada direção é provada separadamente.

Ida ($\Rightarrow$): Se G possui uma clique de tamanho k, então (F', K') é satisfatível. Suponha que exista um conjunto $S \subseteq V(G)$ de vértices formando uma clique de tamanho pelo menos k, isto é, $|S| \geq k$ e para todo par de vértices distintos $v_i, v_j \in S$, existe uma aresta $(v_i, v_j) \in E(G)$. Constrói-se uma valoração que satisfaz pelo menos K' cláusulas da fórmula F'.

Defina a valoração da seguinte forma: para cada variável x_i, atribua $x_i = 1$ se o vértice v_i pertence ao conjunto S e $x_i = 0$ caso contrário. Adicionalmente, atribua $z = 1$ à variável auxiliar. Analisa-se quantas cláusulas são satisfeitas por essa valoração.

Primeiro, são analisadas as cláusulas de seleção. Para cada vértice $v_i \in V(G)$, há duas cláusulas: $(x_i \vee z)$ e $(x_i \vee \neg z)$. Como foi definido $z = 1$, todas as cláusulas do tipo $(x_i \vee z)$ são satisfeitas, independentemente do valor de x_i, o que contribui com $|V(G)|$ cláusulas satisfeitas. Já para as cláusulas $(x_i \vee \neg z)$, temos $\neg z = 0$ nessa valoração, de modo que elas são satisfeitas apenas quando $x_i = 1$. Como foi atribuído $x_i = 1$ aos k vértices do conjunto S, k cláusulas desse tipo são satisfeitas.

Agora são consideradas as cláusulas de incompatibilidade. Para cada par não-adjacente $(v_i, v_j) \notin E(G)$, temos a cláusula $(\neg x_i \vee \neg x_j)$. Essa cláusula é falsa apenas quando ambos $x_i = 1$ e $x_j = 1$, o que aconteceria se ambos os vértices v_i e v_j pertencessem ao conjunto S. No entanto, por hipótese, S é uma clique, portanto todos os pares de vértices em S são adjacentes. Isso significa que não existe nenhum par $(v_i, v_j) \notin E(G)$ com ambos $v_i, v_j \in S$. Consequentemente, para todo par não-adjacente $(v_i, v_j) \notin E(G)$, pelo menos um dos vértices não pertence a S, garantindo que pelo menos uma das variáveis x_i ou x_j vale 0, o que torna a cláusula $(\neg x_i \vee \neg x_j)$ verdadeira. Portanto, todas as $|\overline{E}|$ cláusulas de incompatibilidade são satisfeitas.

Somando as contribuições, obtém-se $|V(G)| + k + |\overline{E}| = K'$ cláusulas satisfeitas, provando que $(F',K') \in$ Max-2SAT.

Volta ($\Leftarrow$): Se (F',K') é satisfatível, então G possui uma clique de tamanho k. Suponha agora que exista uma valoração que satisfaz pelo menos K' cláusulas da fórmula F'. É demonstrado que pode-se extrair dessa valoração um conjunto de vértices que forma uma clique de tamanho pelo menos k no grafo G.

Sem perda de generalidade, pode-se assumir que a variável auxiliar z recebe o valor 1 nesta valoração. Caso $z = 0$ na valoração original, considera-se uma valoração alternativa onde invertemos o valor de z para 1 e mantemos os valores de todas as variáveis x_i inalterados. Como cada vértice contribui com duas cláusulas $(x_i \vee z)$ e $(x_i \vee \neg z)$, inverter z apenas troca qual dessas cláusulas é satisfeita para vértices com $x_i = 0$, sem alterar o total de cláusulas satisfeitas. Portanto, é possível trabalhar com uma valoração onde $z = 1$.

Com $z = 1$, todas as $|V(G)|$ cláusulas da forma $(x_i \vee z)$ são automaticamente satisfeitas. Isso deixa espaço para $K' - |V(G)| = k + |\overline{E}|$ cláusulas adicionais serem satisfeitas. Essas cláusulas adicionais vêm de duas fontes: as cláusulas $(x_i \vee \neg z)$ para vértices com $x_i = 1$ e as cláusulas de incompatibilidade $(\neg x_i \vee \neg x_j)$.

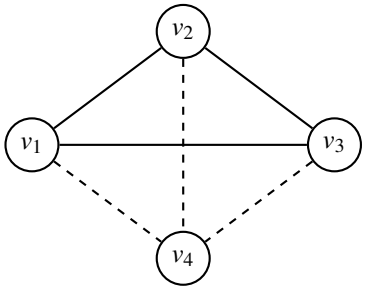


Figura 6: Grafo de entrada para a redução: clique $\{v_1, v_2, v_3\}$ e vértice isolado v_4.

Defina $S = \{v_i \mid x_i = 1\}$ como o conjunto de vértices cujas variáveis foram atribuídas como verdadeiras. Para satisfazer pelo menos $k + |\overline{E}|$ cláusulas adicionais, é necessário ter pelo menos $|S|$ cláusulas do tipo $(x_i \vee \neg z)$ satisfeitas (uma para cada vértice em S) e todas as $|\overline{E}|$ cláusulas de incompatibilidade satisfeitas.

Se alguma cláusula de incompatibilidade $(\neg x_i \vee \neg x_j)$ não for satisfeita, isso significa que ambos $x_i = 1$ e $x_j = 1$, mas $(v_i, v_j) \notin E(G)$. Cada cláusula de incompatibilidade violada reduz o número total de cláusulas satisfeitas em uma unidade. Para manter o total em pelo menos K', seria necessário que mais cláusulas $(x_i \vee \neg z)$ fossem satisfeitas, o que requereria mais vértices em S. No entanto, adicionar mais vértices aumenta o risco de violar mais cláusulas de incompatibilidade. De fato, pode-se verificar algebricamente que violar qualquer cláusula de incompatibilidade torna impossível atingir exatamente K' cláusulas satisfeitas com a construção apresentada.

Portanto, todas as $|\overline{E}|$ cláusulas de incompatibilidade devem ser satisfeitas, o que garante que não existe nenhum par $(v_i, v_j) \notin E(G)$ com ambos $v_i, v_j \in S$. Logo, S forma uma clique de tamanho pelo menos k no grafo G.

Para consolidar a compreensão da redução, é apresentado um exemplo completo que percorre todas as etapas da transformação, desde o grafo de entrada até a verificação da valoração resultante. Considere um grafo com quatro vértices $V(G) = \{v_1, v_2, v_3, v_4\}$ e arestas $E(G) = \{(v_1, v_2), (v_2, v_3), (v_1, v_3)\}$. A Figura 6 ilustra esse grafo, onde os vértices v_1, v_2 e v_3 formam uma clique de tamanho 3, representada pelas linhas contínuas que conectam cada par desses três vértices. O vértice v_4 não possui arestas com nenhum dos outros vértices, o que é indicado pelas linhas tracejadas que representam os pares não adjacentes (v_1, v_4); (v_2, v_4); (v_3, v_4). Essa distinção visual entre arestas presentes e ausentes é fundamental para compreender como a redução constrói as cláusulas de incompatibilidade.

A partir da clique de tamanho 3 identificada no grafo, a construção gera as cláusulas:

$$(x_i \vee z), \quad (x_i \vee \neg z) \quad \text{para } i = 1,2,3,4,$$

e, para as não-arestas:

$$(\neg x_1 \vee \neg x_4), \quad (\neg x_2 \vee \neg x_4), \quad (\neg x_3 \vee \neg x_4).$$

A valoração $x_1 = x_2 = x_3 = 1$, $x_4 = 0$ e $z = 1$ satisfaz exatamente K' cláusulas, como requerido. A Figura 7 ilustra detalhadamente essa verificação, mostrando quais cláusulas

Verificação: $x_1 = x_2 = x_3 = 1, x_4 = 0, z = 1$

Seleção $(x_i \vee z)$**: 4 cláusulas**

Seleção $(x_i \vee \neg z)$**: 3 cláusulas**

Incompatibilidade: 3 cláusulas

Total: $4+3+3 = 10 = K'$ ✓

Figura 7: Verificação detalhada da valoração que satisfaz $K' = 10$ cláusulas.

são satisfeitas e quais não são, organizadas por tipo. Na coluna da esquerda, observa-se que todas as cláusulas de seleção positiva $(x_i \vee z)$ são satisfeitas porque $z = 1$. Na coluna central, as cláusulas de seleção negativa $(x_i \vee \neg z)$ são satisfeitas apenas para os vértices selecionados (x_1, x_2, x_3), totalizando 3 cláusulas, enquanto a cláusula $(x_4 \vee \neg z)$ não é satisfeita pois $x_4 = 0$ e $\neg z = 0$. Na coluna da direita, todas as cláusulas de incompatibilidade são satisfeitas porque $x_4 = 0$ torna verdadeira qualquer cláusula da forma $(\neg x_i \vee \neg x_4)$.

Esse exemplo demonstra concretamente como a estrutura da clique no grafo original é preservada na fórmula Max-2SAT: os três vértices da clique correspondem aos três vértices com $x_i = 1$, que são exatamente os responsáveis por satisfazer as cláusulas extras de seleção negativa. Simultaneamente, o fato de esses três vértices serem mutuamente adjacentes garante que nenhuma cláusula de incompatibilidade é gerada entre eles, permitindo que todas as cláusulas de incompatibilidade (que envolvem apenas v_4) sejam satisfeitas.

A redução com a variável auxiliar z evidencia diversos aspectos pedagógicos relevantes. Ela mostra como cláusulas de dois literais podem impor restrições estruturais fortes sobre as possíveis valorações e como a seleção de vértices válidos depende da satisfação simultânea de múltiplas cláusulas independentes. Além disso, as cláusulas negativas traduzem de maneira direta as relações de incompatibilidade no grafo, reforçando a conexão entre propriedades combinatórias e expressões booleanas. A definição precisa do parâmetro K' demonstra como controlar o tamanho da clique desejada por meio da contagem de cláusulas satisfeitas. Por fim, essa construção evidencia por que reduções entre problemas de grafos e fórmulas booleanas constituem ferramenta fundamental no estudo de $\mathcal{NP}$-completude.

VI. Resultados e Reflexões

Apresentada a demonstração de $\mathcal{NP}$-completude do Max-2SAT, esta seção discute os principais resultados obtidos, bem como reflexões conceituais e pedagógicas sobre o processo de construção da redução e sobre os elementos que tornaram essa abordagem útil para o aprendizado em Teoria da Computação.

A demonstração apresentada confirmou que o problema Max-2SAT pertence à classe $\mathcal{NP}$, uma vez que o número de cláusulas satisfeitas por uma valoração pode ser verificado em tempo linear no tamanho da instância. Basta percorrer cada cláusula uma única vez, avaliar se ela é verdadeira sob a valoração fornecida como certificado, e contar quantas são satisfeitas. Como cada cláusula contém no máximo dois literais, essa avaliação é realizada em tempo constante por cláusula, resultando em complexidade total $O(m)$, onde m é o número de cláusulas.

Além disso, foi demonstrado que Max-2SAT é $\mathcal{NP}$-difícil, pois o problema Clique, que é $\mathcal{NP}$-completo conforme estabelecido por Karp [6], foi reduzido a ele por meio de uma função de transformação computável em tempo polinomial. A redução constrói uma fórmula booleana cujo tamanho é polinomial no tamanho do grafo de entrada: o número de variáveis é $|V| + 1$ e o número de cláusulas é $2|V| + |\overline{E}|$, onde $|\overline{E}| \leq \binom{|V|}{2}$. A construção de cada cláusula requer tempo constante, portanto a transformação completa opera em tempo $O(|V|^2)$.

A formulação com a variável auxiliar z mostrou-se útil, pois permite controlar o número total de cláusulas satisfeitas sem recorrer a construções mais extensas. Essa técnica evita a criação de cláusulas com mais de dois literais e preserva a estrutura típica do Max-2SAT. A variável z funciona como um mecanismo de balanceamento que garante que vértices escolhidos contribuam com exatamente uma cláusula a mais do que vértices não escolhidos, traduzindo o tamanho da clique diretamente na quantidade de cláusulas satisfeitas.

O valor limite K' foi definido para refletir com precisão a estrutura combinatória do grafo original. A decomposição $K' = |V| + k + |\overline{E}|$ incorpora três componentes distintos: a base fixa de cláusulas sempre satisfeitas, o ganho proporcional ao tamanho da clique e a penalização associada à violação das cláusulas de incompatibilidade. Essa construção assegura que uma valoração que satisfaça exatamente K' cláusulas corresponda a uma clique de tamanho k, estabelecendo equivalência completa entre os dois problemas.

Esses resultados mostram que pequenas alterações estruturais em problemas que parecem simples, como a transição de 2-SAT para Max-2SAT, podem alterar de forma profunda sua complexidade. Enquanto 2-SAT admite solução em tempo linear por meio do grafo de implicações, sua versão de maximização torna-se tão difícil quanto qualquer problema em NP, evidenciando a fronteira entre tratabilidade e intratabilidade.

O desenvolvimento da redução Clique $\leq_p$ Max-2SAT revelou diversos aspectos importantes para o ensino e compreensão de problemas $\mathcal{NP}$-completos. A transformação de relações de adjacência em cláusulas de dois literais torna explícito como propriedades estruturais de grafos podem ser modeladas por fórmulas booleanas. Cada aresta ou não-aresta no grafo corresponde de forma direta a uma restrição lógica, estabelecendo um dicionário claro entre os dois domínios. Essa correspondência evidencia que problemas aparentemente distintos compartilham estrutura matemática profunda, sendo manifestações diferentes de uma mesma dificuldade computacional subjacente.

A expressão $(\neg x_i \vee \neg x_j)$ traduz a proibição de escolher dois vértices não adjacentes, mostrando como restrições combinatórias são mapeadas para restrições lógicas. Essa cláusula funciona como uma "barreira lógica"que impede configurações inválidas, e sua violação resulta na redução do número total de cláusulas satisfeitas. Compreender esse mecanismo de penalização é fundamental para desenvolver

 ISSN: 2675-3588

intuição sobre como problemas de otimização combinatória podem ser codificados em fórmulas booleanas.

A introdução da variável auxiliar z simplifica a contagem de cláusulas, garantindo equilíbrio entre as expressões do tipo $(x_i \vee z)$ e $(x_i \vee \neg z)$ e permitindo a definição precisa do parâmetro K'. Sem essa variável, seria necessário construir gadgets mais complexos ou usar cláusulas maiores, evidenciando a fronteira. A técnica de usar variáveis auxiliares para controlar comportamentos globais da fórmula é aplicável em construções de reduções, e sua apresentação neste contexto fornece modelo útil para outros problemas.

O ajuste de K' evidencia uma técnica comum em reduções, na qual o número de cláusulas satisfeitas reflete diretamente o tamanho da estrutura procurada no problema original. Essa correspondência numérica precisa entre parâmetros do problema fonte e problema alvo é característica essencial de reduções bem construídas. Estudantes muitas vezes têm dificuldade em determinar parâmetros corretos para problemas de otimização; o exemplo apresentado demonstra metodologia sistemática baseada na análise de contribuições independentes de cada componente da construção.

Provar ambas as direções da equivalência, isto é, que uma clique gera uma valoração válida e que uma valoração válida gera uma clique, reforça o raciocínio formal necessário para reduções corretas. Muitos estudantes cometem o erro de provar apenas uma direção ou de assumir equivalência sem justificativa rigorosa. A demonstração cuidadosa apresentada serve como modelo de argumentação matemática, enfatizando a importância de considerar todas as possibilidades e eliminar casos degenerados.

Analisar como cada valoração influencia o conjunto das cláusulas satisfeitas ajuda a desenvolver intuição sobre como problemas booleanos capturam propriedades de grafos. Ao considerar sistematicamente os efeitos de atribuir valores verdadeiro ou falso a cada variável, estudantes desenvolvem compreensão profunda de como restrições locais (satisfação de cláusulas individuais) emergem como propriedades globais (existência de cliques). Essa conexão entre nível local e global é fundamental não apenas em complexidade computacional, mas em toda ciência da computação.

Ao final, a redução analisada oferece uma visão sólida sobre a intratabilidade do Max-2SAT e sobre a versatilidade das reduções polinomiais. A construção estudada demonstra, de forma acessível e estruturada, como problemas de natureza combinatória e lógica podem ser relacionados de maneira precisa, contribuindo significativamente para o aprendizado prático de $\mathcal{NP}$-completude.

+3O material desenvolvido pode ser empregado em diversos contextos de ensino. Em disciplinas de *graduação* em Teoria da Computação ou Análise de Algoritmos, a redução pode ser apresentada como estudo de caso após a introdução dos conceitos de $\mathcal{NP}$-completude, permitindo que estudantes acompanhem passo a passo uma demonstração completa antes de desenvolverem suas próprias provas. Em cursos de *pós-graduação*, o material pode servir como ponto de partida para discussões sobre técnicas avançadas de redução, limites de aproximabilidade e conexões com outros problemas de satisfatibilidade. Como *atividade prática*, sugere-se propor aos estudantes a verificação manual da redução para grafos pequenos, a implementação computacional do algoritmo de transformação, ou a adaptação da técnica para variantes como Max-3SAT ou Weighted Max-2SAT. A estrutura modular da apresentação — com figuras, tabelas-resumo e exemplos comentados — facilita a segmentação do conteúdo em múltiplas aulas ou a utilização em metodologias ativas baseadas em seminários, conforme discutido por Lassance, Bianchini e Santos [11].

O material apresentado serve tanto como recurso didático para compreensão de técnicas específicas quanto como exemplar metodológico para desenvolvimento de novas reduções, cumprindo assim o objetivo pedagógico central deste trabalho.

VII. Considerações Finais

Este artigo apresentou uma demonstração formal e didática da $\mathcal{NP}$-completude do problema Max-2SAT por meio de redução polinomial a partir do problema Clique. A prova foi estruturada demonstrando-se primeiro a pertinência à classe $\mathcal{NP}$ mediante certificado verificável em tempo polinomial, e em seguida a $\mathcal{NP}$-dificuldade através de transformação polinomial que preserva equivalência entre instâncias.

A construção proposta utiliza variável auxiliar z para controlar o número de cláusulas satisfeitas, ajustando o parâmetro K' para refletir de forma precisa o tamanho da clique desejada. As cláusulas de seleção incentivam a escolha de vértices, enquanto as cláusulas de incompatibilidade impedem seleção simultânea de vértices não adjacentes, evidenciando como relações combinatórias em grafos são codificadas por fórmulas booleanas.

Entre as dificuldades encontradas, destacam-se a escolha adequada do parâmetro K' e a formalização rigorosa da prova de corretude em ambas as direções. A compreensão do papel da variável z exigiu análise cuidadosa de como cada tipo de cláusula contribui para a contagem total.

Como limitações, o trabalho concentrou-se na versão de decisão do Max-2SAT e na redução a partir de Clique. Outras reduções, como baseadas em 3-SAT ou Vertex Cover, podem oferecer perspectivas complementares. Para trabalhos futuros, sugere-se explorar variantes parametrizadas, analisar complexidade em classes especiais de grafos, desenvolver algoritmos aproximativos e investigar aplicações com SAT solvers modernos.

O material produzido serve como recurso didático para disciplinas de Teoria da Computação, contribuindo para a formação de estudantes em Ciência da Computação. Espera-se que inspire novas produções que articulem rigor técnico e finalidade pedagógica, fortalecendo a comunidade de ensino e pesquisa em Teoria da Computação.

Referências

[1] S. A. Cook, "The complexity of theorem-proving procedures," *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pp. 151–158, 1971.

[2] B. Aspvall, M. F. Plass, and R. E. Tarjan, "A linear-time algorithm for testing the truth of certain quantified boolean formulas," *Information Processing Letters*, vol. 8, no. 3, pp. 121–123, 1979.

[3] M. R. Garey, D. S. Johnson, and L. Stockmeyer, "Some simplified np-complete graph problems," *Theoretical Computer Science*, vol. 1, no. 3, pp. 237–267, 1976.

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

[5] M. Sipser, *Introduction to the Theory of Computation*, 2nd ed. Boston: Thomson Course Technology, 2006.

[6] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, Eds. New York, NY; London: Plenum Press, 1972, pp. 85–103.

[7] C. H. Papadimitriou, *Computational Complexity*. Addison-Wesley, 1994.

[8] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *Journal of the ACM*, vol. 42, no. 6, pp. 1115–1145, 1995.

[9] L. Trevisan, G. B. Sorkin, M. Sudan, and D. P. Williamson, "Gadgets, approximation, and linear programming," in *Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS)*, 1996, pp. 617–626. [Online]. Available: https://ieeexplore.ieee.org/document/548521

[10] S. Khanna, M. Sudan, L. Trevisan, and D. P. Williamson, "The approximability of constraint satisfaction problems," *SIAM Journal on Computing*, vol. 30, no. 6, pp. 1863–1920, 2001. [Online]. Available: https://doi.org/10.1137/S0097539799349948

[11] Y. M. Lassance Di Vilhena Y Cantañede, G. d. B. Bianchini, and T. D. d. Santos, "Reflexões e práticas pedagógicas no escopo da disciplina de teoria da computação," *Academic Journal on Computing, Engineering and Applied Mathematics*, vol. 6, no. 2, pp. 10–17, October 2025.

 ISSN: 2675-3588

SET PACKING: Contribuições Pedagógicas para o Aprendizado no Escopo da Teoria da Computação

SET PACKING: Pedagogical Contributions for Learning within the Scope of the Theory of Computation

Emanuel B. Fonseca[1], Victhor C. Magalhães[1], Daniel Martins da Silva[2] e Tanilson D. dos Santos[1]

[1] *Universidade Federal do Tocantins, Ciência da Computação, Palmas, Brasil*
[2] *Universidade Federal do Norte do Tocantins, Araguaína, Brasil*

Data de recebimento do manuscrito: 30/11/2025
Data de aceitação do manuscrito: 30/01/2026
Data de publicação: 10/02/2026

Resumo—Este trabalho propõe uma abordagem metodológica e didática para a reprodução e elucidação dos conceitos de NP-Completude e da intratabilidade inerente a problemas computacionais, especialmente os combinatórios, utilizando o *Set Packing Problem* (SP) como estudo de caso. A metodologia consiste na exposição detalhada do problema, seguida pela reprodução da demonstração formal do pertencimento do SP à classe NP-Completa, incluindo a construção e análise de seu verificador polinomial e a apresentação passo a passo da técnica de redução polinomial de *CLIQUE* para *SP*. O principal resultado e a contribuição deste artigo é o desenvolvimento de um material pedagógico que visa aprimorar a compreensão integral dos conceitos da Teoria da Computação, auxiliando o público leigo a assimilar de forma eficaz o significado da complexidade computacional.

Palavras-chave—Teoria da Computação, Complexidade Computacional, NP-Completude, Empacotamento de Conjuntos, Redução Polinomial

Abstract—*This work proposes a methodological and didactic approach for the reproduction and elucidation of the concepts of NP-Completeness and the inherent intractability of combinatorial problems, using the Set Packing Problem (SP) as a case study. The methodology consists of a detailed exposition of the problem, followed by the reproduction of the formal demonstration of SP's membership in the NP-Complete class, including the construction and analysis of its polynomial verifier and the step-by-step presentation of the polynomial reduction technique from CLIQUE to SP. The main result and contribution of this article is the development of pedagogical material that aims to enhance the integral understanding of Theory of Computation concepts, assisting the public in effectively assimilating the meaning of computational complexity.*

Keywords—*Theory of Computation, Computational Complexity, NP-Completeness, Set Packing Problem, Polynomial Reduction*

I. Introdução

Teoria da Computação (TC) é a base teórica para a Ciência da Computação e Engenharia da Computação, fornecendo as ferramentas conceituais necessárias para compreender o que pode ser computado e, crucialmente, com qual eficiência. Foi no início da década de 1970 que a disciplina ganhou uma nova dimensão com o surgimento da *Teoria da Complexidade Computacional*, impulsionada pelo trabalho de Stephen Cook (1971) [1] e, notavelmente, por Richard Karp (1972) [2]. Este campo de estudo buscou categorizar problemas com base nos recursos (tempo e espaço) requeridos por seus melhores algoritmos de solução, culminando na definição das classes $\mathcal{P}$ (*Polynomial Time*) e $\mathcal{NP}$ (*Nondeterministic Polynomial Time*) e na formulação do problema $\mathcal{P}$ **versus** $\mathcal{NP}$, definido como um dos Millennium Prize Problems pelo Clay Mathematics Institute [3].

Neste contexto, a complexidade representa o que de fato significa a *intratabilidade* de problemas, ou seja, a prova de que certas questões computacionais não podem ser resolvidas de forma eficiente (em tempo polinomial) por qualquer algoritmo determinístico conhecido, a menos que $\mathcal{P}$=$\mathcal{NP}$. A compreensão da NP-Completude é, portanto, essencial para que o futuro profissional saiba quando buscar soluções exatas ou métodos alternativos como algoritmos aproximados e heurísticos.

Considere por exemplo um conjunto universo U e uma coleção de subconjuntos S. O objetivo é verificar se existe

Dados de contato: Emanuel Badaró Fonseca, badaro.fonseca@uft.edu.br

uma subcoleção dentro de S contendo pelo menos k subconjuntos que sejam mutuamente disjuntos, isto é, que não compartilhem nenhum elemento comum entre si (a interseção entre qualquer subconjunto escolhido é vazia). Este é o Problema do Set Packing (Empacoteamnto de Conjuntos), um dos 21 problemas NP-Completos de Karp [2] e o qual será utilizado como estudo de caso neste artigo.

A complexidade inerente à Teoria da Computação, especificamente no que tange às reduções polinomiais e à classe NP-Completa, representa um desafio pedagógico constante, apesar da relevância teórica de problemas como o Set Packing. Com o objetivo de mitigar essas dificuldades, este trabalho propõe uma construção pedagógica da prova de NP-Completude do Set Packing. A estrutura do artigo segue uma lógica sequencial: a Seção 2 fornece o embasamento teórico e as definições fundamentais. A Seção 3 examina a literatura pertinente e trabalhos correlatos. As Seções 4 e 5 constituem o cerne do trabalho, apresentando a descrição e a demonstração formal do problema, assim como a estratégia de redução. As contribuições e reflexões sobre o aprendizado são debatidas na Seção 6, seguidas pelas conclusões na Seção 7.

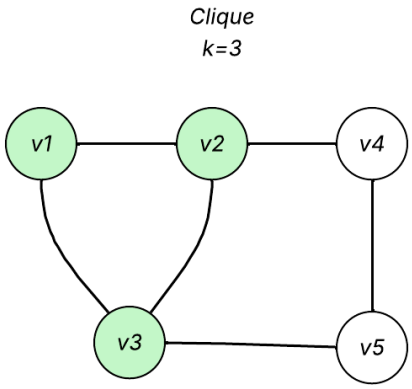


Figura 1: Exemplo de uma clique de tamanho 3. Clique $C = \{v_1, v_2, v_3\}$.

II. PRELIMINARES

Esta seção apresenta os conceitos e ferramentas de análise pertinentes que serão utilizados ao longo de todo este trabalho.

Um *conjunto* é definido como uma coleção não ordenada de elementos distintos. A partir deste conceito, estabelece-se a relação de *subconjunto*: diz-se que um conjunto B é um subconjunto de um conjunto A (denotado por $B \subseteq A$) se todos os elementos presentes em B são também elementos de A. Por exemplo, dado o conjunto $A = \{1,2,3\}$, podemos definir um subconjunto $B = \{1,2\}$. Como ambos os elementos de B estão em A, temos que $B \subseteq A$.

Um *grafo* G é definido formalmente como um par ordenado $G = (V,E)$, composto por um conjunto finito e não vazio de vértices V e um conjunto de arestas E. Neste contexto, os vértices constituem os elementos de V e representam as entidades ou objetos que estão sendo modelados. Já as arestas são definidas como pares não ordenados (u,v) de vértices distintos de V, representando as conexões ou relações estabelecidas entre esses vértices.

Considere o grafo presente na Figura 1 como exemplo, onde o conjunto de vértices é $V = \{v_1, v_2, v_3, v_4, v_5\}$ e o conjunto de arestas é $E = \{(v_1, v_2), (v_2, v_3), (v_1, v_3), (v_4, v_5), (v_3, v_5)\}$. Neste caso, o vértice v_1 está conectado ao vértice v_2 pela aresta (v_1, v_2).

Uma *clique* em um grafo é um subconjunto de vértices $C \subseteq V$ tal que todo par de vértices distintos em C está conectado por uma aresta em E. A Figura 1 mostra um exemplo visual de uma clique de tamanho 3 em um grafo.

Na Teoria da computação, um *problema de decisão* é uma questão que tem uma resposta simples de “sim” ou “não”. É como fazer uma pergunta que pode ser respondida com um "verdadeiro" ou "falso".

Utilizando o grafo da Figura 1 e $k = 3$, a pergunta “Existe uma clique de tamanho 3 neste grafo?” é um problema de decisão cuja resposta é “Sim”.

A *complexidade computacional* é o estudo de quão eficientemente um algoritmo (uma receita para resolver um problema) pode resolver um problema em termos de tempo e recursos (como memória). Medimos o “tempo” pelo número de passos que o algoritmo leva para rodar, especialmente à medida que o tamanho da entrada aumenta.

Para ordenar uma lista de n números, um algoritmo simples pode levar n^2 passos (como o Bubble Sort), enquanto um mais eficiente leva $n \log n$ passos (como o Merge Sort).

Um problema é classificado como pertencente ao Tempo Polinomial $\mathcal{P}$ quando existe um algoritmo capaz de encontrar sua resposta (do tipo sim ou não, para problemas de decisão) em um número de passos que cresce, no máximo, como um polinômio do tamanho da entrada. Por essa razão, problemas em P são geralmente categorizados na literatura como “fáceis” ou “eficientemente solúveis”.

A Classe $\mathcal{NP}$ (Non-deterministic Polynomial time) agrupa problemas de decisão para os quais, dada uma “solução candidata” (denominada certificado), é possível *verificar* sua correção em tempo polinomial. Esta definição não implica necessariamente que a solução possa ser encontrada de forma eficiente, mas sim que podemos confirmar rapidamente a validade de uma proposta apresentada. Vale ressaltar, por fim, que todo problema pertencente à classe $\mathcal{P}$ está também contido em $\mathcal{NP}$.

Por exemplo, no jogo Sudoku, encontrar a solução para um tabuleiro vazio pode ser demorado, mas dada uma grade preenchida (o certificado), verificar se ela segue as regras (não repetir números nas linhas, colunas e quadrantes) é muito rápido (polinomial).

A *Redução Polinomial* ($A \leq_p B$) é uma ferramenta formal utilizada para demonstrar que um problema é “pelo menos tão difícil” quanto outro. O processo consiste na capacidade de transformar eficientemente (em tempo polinomial) qualquer instância de um problema A em uma instância de um problema B, preservando a equivalência das respostas: a instância de A é positiva se, e somente se, a instância correspondente de B também o for.

Como exemplo, suponha que o Problema A seja “encontrar a saída de um labirinto físico de sebes”. Se soubermos

 ISSN: 2675-3588

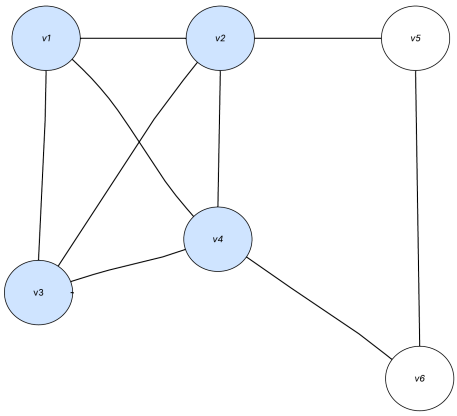


Figura 2: Entrada: o grafo G e $k = 4$. A resposta é "Sim", pois $\{v_1, v_2, v_3, v_4\}$ é uma clique.

transformar (reduzir) este labirinto em um desenho de Grafo (Problema B), onde cada cruzamento é um vértice e cada corredor é uma aresta, podemos usar um algoritmo de computador conhecido para resolver B. Assim, resolver o grafo resolve o labirinto.

O problema da Clique é um problema de decisão NP-Completo.

CLIQUE

Entrada: Um grafo $G = (V, E)$ e um inteiro $k \geq 1$.
Questão: Existe um subconjunto de vértices $V' \subseteq V$ tal que $|V'| \geq k$ e, para todo par de vértices distintos em V', existe uma aresta em E que os conecta?

Considere como exemplo a Figura 2.

O problema do Conjunto Independente também é um problema de decisão NP-Completo.

INDEPENDENT SET

Entrada: Um grafo $G = (V, E)$ e um inteiro $k \geq 1$.
Questão: Existe um subconjunto de vértices $V' \subseteq V$ tal que $|V'| \geq k$ e não há arestas em E conectando quaisquer dois vértices em V'?

Considere como exemplo de Independent Set a Figura 3.

O problema do Empacotamento de Conjuntos é central para este trabalho e pertence à classe dos problemas NP-Completos.

SET PACKING

Entrada: Uma coleção $C = \{S_1, S_2, \ldots, S_m\}$ de subconjuntos de um conjunto universal U, e um inteiro $k \geq 1$.
Questão: Existe uma subcoleção $C' \subseteq C$ tal que $|C'| \geq k$ e todos os conjuntos em C' são mutuamente disjuntos (ou seja, para quaisquer dois conjuntos $S_i, S_j \in C'$ com $i \neq j$, tem-se $S_i \cap S_j = \emptyset$)?

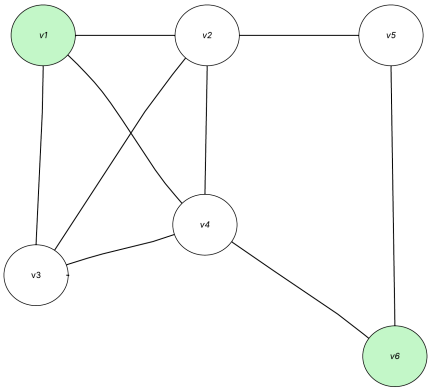


Figura 3: Entrada: Grafo G e $k = 2$. A resposta é "Sim", pois o subconjunto $\{v_1, v_6\}$ tem tamanho 2 e não existe a aresta (v_1, v_6) em $E(G)$.

Exemplo: Entrada: $U = \{1,2,3,4\}$, $C = \{\{1,2\},\{2,3\},\{3,4\}\}$ e $k = 2$. A resposta é "Sim", pois a subcoleção $C' = \{\{1,2\},\{3,4\}\}$ tem tamanho 2 e os conjuntos são disjuntos.

III. Trabalhos Relacionados

A literatura sobre o problema do Set Packing (SP) é extensa, abrangendo desde as provas fundamentais de complexidade até aplicações modernas em teoria dos jogos e otimização. Nesta seção, destacamos trabalhos que fundamentam a teoria, exploram limites de tratabilidade e dialogam com a proposta pedagógica deste artigo.

A referência primária para a classificação do Set Packing é o trabalho clássico de Karp [2], onde demonstrou em suas pesquisas a redutibilidade entre 21 problemas combinatórios, estabelecendo o Set Packing como NP-Completo através de uma cadeia de reduções originada na Satisfiabilidade (SAT). Este trabalho define a posição do problema na hierarquia de complexidade. Complementarmente, Garey e Johnson [4] sistematizaram a metodologia de provas de NP-Completude. O presente artigo adota a estrutura formal de construção e verificação polinomial proposta por eles, adaptando seu rigor técnico a uma abordagem pedagógica.

Avançando para estratégias de resolução exata e modelagem, no trabalho de Delorme, Gandibleux e Rodriguez (2004) [5], podemos notar que eles trabalham com uma nova abordagem de modelagem para a resolução exata. Na pesquisa, os autores abordam o problema de Set Packing propondo uma transformação para o problema de clique máxima e obtêm uma redução significativa na complexidade do espaço de busca. Os resultados são interessantes por demonstrarem como a reformulação do modelo matemático e o uso de limitantes superiores podem acelerar a resolução de instâncias difíceis.

Expandindo o escopo para generalizações com aplicações práticas, no trabalho de Muritiba et al. (2010) [6], podemos notar que eles trabalham com o problema de empacotamento de bins com conflitos (BPPC). Na pesquisa, os autores abordam o problema propondo novos limites inferiores baseados na computação de cliques maximais, novos

limites superiores através de uma abordagem meta-heurística envolvendo busca tabu e um operador de cruzamento, e um algoritmo exato baseado em uma formulação de cobertura de conjuntos, resolvido por meio de geração de colunas e branch-and-price. Os resultados são interessantes por demonstrar a eficácia dos algoritmos propostos em um vasto conjunto de instâncias de referência da literatura, resolvendo 780 de 800 instâncias para a otimalidade e melhorando consistentemente algoritmos anteriores.

Investigando a tratabilidade sob condições específicas, no trabalho de Jia, Zhang e Chen (2004) [7], podemos notar que eles trabalham com a complexidade parametrizada do problema de empacotamento de conjuntos. Na pesquisa, os autores abordam o problema de m-Set Packing (onde o tamanho de cada conjunto é limitado por uma constante m) propondo um algoritmo eficiente de complexidade parametrizada e obtêm a prova de que o problema é tratável por parâmetro fixo (FPT) em relação ao tamanho da solução k. Os resultados são interessantes por demonstrarem que, ao restringir o tamanho dos conjuntos, é possível superar a intratabilidade geral e encontrar soluções exatas de forma eficiente para valores pequenos de k.

Sob a ótica de soluções aproximadas para o caso geral, no trabalho de Fürer e Yu (2014) [8], podemos notar que eles trabalham com a análise teórica de algoritmos de aproximação. Na pesquisa, os autores abordam o problema de k-Set Packing investigando o poder das melhorias locais (local improvements) e obtêm limites de aproximação refinados para o problema. Os resultados são interessantes por aprofundarem o entendimento sobre as limitações e capacidades da busca local, fornecendo garantias mais justas para a qualidade das soluções encontradas por essa classe de algoritmos.

Retornando às inovações em modelagem matemática, no trabalho de Alidaee et al. (2008) [9], podemos notar que eles trabalham com uma nova abordagem baseada em Programação Quadrática Binária Irrestrita (UBQP/QUBO). Na pesquisa, os autores abordam o problema de Set Packing transformando as restrições de disjunção em penalidades na função objetivo quadrática e obtêm soluções de alta qualidade que rivalizam com métodos especializados em termos de tempo e eficiência. Os resultados são interessantes por demonstrarem que uma estrutura unificada e sem restrições pode simplificar a resolução de problemas combinatórios complexos, permitindo o uso de heurísticas genéricas robustas.

Por fim, o trabalho de Lassance e Bianchini [10] investigou o impacto de estratégias didáticas no ensino de Teoria da Computação. O estudo conclui que abordagens descritivas e a participação ativa dos discentes (via seminários) são eficazes para diluir a complexidade das reduções polinomiais. Seguindo esta diretriz pedagógica, nosso trabalho adota uma estratégia de decomposição visual e prova passo a passo, visando contribuir com o entendimento de conceitos abstratos de intratabilidade computacional pelo público em geral.

IV. Descrição do Problema

O problema do Set Packing (Empacotamento de Conjuntos), doravante denominado SP, é um problema fundamental na otimização combinatória e na teoria da complexidade computacional. Formalmente, dado um universo finito U e uma família de subconjuntos $S = \{S_1, S_2, \ldots, S_m\}$, onde $S_i \subseteq U$, um *packing* é uma subcoleção $S' \subseteq S$ tal que todos os conjuntos em S' são mutuamente disjuntos, ou seja, $S_i \cap S_j = \emptyset$ para quaisquer $S_i, S_j \in S'$ distintos [4].

O problema pode ser abordado sob duas perspectivas. Na versão de *otimização*, o objetivo é encontrar a subcoleção S' de cardinalidade máxima. Na versão de *decisão* — a qual utilizamos para a prova de NP-completude — a entrada inclui um inteiro k, e a pergunta é se existe um packing de tamanho pelo menos k ($|S'| \geq k$) [2]. Uma generalização comum é o *Weighted Set Packing*, onde cada conjunto S_i possui um peso w_i, e o objetivo é maximizar a soma dos pesos dos conjuntos disjuntos selecionados.

A relevância do SP decorre de sua capacidade de modelar situações de alocação de recursos onde o compartilhamento é impossível (restrição de exclusividade). A aplicação mais notável ocorre em *Leilões Combinatórios* [11]. Neste cenário, um leiloeiro tem um conjunto de itens distintos (U) e os licitantes oferecem lances por "pacotes" de itens (S_i). Como cada item só pode ser vendido uma vez, o leiloeiro deve selecionar um conjunto de lances vencedores que não disputem o mesmo item, maximizando o lucro total. Outras aplicações críticas incluem o *Airline Crew Scheduling* (escalonamento de tripulações), onde cada voo deve ser coberto por uma única equipe e as equipes (conjuntos de voos) não podem estar em dois lugares ao mesmo tempo [12].

Para fins pedagógicos, o SP pode ser visualizado como o dilema de um organizador de festas que possui uma lista de grupos de amigos que desejam comparecer ao evento, Figura 4. O universo U representa as cadeiras disponíveis na mesa principal. Cada grupo (S_i) exige sentar-se em cadeiras específicas e recusa-se a compartilhar seus assentos com estranhos. Se um grupo deseja a cadeira 3 e outro também deseja a cadeira 3, eles são incompatíveis. O desafio do organizador é aceitar o maior número possível de grupos sem gerar conflitos de assentos.

Embora o SP seja NP-difícil no caso geral, existem subclasses importantes que admitem algoritmos eficientes:

- **Cardinalidade Limitada ($|S_i| \leq 2$):** Se todos os conjuntos na família S tiverem no máximo 2 elementos, o problema torna-se equivalente ao *Maximum Matching* (Emparelhamento Máximo) em grafos. Neste caso, os elementos de U são vértices e os conjuntos S_i são arestas. O problema pode ser resolvido em tempo polinomial $O(E\sqrt{V})$ pelo algoritmo de Edmonds [13].

- **Grafos de Intervalo:** Se os elementos de U puderem ser ordenados linearmente tal que cada S_i forme um intervalo contíguo, o problema equivale ao *Interval Scheduling*, que pode ser resolvido com uma estratégia gulosa em $O(n \log n)$ [14].

Para o caso geral, algoritmos exatos baseados em programação dinâmica ou inclusão-exclusão atingem complexidades da ordem de $O^*(2^n)$, o que é impraticável para instâncias grandes [15]. Essa intratabilidade computacional, contrastando com a eficiência das subclasses restritas apresentadas, sugere que o Set Packing pertence à classe dos problemas mais difíceis da computação. A seção a seguir formaliza essa intuição, provando a NP-Completude

 ISSN: 2675-3588

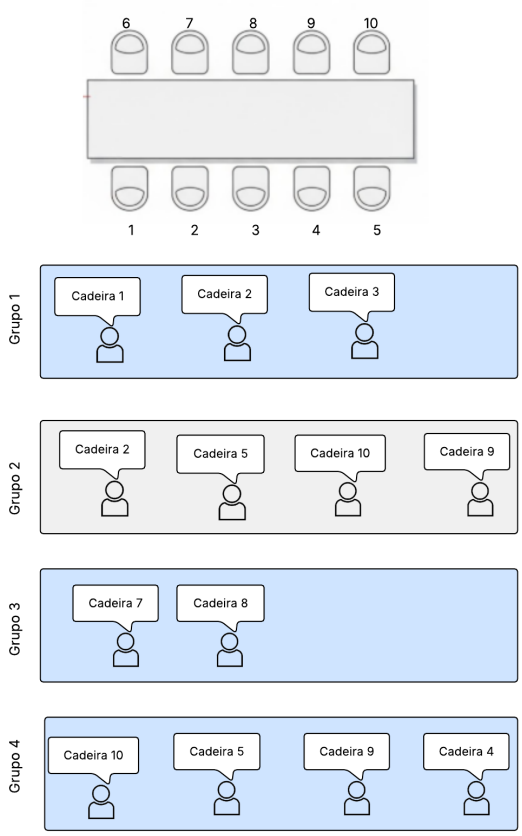


Figura 4: Representação visual de uma instância de Set Packing. O universo $U = \{1,..,10\}$. Os grupos 1, 3 e 4 formam um packing válido (azul). O grupo 2 cria conflito com os grupos 1 e 4.

do problema através de uma redução polinomial a partir do problema da Clique.

V. Demonstração e Contribuições

A seguir, estabelecemos o lema principal deste artigo.

Lema 1. *Set Packing é NP-Completo.*

Proof. Seguindo o esquema clássico, dividimos a prova em duas etapas identificadas: (1) provar que $SP \in \mathcal{NP}$ (NP-pertinência) e (2) provar que SP é NP-difícil por meio de uma redução polinomial apropriada.

Tome (1)

Para mostrar que $SP \in \mathcal{NP}$, basta exibir um verificador polinomial que, dada uma solução candidata S', determine se ela constitui um subconjunto de l conjuntos mutuamente disjuntos.

A instância do problema consiste em um universo U com $n = |U|$ elementos e uma coleção de m conjuntos $S = \{S_1, \ldots, S_m\}$. O certificado é uma subcoleção $S' \subseteq S$ supostamente de tamanho l.

O verificador executa duas tarefas: (i) verificar se $|S'| = l$ e (ii) verificar a disjunção par a par entre os conjuntos de S'.

Temos que o primeiro passo é executado em $O(|S'|)$, que é limitado por $O(m)$, pois $|S'| \leq m$.

Para o segundo passo, o algoritmo percorre cada par distinto de conjuntos em S'. O número de pares é dado pela combinação de $|S'|$ elementos tomados 2 a 2:

$$\binom{|S'|}{2} = \frac{|S'|!}{2!(|S'|-2)!} = \frac{|S'|(|S'|-1)}{2} = \frac{|S'|^2 - |S'|}{2}.$$

Como o termo dominante é quadrático e sabemos que $|S'| \leq m$, conclui-se que o número de verificações é limitado por $O(m^2)$.

Para cada par (S_i, S_j), verifica-se se $S_i \cap S_j = \emptyset$. Se representarmos cada conjunto como uma lista de elementos, o teste de interseção pode ser feito verificando elemento a elemento, exigindo tempo proporcional ao tamanho total do universo. Assim, cada teste leva tempo $O(n)$.

Portanto, o tempo total gasto na verificação das disjunções é:

$$O(m^2) \cdot O(n) = O(m^2 n).$$

Neste termo, o fator quadrático m^2 provém da comparação de todos os pares possíveis de conjuntos, enquanto o fator linear n surge do custo computacional de verificar a interseção de dois conjuntos específicos.

Somando todos os passos, obtemos um verificador com custo máximo

$$O(m) + O(m^2 n) = O(m^2 n),$$

que é polinomial no tamanho da entrada. Assim, concluímos que $SP \in \mathcal{NP}$.

Tome (2)

Para provar a NP-dificuldade, reduziremos o problema da Clique ao Set Packing. O problema da Clique é escolhido como origem pois a relação "dois vértices são conectados" pode ser mapeada inversamente para "dois conjuntos são disjuntos" se construirmos o universo baseados nas arestas que *não* existem. Assim temos a seguinte redução: CLIQUE $\leq_p$ SP.

Seguiremos com a estratégia de construção em que a redução $f(G,k) = (U,S,l)$ transforma uma instância de Clique em uma de Set Packing preservando a propriedade isomórfica

$$\textit{Vértices adjacentes em } G \iff \textit{Conjuntos disjuntos em } S.$$

A construção define o inteiro l com o mesmo tamanho da clique, ou seja, $l = k$. O universo U é composto pelas **não-arestas** de G, de modo que cada par de vértices que *não* está conectado em G vira um elemento em U:

$$U = \{(v_i, v_j) \mid v_i, v_j \in V, i \neq j, (v_i, v_j) \notin E\}$$

O propósito desta construção é que, se dois vértices não têm aresta, eles "compartilham" um conflito (o elemento em U), impedindo que sejam escolhidos juntos. Para a coleção S, para cada vértice $v_i \in V$, criamos um conjunto S_i que contém todas as não-arestas incidentes a v_i:

$$S_i = \{(v_i, v_j) \in U \mid j \neq i\}$$

A Figura 5 ilustra um exemplo visual desta redução, destacando como as não-arestas formam o universo de conflito.

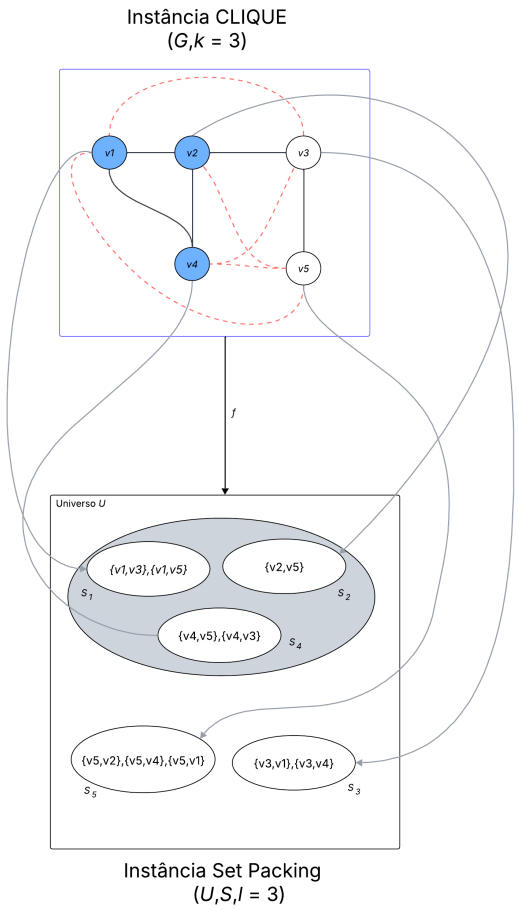


Figura 5: A existência da clique $C = \{v_1, v_2, v_4\}$ (em azul) no grafo implica a existência de uma subcoleção de conjuntos disjuntos $S' = \{S_1, S_2, S_4\}$ na instância de Set Packing.

No exemplo ilustrado na Figura 5, é possível observar a redução f de CLIQUE para Set Packing observa-se que os vértices v_3 e v_5 não são adjacentes a todos os membros da clique, o que gera um conjunto de não-arestas (como (v_1, v_3), (v_3, v_4), (v_2, v_5), (v_4, v_5) e (v_1, v_5)) que passam a constituir o universo U. Pela regra de construção, onde cada S_i contém as não-arestas incidentes a v_i, os conjuntos S_3 e S_5 acabam compartilhando elementos com outros conjuntos, o que representa conflitos. Em contrapartida, como v_1, v_2 e v_4 estão plenamente conectados entre si, não existem não-arestas entre eles no universo, garantindo que seus conjuntos correspondentes S_1, S_2 e S_4 sejam mutuamente disjuntos, satisfazendo a condição de validade do Set Packing.

Contudo, não basta apenas montar a construção da redução. Também precisamos mostrar que (i) esta redução é polinomial e (ii) preserva a simetria entre os problemas.

(i). Devemos provar que a transformação preserva a resposta do problema original, ou seja, (G,k) tem Clique $\iff$ (U,S,l) tem Set Packing.
($\Rightarrow$) *Se G tem clique de tamanho k, então* (U,S) *tem packing de tamanho* $l = k$.

Proof. Seja C a clique em G. Selecionamos os conjuntos correspondentes $S' = \{S_i \mid v_i \in C\}$. Temos $|S'| = |C| = k = l$. Para quaisquer dois conjuntos distintos $S_i, S_j \in S'$, os vértices correspondentes v_i, v_j estão na clique. Logo, a aresta (v_i, v_j) *existe* em E. Como U contém apenas *não-arestas*, o par $(v_i, v_j) \notin U$. A única interseção possível entre S_i e S_j seria o elemento (v_i, v_j). Como $(v_i, v_j) \notin U$, ele não pode estar nem em S_i nem em S_j. Logo, $S_i \cap S_j = \emptyset$. Assim, S' é um Set Packing válido. □

($\Leftarrow$) *Se* (U,S) *tem packing de tamanho* $l = k$*, então G tem clique de tamanho k.*

Proof. Seja S' o packing. Selecionamos os vértices $C = \{v_i \mid S_i \in S'\}$. Temos $|C| = |S'| = l = k$. Suponha, por contradição, que C *não* seja uma clique. Então existem dois vértices $v_i, v_j \in C$ tal que a aresta $(v_i, v_j) \notin E$. Se a aresta não existe, então o par (v_i, v_j) é uma não-aresta, logo $(v_i, v_j) \in U$. Pela construção, S_i contém todas as não-arestas de v_i (incluindo (v_i, v_j)) e S_j contém todas as de v_j (incluindo (v_i, v_j)). Portanto, $(v_i, v_j) \in S_i \cap S_j$, o que implica $S_i \cap S_j \neq \emptyset$. Isso contradiz a hipótese de que S' é um packing (conjuntos disjuntos). Logo, a aresta deve existir para todos os pares, e C é uma clique. □

(ii). A construção do universo U exige iterar sobre todos os pares de vértices, uma operação limitada por $\binom{n}{2} = O(n^2)$. A construção da coleção S exige criar n conjuntos, onde para cada um verificamos $n-1$ pares, totalizando também $O(n^2)$. Portanto, conclui-se que a função de redução f é executada em tempo polinomial: $O(n^2)$.

Assim por (1) e (2), provamos o Lema 1 ao demonstrarmos que o Set Packing é NP-Completo, validando sua pertinência a $\mathcal{NP}$ e em seguinda apresentando uma redução polinomial a partir do CLIQUE. Esta prova ilustra o uso de "conflitos" (neste caso, não-arestas) como elementos básicos de construção para forçar restrições de exclusão mútua. □

VI. Resultados e Reflexões

A elaboração deste trabalho resultou na produção de um material didático autossuficiente para o estudo da NP-Completude do problema *Set Packing* (SP). Diferentemente de abordagens tradicionais que frequentemente omitem passos intermediários das reduções polinomiais, os resultados aqui apresentados focam na explicitação da lógica construtiva. A principal contribuição pedagógica deste estudo é a sistematização visual e analógica da redução $CLIQUE \leq_p SP$. Na literatura clássica, a definição do universo U como o conjunto de "não-arestas" é frequentemente apresentada de forma puramente algébrica, o que dificulta a visualização geométrica por parte do estudante. Para mitigar essa barreira, desenvolvemos a analogia do "Organizador de Festas", uma narrativa lúdica que provou-se eficaz para traduzir a restrição abstrata de "interseção vazia" para uma restrição concreta de "conflito de assentos", facilitando a intuição inicial sobre o problema. Adicionalmente, a diagramação da redução apresentada nas Figuras 5 e 4 permite ao aluno rastrear visualmente como um vértice no grafo se transforma em um conjunto, e como a ausência de uma aresta se materializa em um elemento compartilhado no universo U.

Durante a estruturação da prova de NP-Dificuldade, identificamos que a maior dificuldade cognitiva reside na "inversão lógica" exigida pela redução a partir do problema

 ISSN: 2675-3588

da Clique. Enquanto reduções a partir do *Independent Set* (IS) mapeiam arestas diretamente para elementos do universo (conflito direto), a redução a partir da Clique exige o uso do grafo complementar (ou não-arestas). Essa distinção é sutil e é uma fonte comum de erro. Para superar esse obstáculo, adotamos a estratégia de definir explicitamente o universo U como um conjunto de "conflitos potenciais", reforçando que, para que um *packing* (pacote de vértices) seja válido, os elementos não podem ter conflitos (não-arestas) entre si — o que força a existência das arestas no grafo original.

Em termos de aplicabilidade acadêmica, este artigo foi estruturado para servir como material complementar na disciplina de Teoria da Computação. A Seção 4 (Descrição do Problema) pode ser utilizada como texto introdutório para aulas sobre problemas de empacotamento, enquanto a Seção 5 (Demonstração) serve como guia para listas de exercícios avançados que exigem a formalização de reduções.

Por fim, embora o foco deste trabalho seja a intratabilidade (NP-Completude), é importante refletir sobre o comportamento prático. Em cenários reais, como os leilões combinatórios mencionados, não se busca a prova de inexistência de solução, mas sim a melhor solução possível em tempo hábil. Experimentos simples com algoritmos gulosos (selecionar o menor conjunto disponível iterativamente) demonstram que, embora não garantam a solução ótima (o k máximo), oferecem aproximações rápidas. Esta constatação reforça a importância pedagógica de distinguir entre a *dificuldade do pior caso* (foco da teoria $\mathcal{NP}$) e a *solubilidade prática* via heurísticas.

VII. Considerações Finais

Este trabalho cumpriu seu objetivo principal de provar a NP-Completude do problema *Set Packing* (SP), oferecendo uma abordagem pedagógica que preenche a lacuna entre a definição formal e a intuição geométrica. Através da redução polinomial a partir do problema da Clique ($CLIQUE \leq_p SP$), demonstramos que a dificuldade computacional de encontrar grupos mutuamente exclusivos em uma coleção é equivalente a encontrar subgrafos completos. O principal resultado obtido não foi apenas a reafirmação da complexidade do problema, mas a sistematização de um método de ensino que utiliza analogias lúdicas (o "Organizador de Festas") e diagramas visuais passo a passo para facilitar a assimilação de conceitos abstratos por estudantes de graduação.

É importante ressaltar, contudo, que o escopo deste artigo limitou-se à análise da *complexidade de pior caso* e à versão de *decisão* do problema. Entre as limitações, destaca-se a ausência de implementação de solucionadores exatos (como *branch-and-bound*) ou heurísticos para resolver instâncias do problema, visto que a implementação restringiu-se ao algoritmo verificador polinomial para validação da classe $\mathcal{NP}$, e não para avaliação de desempenho em *benchmarks*. Além disso, optou-se pelo foco em uma redução única via Clique para garantir a profundidade e a clareza didática, em detrimento da abrangência de outras reduções possíveis, como a partir do *Exact Cover* ou *3-SAT*.

A base teórica estabelecida neste artigo abre diversas frentes para investigação acadêmica e desenvolvimento didático em trabalhos futuros. Uma extensão natural seria o estudo comparativo de algoritmos gulosos e meta-heurísticas (como Algoritmos Genéticos ou *Simulated Annealing*) para a versão de otimização do Set Packing, analisando o *gap* de aproximação dessas soluções. Outra perspectiva relevante é a análise em classes de grafos especiais, visto que o Set Packing torna-se solúvel em tempo polinomial quando o universo e os conjuntos podem ser modelados como grafos de intervalo ou grafos de corda. Adicionalmente, sugere-se investigar a complexidade parametrizada (FPT) para verificar a tratabilidade do problema para valores pequenos de k. Por fim, propõe-se a extensão pedagógica através do desenvolvimento de uma ferramenta de software interativa que permita aos alunos desenharem grafos e visualizarem, em tempo real, a transformação dos vértices e arestas nos conjuntos do Set Packing. Conclui-se que o Set Packing, apesar de sua complexidade inerente, é um excelente veículo para o ensino da Teoria da Computação, servindo como porta de entrada para discussões mais amplas sobre otimização combinatória e limites da computabilidade.

Referências

[1] S. A. Cook, "The complexity of theorem-proving procedures," in *Proceedings of the third annual ACM symposium on Theory of computing*, 1971, pp. 151–158.

[2] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of computer computations*, R. E. Miller and J. W. Thatcher, Eds. Plenum Press, 1972, pp. 85–103.

[3] Clay Mathematics Institute, "Millennium prize problems," https://www.claymath.org/millennium-problems/, 2000, acesso em: 30 nov. 2025.

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

[5] M. Delorme, X. Gandibleux, and J. Rodriguez, "A new approach for modeling and solving set packing problems," *European Journal of Operational Research*, vol. 152, no. 3, pp. 626–645, 2004.

[6] A. E. F. Muritiba, M. Iori, E. Malaguti, and P. Toth, "Algorithms for the bin packing problem with conflicts," *INFORMS Journal on Computing*, vol. 22, no. 3, pp. 401–415, 2010.

[7] W. Jia, C. Zhang, and J. Chen, "An efficient parameterized algorithm for m-set packing," *Journal of Algorithms*, vol. 50, no. 1, pp. 106–117, 2004.

[8] M. Fürer and H. Yu, "Approximating the k-set packing problem by local improvements," in *International Symposium on Combinatorial Optimization (ISCO 2014)*, ser. Lecture Notes in Computer Science, vol. 8596. Springer, 2014, pp. 408–420.

[9] B. Alidaee, G. Kochenberger, K. Lewis, M. Lewis, and H. Wang, "A new approach for modeling and solving set packing problems," *European Journal of Operational Research*, vol. 186, no. 2, pp. 504–512, 2008.

[10] B. Yasser, Lassance e Guilherme, "Reflexões e práticas pedagógicas no escopo da disciplina deteoria da computação," *Academic Journal on Computing, Engineering and Applied Mathematics*, 2025, referência pedagógica da disciplina de Teoria da Computação.

[11] S. DeVries and R. Vohra, *Combinatorial Auctions: A Survey*. Springer, 2003.

[12] K. L. Hoffman and M. Padberg, "Cutting plane algorithms in planning and scheduling," *Annals of Operations Research*, vol. 43, no. 1, pp. 55–85, 1993.

[13] J. Edmonds, "Paths, trees, and flowers," in *Combinatorial Optimization: Proceedings of the Symposium in Applied Mathematics*. American Mathematical Society, 1965, vol. 18, pp. 48–57.

[14] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*, 5th ed. Berlin: Springer, 2012.

[15] A. Björklund, T. Husfeldt, and M. Koivisto, "Set packing and covering with character," in *Proc. 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2010, pp. 1090–1099.

ISSN: 2675-3588

Reprodução e Contribuições Pedagógicas: Casamento Máximo em Grafos Bipartidos e suas Generalizações

Reproduction and Pedagogical Contributions: Maximum Matching in Bipartite Graphs and its Generalizations

Vitória Milhomem Soares[1], Matheus de Sousa Silva[1], Daniel Martins da Silva[1] e Tanilson Dias dos Santos[1]

[1] *Universidade Federal do Tocantins, Ciência da Computação, Tocantins, Brasil*

Data de recebimento do manuscrito: 01/12/2025
Data de aceitação do manuscrito: 05/01/2026
Data de publicação: 10/02/2026

Resumo—Este artigo apresenta um estudo didático sobre o problema do *Casamento Máximo* em grafos, com ênfase na estrutura de grafos bipartidos e suas generalizações. O objetivo é reproduzir resultados fundamentais que fogem da abordagem clássica de König e Hall. Para isso, exploramos o Teorema de Tutte, que condiciona o emparelhamento perfeito à análise de componentes ímpares, e o Teorema de Dilworth, que estabelece uma dualidade com conjuntos parcialmente ordenados (posets). A metodologia emprega a análise de provas, utilizando técnicas de redução e decomposição, acompanhada de exemplos lúdicos e visualizações estratégicas. Como resultados centrais, demonstramos que a condição de Tutte é o obstáculo estrutural universal para o emparelhamento perfeito, e que a equivalência de Dilworth é a base para a eficiência algorítmica, como exemplificado pelos trabalhos de Kameda e Munro. Em conclusão, este estudo preenche lacunas conceituais e oferece uma contribuição pedagógica significativa, tornando o rigor da Teoria dos Grafos mais acessível a estudantes de graduação e promovendo uma visão unificada sobre a existência de emparelhamentos e coberturas de cadeias.

Palavras-chave—Casamento Máximo, Teorema de Tutte, Teorema de Dilworth, Kameda-Munro, Didática em Grafos.

Abstract—*This paper presents a didactic study on the Maximum Matching problem in graphs, with an emphasis on bipartite graph structures and their generalizations. The objective is to reproduce fundamental results that move beyond the classical approach of König and Hall. To this end, we explore Tutte's Theorem, which conditions perfect matching on the analysis of odd components, and Dilworth's Theorem, which establishes a duality with partially ordered sets (posets). The methodology employs the analysis of proofs, utilizing techniques of reduction and decomposition, accompanied by illustrative examples and strategic visualizations. As central results, we demonstrate that Tutte's condition is the universal structural obstacle to perfect matching, and that Dilworth's equivalence establishes a rigorous reduction between poset decomposition and bipartite matching, enabling efficient polynomial-time solutions as exemplified by the works of Kameda and Munro. In conclusion, this study fills conceptual gaps and offers a significant pedagogical contribution, making the rigor of Graph Theory more accessible to undergraduate students and promoting a unified view on the existence of matchings and chain covers.*

Keywords—*Maximum Matching, Tutte's Theorem, Dilworth's Theorem, Kameda-Munro, Graph Theory Education.*

I. Introdução

A *Teoria dos Grafos* atua como a linguagem universal da Ciência da Computação, oferecendo a estrutura necessária para modelar desde redes sociais complexas até a arquitetura microscópica de circuitos integrados [2] . No centro dessa teoria, o *Problema do Casamento Máximo (Maximum Matching)* ocupa uma posição de destaque. Em termos simples, este problema busca encontrar a maior quantidade possível de pares dentro de um grupo, sem que ninguém "sobre" ou participe de mais de um par. Embora a definição pareça simples, sua resolução possui implicações profundas e diretas em cenários reais como: a alocação eficiente de tarefas em processadores, a distribuição de médicos em plantões hospitalares e a otimização de sistemas de recomendação.

Tradicionalmente, o ensino introdutório de *emparelhamentos* em grafos concentra-se quase exclusivamente em grafos *bipartidos* — cenários onde os vértices podem ser divididos em dois grupos distintos (como tarefas e

Dados de contato: Vitória M. Soares, vitoria.milhomem@mail.uft.edu.br

trabalhadores). Nesses casos, os clássicos teoremas de *Hall* [5] e *König* [9] oferecem soluções fundamentais e bem conhecidas. No entanto, o mundo real nem sempre é bipartido. Quando as restrições de conexão são mais complexas e formam grafos gerais, as ferramentas básicas deixam de funcionar. É neste ponto que este artigo se insere: propomos uma abordagem pedagógica para transpor a barreira dos grafos bipartidos, explorando o *Teorema de Tutte* [14], que generaliza a existência de emparelhamentos através de uma análise de paridade topológica (componentes ímpares).

Além de tratar de grafos gerais, buscamos conectar a teoria dos grafos à *teoria da ordem*. Para isso, revisitamos o *Teorema de Dilworth* [3] , que estabelece uma dualidade surpreendente entre o tamanho de emparelhamentos e a estrutura de conjuntos parcialmente ordenados (posets). Para amarrar a teoria à prática computacional, discutimos como essas propriedades estruturais fundamentam algoritmos eficientes, como os estudados por *Kameda* e *Munro* [8] , que utilizam tais decomposições para resolver o problema em tempo polinomial.

Portanto, o objetivo deste trabalho é duplo e focado na didática. Buscamos fornecer demonstrações passo a passo e rigorosas destes teoremas avançados, bem como oferecer contribuições pedagógicas concretas. A intenção é utilizar exemplos lúdicos e visualizações estratégicas para facilitar a intuição do estudante, revelando conceitos abstratos como a barreira de componentes ímpares ou a cobertura de cadeias, transformando a demonstração matemática em uma narrativa lógica e compreensível.

As seções subsequentes guiarão o leitor por essa jornada, começando pelas Preliminares (Seção II), seguidas pelos Trabalhos Relacionados (Seção III) e a Descrição do Problema (Seção IV), avançando para as Demonstrações passo a passo (Seção V), a análise de Resultados e Reflexões (Seção VI), culminando nas Considerações Finais (Seção VII). No entanto, para que o rigor e a didática pretendidos sejam plenamente alcançados, é imperativo que o leitor domine o vocabulário e as estruturas básicas que sustentam toda esta competência.

Desta forma, para que a complexidade dos teoremas principais possa ser abordada, dedicamos a próxima seção, Preliminares, a estabelecer o vocabulário formal e a intuição essencial sobre emparelhamentos e as estruturas de paridade que serão cruciais nas demonstrações subsequentes.

II. Preliminares

Um *grafo* $G = (V,E)$ é uma estrutura composta por um conjunto de vértices V (pontos) e um conjunto de arestas E (linhas que conectam esses pontos) [2]. Um grafo é dito *bipartido* quando o seu conjunto de vértices V pode ser particionado em dois grupos disjuntos, A e B, de tal forma que todas as arestas conectam um vértice de A a um vértice de B. Não existem arestas conectando dois vértices dentro do mesmo grupo (ex: não há arestas de A para A).

Nota Didática: Para compreender intuitivamente a distinção estrutural de um grafo bipartido, imagine que o conjunto de vértices do grafo é particionado em dois subconjuntos disjuntos e independentes, que rotulamos como A e B. A Figura 1 ilustra visualmente essa partição. Podemos

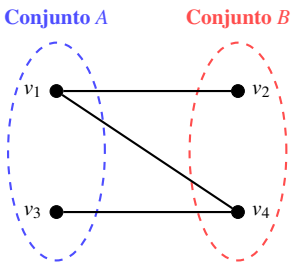


Figura 1: Exemplo visual de Grafo Bipartido $G = (A \cup B, E)$. Note a ausência de arestas "verticais" dentro de cada conjunto.

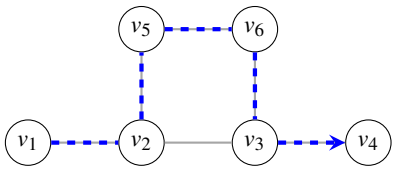


Figura 2: A linha pontilhada em azul é um exemplo de um Caminho Simples, onde nenhum vértice é repetido.

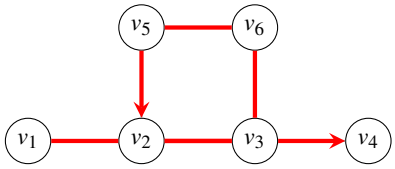


Figura 3: Exemplo de um Caminho não Simples, com início em v_1 e término em v_4. O caminho repete os vértices v_2 e v_3 ao passar pelo ciclo $v_2 - v_3 - v_6 - v_5 - v_2$.

descrever esses subconjuntos através de um exemplo lúdico: considere os vértices do subconjunto A como "Tarefas" e os vértices do subconjunto B como "Trabalhadores". A regra fundamental de um grafo bipartido é a sua restrição de conectividade: as interações (representadas pelas arestas) só podem ocorrer entre um vértice pertencente ao subconjunto A e um vértice pertencente ao subconjunto B. É crucial notar que não existe conectividade interna; ou seja, não há arestas entre dois vértices que pertençam ao mesmo subconjunto (A ou B). Essa restrição impõe uma estrutura menos densa e mais restrita, facilitando a análise e a busca por emparelhamentos. Em contraste, um grafo geral permite interações irrestritas, o que pode levar à formação de ciclos ímpares (como triângulos), que são a principal fonte de complicação e o foco do Teorema de Tutte [14].

O *grau* de um vértice em um grafo é o número de arestas que estão conectadas a ele. A *vizinhança* de um vértice v em um grafo G é o conjunto composto por todos os vértices adjacentes a v, onde ,vértices adjacentes, são aqueles conectados por uma aresta a v.

Um *caminho* em um grafo é uma sequência de vértices interligados por arestas, onde o último vértice de uma aresta é o primeiro da próxima. Um caminho simples é aquele que não repete vértices. O comprimento de um caminho é a quantidade de arestas que o compõem [2]. A Figura 2 ilustra um exemplo claro de um caminho simples, em contraste com o caminho não simples, onde a repetição de vértices ocorre, conforme detalhado na Figura 3.

Um *subgrafo* de um grafo G, essencialmente, é um grafo cujo conjunto de vértices e conjunto de arestas são subconjuntos de G. Uma *componente conexa* em um grafo é um subgrafo onde todos os vértices estão conectados entre

ISSN: 2675-3588

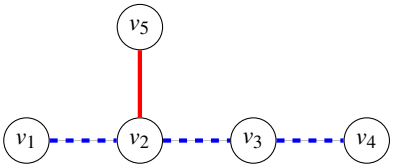


Figura 4: Comparação entre o Caminho Máximo (azul tracejado) e um Caminho Maximal que não é máximo (vermelho)

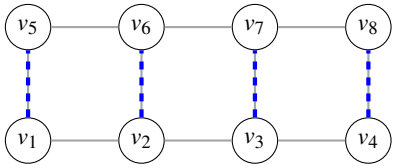


Figura 5: Exemplo de um Emparelhamento Perfeito. As arestas destacadas formam um emparelhamento que satura todos os 8 vértices do grafo.

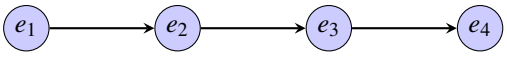


Figura 6: Exemplo visual de uma Cadeia. Elementos em sequência ordenada.

Figura 7: Exemplo visual de uma Anticadeia. Elementos totalmente independentes.

si por caminhos, formando um "pedaço" isolado do grafo original [2].

A análise dessas subestruturas nos leva à necessidade de distinguir entre o maior elemento local e o maior elemento global, conceitos fundamentais na otimização: um elemento $M \in S$ é classificado como *máximo* se for maior ou igual a todos os outros elementos em S. Esta é uma propriedade de natureza global. Se um elemento máximo existe, ele é intrinsecamente único dentro do conjunto. Por outro lado, um elemento $m \in S$ é classificado como *maximal* se não houver nenhum outro elemento em S que seja estritamente maior do que m na relação de ordem definida. Esta é uma propriedade de natureza local, o que implica que um conjunto pode conter múltiplos elementos maximais que não são comparáveis entre si.

A distinção reside na comparabilidade: um elemento máximo domina todos os outros, enquanto um elemento maximal apenas garante não ser dominado por nenhum outro. Consequentemente, todo elemento máximo é, por definição, maximal; contudo, a recíproca não é verdadeira.

A Figura 4 demonstra visualmente a diferença conceitual: o caminho $P_m = v_2 - v_5$ (em vermelho) é classificado como maximal porque, sendo v_5 um vértice de grau 1, ele não pode ser estendido. Contudo, ele não é máximo, pois o grafo contém o caminho $P_M = v_1 - v_2 - v_3 - v_4$ (em azul), que possui 3 arestas e representa o maior caminho possível do grafo.

Dado um grafo $G = (V, E)$, um *emparelhamento* M é um subconjunto de arestas, $M \subseteq E$, tal que quaisquer duas arestas em M não possuem vértices em comum. Um emparelhamento M é máximo se o número de arestas em M, $|M|$, é o maior possível dentre todos os emparelhamentos existentes no grafo G. Já um emparelhamento M é perfeito se satura (cobre) todos os vértices em V. Isso implica que todo vértice $v \in V$ é ponta de exatamente uma aresta em M. É importante notar que um emparelhamento perfeito só pode existir se o número de vértices $|V|$ for par.

Intuição: Um emparelhamento representa "escolhas exclusivas". Se os vértices fossem pessoas e as arestas fossem parcerias de dança, um emparelhamento garantiria que ninguém está tentando dançar com duas pessoas ao mesmo tempo. O emparelhamento perfeito, onde ninguém fica sem par, ilustra o resultado ideal do Casamento Máximo. A saturação de todos os vértices é o objetivo que a Figura 5 demonstra.

Seja G um grafo (ou um subgrafo), um componente ímpar é uma componente conexa do grafo que possui uma quantidade ímpar de vértices $(1, 3, 5,$ etc.$)$. A quantidade total dessas componentes ímpares no grafo é denotada por $o(G)$.

Este é o coração do Teorema de Tutte [14]. Emparelhamentos sempre formam pares (número par: $2, 4, 6 \ldots$). Em um componente com número ímpar de vértices, é matematicamente impossível emparelhar todos internamente: sempre sobrará pelo menos um vértice. Essa "sobra" cria a necessidade de buscar par fora do componente.

Uma relação de *ordem parcial* em um grafo é uma estrutura que define uma hierarquia ou precedência entre alguns dos seus vértices. Um *conjunto parcialmente ordenado* (poset) em um grafo é uma representação visual de um conjunto de elementos onde uma relação de ordem parcial é definida.

Considere um poset P, onde existe uma relação de ordem "$\leq$" definida entre alguns elementos. Nessa estrutura, uma *Cadeia* é um subconjunto de elementos onde todos são comparáveis entre si, seguindo uma sequência linear (como uma fila indiana ou uma linha do tempo, onde $a \leq b \leq c$). Em contraste, uma *Anticadeia* é um subconjunto de elementos onde ninguém é comparável com ninguém, representando elementos totalmente independentes ou simultâneos.

Uma analogia para compreender essas estruturas é a árvore genealógica. Uma Cadeia representa uma linhagem direta (Bisavô $\rightarrow$ Avô $\rightarrow$ Pai $\rightarrow$ Filho), onde a hierarquia é clara e sequencial. Observando a Figura 6, os vértices e_1, e_2, e_3 e e_4 exemplificam essa relação de ordem total: a presença das arestas direcionadas indicando que e_1 leva a e_2, que por sua vez leva a e_3, confirma que todos os elementos neste caminho são comparáveis entre si.

Em contraste, uma Anticadeia corresponde a um grupo de irmãos ou primos que não possuem relação de descendência direta entre si. Conforme ilustrado na Figura 7, os vértices denotados por c_1, c_2, c_3 e c_4 materializam essa propriedade: a ausência total de arestas conectando c_1 a c_2, ou qualquer outro par, evidencia que eles são incomparáveis. Eles coexistem no mesmo "nível" hierárquico sem que nenhum elemento preceda ou suceda o outro, mantendo essa independência mútua até o enésimo elemento c_k.

Munidos dessas definições fundamentais, dispomos do vocabulário necessário para compreender a evolução histórica da teoria. Essas estruturas básicas não são meras abstrações; elas serviram como blocos de construção para os teoremas de dualidade que definem a área. Para além dos clássicos, a seção a seguir contextualiza como os pioneiros da teoria dos grafos manipularam esses conceitos para transitar de soluções em estruturas simples para a

complexidade dos grafos gerais, e como obras mais recentes vêm complementando e refinando essas técnicas estruturais, tornando-as aplicáveis aos desafios computacionais atuais.

Dessa forma, a revisão bibliográfica subsequente organiza-se para refletir sobre trabalhos relacionados ao tema central do nosso estudo.

III. Trabalhos Relacionados

A literatura fundamental sobre emparelhamentos, que serve de base para as definições utilizadas neste artigo, remonta ao período clássico da Teoria dos Grafos. As obras de König [9] e Hall [5] estabeleceram as condições de existência em grafos bipartidos, enquanto Berge [1] introduziu a dinâmica dos caminhos aumentantes. Para o contexto de grafos gerais e ordens parciais, as generalizações propostas por Tutte [14] e Dilworth [3] são as referências primárias. Embora estes trabalhos sejam seminais, a pesquisa na área continua ativa, focando-se especialmente na eficiência algorítmica e em novas abordagens pedagógicas.

No âmbito da otimização algorítmica e suas aplicações em Inteligência Artificial, o trabalho de Tassa [13] oferece uma perspectiva relevante sobre a identificação de arestas. O autor investiga o problema de encontrar todas as arestas "maximamente emparelháveis" (aquelas que pertencem a pelo menos um emparelhamento máximo) em grafos bipartidos. Tassa [13] propõe um algoritmo baseado na decomposição do grafo em componentes fortemente conexos, otimizando a abordagem anterior ao reduzir o tamanho do grafo direcionado auxiliar para $max\{|V_1|,|V_2|\}$ nós. Além disso, o estudo estabelece uma conexão importante com a área de Problemas de Satisfação de Restrições (CSPs), reconhecendo que técnicas similares foram exploradas pioneiramente por Régin [12] para algoritmos de filtragem. Essa linha de pesquisa demonstra como os conceitos teóricos de emparelhamento, discutidos em nosso trabalho, são instrumentalizados para resolver problemas complexos de privacidade de dados e filtragem de restrições.

Contemporaneamente, o algoritmo de Micali e Vazirani continua sendo referência central para problemas de emparelhamento em grafos gerais. Peterson e Loui [11] oferecem uma exposição clara e rigorosa deste algoritmo, que opera em tempo $O(\sqrt{|V|}\cdot|E|)$ e permanece como o algoritmo sequencial mais eficiente conhecido para emparelhamento de cardinalidade máxima. A importância deste trabalho vai além da implementação: ele estabelece as bases teóricas que permitem a paralelização e distribuição de algoritmos de emparelhamento. Compreender profundamente este algoritmo é fundamental para estudantes que buscam avançar para domínios mais complexos de otimização combinatória, pois suas técnicas de tratamento de ciclos ímpares inspiraram desenvolvimentos posteriores em algoritmos distribuídos.

Expandindo a abordagem de Micali e Vazirani para ambientes distribuídos, o trabalho de Huang e Su [7] apresenta um algoritmo polinomial $\text{poly}(1/\varepsilon,\log n)$-round para obter uma aproximação $(1-\varepsilon)$ do emparelhamento máximo ponderado em grafos gerais no modelo CONGEST distribuído. Este avanço resolve um problema em aberto de longa data na área de algoritmos distribuídos, generalizando resultados prévios que funcionavam apenas em classes especiais de grafos (bipartidos e grafos livres de menores). A contribuição de Huang e Su demonstra que a estrutura de obstrução de Tutte permanece relevante e pode ser explorada de forma eficiente mesmo em cenários distribuídos, onde a comunicação entre processadores é limitada.

Paralelamente à evolução técnica, a transposição didática desses conceitos complexos tem sido objeto de estudo recente. Lassance et al. [10] argumentam que a barreira de entrada para o entendimento de grafos gerais não é puramente matemática, mas estrutural. O trabalho deles propõe uma reorganização curricular onde a apresentação de teoremas avançados deve ser precedida por uma construção visual rigorosa. Inspirados por essa metodologia, nosso artigo adota a premissa de que a visualização de "obstáculos" — como as componentes ímpares em Tutte — deve ser o ponto de partida do processo de ensino.

Diferentemente dos trabalhos existentes, que priorizam a otimização de desempenho algorítmico em cenários específicos ou a complexidade em sistemas distribuídos, este artigo contribui ao oferecer uma unificação didática entre a Teoria dos Grafos e a Teoria da Ordem. Nossa contribuição reside na sistematização da técnica de redução — especificamente na conversão entre Posets e Emparelhamentos — e na formalização de uma narrativa visual para o Teorema de Tutte. Ao focar na desmistificação dos obstáculos estruturais por meio de provas assistidas por diagramas, este trabalho preenche a lacuna entre o rigor matemático puro e a intuição necessária para o domínio da disciplina por estudantes de graduação.

Com o alicerce histórico referenciado e as conexões com a algoritmia moderna e a pedagogia estabelecidas, torna-se imperativo formalizar o desafio matemático. A seção a seguir delimita o escopo do nosso estudo, transpondo a intuição discutida nestes trabalhos relacionados para uma definição rigorosa de otimização combinatória.

IV. Descrição do Problema

O desafio central abordado neste trabalho é o Problema do Emparelhamento Máximo, fundamental na otimização combinatória. Formalmente, dado um grafo $G=(V,E)$, buscamos identificar um subconjunto de arestas $M\subseteq E$ tal que nenhuma aresta em M compartilhe um vértice comum. Esta propriedade é conhecida como arestas par-a-par disjuntas. O objetivo é maximizar a cardinalidade $|M|$, ou seja, encontrar a configuração que envolva o maior número possível de vértices e minimize vértices não emparelhados. A complexidade computacional para solucionar este problema varia conforme a topologia do grafo. Para grafos bipartidos, algoritmos exatos como o de Hopcroft-Karp [6] operam com alta eficiência em tempo $O(E\sqrt{V})$. Entretanto, em grafos gerais, a ausência de uma bipartição clara permite a existência de estruturas mais rígidas. Isso exige abordagens mais sofisticadas, como o algoritmo de Edmonds (Blossom) [4], para tratar ciclos ímpares.

Para concretizar a distinção estrutural entre essas classes de grafos e motivar a necessidade do Teorema de Tutte [14], propomos a análise de um cenário lúdico denominado "O Baile da UFT". Inicialmente, observamos o caso restrito ilustrado na Figura 8, que representa o Cenário A. Neste

 ISSN: 2675-3588

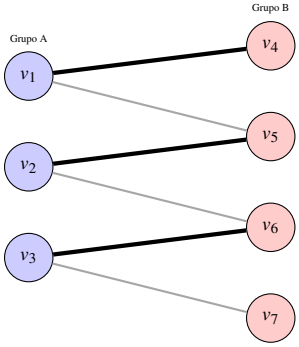


Figura 8: Cenário A: Grafo Bipartido. Os vértices v_i representam alunos. O Grupo A só dança com o Grupo B. As arestas marcadas (emparelhamento) são o resultado máximo.

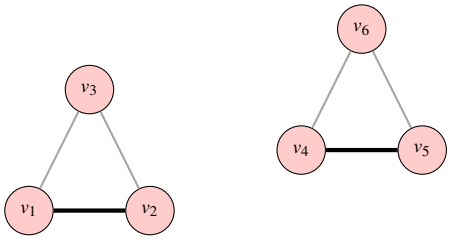


Figura 9: Cenário B: Grafo Geral. Os ciclos ímpares (triângulos) impedem um emparelhamento perfeito. As arestas marcadas mostram o emparelhamento máximo possível, deixando v_3 e v_6 sem par.

grafo bipartido, as regras de interação são estritas: alunos do Grupo A (nós azuis) só podem formar pares com alunos do Grupo B (nós vermelhos). A ausência de arestas internas em cada grupo simplifica a busca pelo emparelhamento máximo, pois não há conflitos de paridade interna a serem resolvidos.

Por outro lado, a complexidade aumenta consideravelmente no Cenário B, apresentado na Figura 9. Aqui, temos um grafo geral onde a regra de formação de pares baseia-se na afinidade, independentemente do grupo de origem. Essa flexibilidade permite a formação de ciclos ímpares, como o triângulo formado pelos vértices v_1, v_2 e v_3. Como pode ser visualizado na figura, se três indivíduos desejam formar pares exclusivamente entre si, é matematicamente impossível que todos sejam atendidos simultaneamente. Portanto, inevitavelmente, um vértice restará sem par. Esta ocorrência do ciclo ímpar é a representação geométrica do obstáculo que impede o emparelhamento perfeito em grafos não-bipartidos.

É neste contexto de impossibilidade estrutural que o Teorema de Tutte [14] se insere, oferecendo uma condição necessária e suficiente baseada na topologia global. Antes de avançarmos para as demonstrações formais, sistematizamos nas Tabelas 1 e 2 os principais problemas abordados, separando a análise de existência da análise de otimização de ordem.

A lógica de resolução apresentada nas tabelas desdobra-se em uma narrativa contínua que fundamenta as provas subsequentes. Inicialmente, na Análise de Paridade (Tutte), a prova estabelece que componentes com número ímpar de vértices possuem uma limitação aritmética inerente, pois nunca podem ser totalmente emparelhados internamente. Assim, cada componente ímpar exige uma conexão com um vértice externo (do conjunto S), de modo que, se o número de componentes ímpares $o(G-S)$ for maior que o número de vértices disponíveis em S, o emparelhamento perfeito torna-

TABELA 1: RESUMO ESTRUTURAL: TEOREMA DE TUTTE

Problema	Existência de Emparelhamento Perfeito (Tutte)
Input	Grafo Geral $G = (V,E)$ e a análise de subconjuntos de vértices removidos S.
Output	Condição Necessária e Suficiente: $o(G-S) \leq \|S\|$.
Resumo	Demonstração baseada na identificação de componentes ímpares como obstáculos estruturais intransponíveis.

TABELA 2: RESUMO ESTRUTURAL: TEOREMA DE DILWORTH

Problema	Decomposição Mínima de Cadeias (Dilworth)
Input	Poset P e sua transformação em Grafo Bipartido.
Output	Teorema Min-Max: tamanho máx. anti-cadeia = mínimo de cadeias.
Resumo	Técnica de Redução: converte a dependência de ordem em um problema de emparelhamento bipartido.

se impossível. Sequencialmente, no que tange à Técnica de Redução (Dilworth), a prova utiliza a construção de um grafo auxiliar para traduzir um conceito abstrato de ordem parcial em um geométrico de arestas. Ao duplicar os vértices do Poset para criar um grafo bipartido, demonstramos que cada aresta do emparelhamento conecta o fim de uma cadeia ao início de outra, resultando na identidade fundamental onde minimizar o número de cadeias é matematicamente equivalente a maximizar o emparelhamento ($|C| = n - |M|$).

Estabelecida a intuição visual de que ciclos ímpares impedem o emparelhamento perfeito e como a redução simplifica problemas de ordem, torna-se imperativo formalizar esses conceitos na próxima seção.

V. DEMONSTRAÇÃO E CONTRIBUIÇÕES

Nesta seção, apresentamos as demonstrações dos dois resultados fundamentais: o Teorema de Tutte [14] e o Teorema de Dilworth [3]. A escolha destes resultados visa expandir o repertório para além das técnicas básicas de caminhos aumentantes, introduzindo o conceito de "obstáculos estruturais". Iniciamos essa análise expandindo o escopo dos grafos bipartidos para os grafos gerais. Enquanto o Teorema de Hall [5] verifica vizinhanças locais, o Teorema de Tutte fornece uma condição global.

Condição de Tutte: Um grafo $G = (V,E)$ possui um emparelhamento perfeito se, e somente se, para todo subconjunto de vértices $S \subseteq V$, vale a desigualdade $o(G-S) \leq |S|$. Aqui, $o(G-S)$ representa o número de componentes conexos com um número ímpar de vértices no grafo resultante da remoção de S.

Partimos da hipótese de que o grafo G possui um emparelhamento perfeito M. Seja S um subconjunto qualquer de vértices removidos e considere as componentes conexas $C_1, C_2, \ldots, C_k$ resultantes dessa remoção. Ao analisarmos uma componente ímpar específica C_i, observamos que a soma das arestas internas não cobre a totalidade dos vértices,

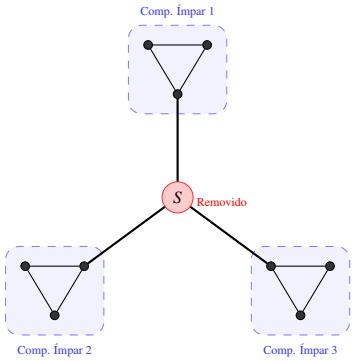


Figura 10: Visualização da falha na condição de Tutte: $o(G-S) > |S|$. Ao retirar o vértice central, restam 3 componentes ímpares isolados.

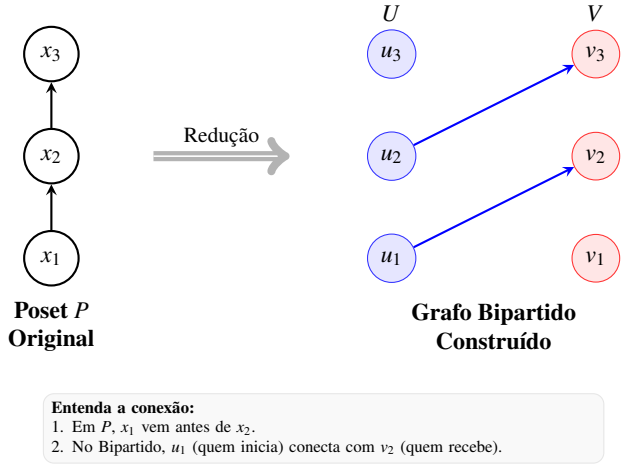


Figura 11: Visualização da Redução: duplicamos os vértices para separar as funções de "antecessor" (U) e "sucessor" (V).

devido à sua cardinalidade ímpar. Obrigatoriamente, pelo menos uma aresta de M deve conectar um vértice de C_i a um elemento externo.

Como C_i é uma componente isolada em $G-S$, não existem arestas ligando-a a outras componentes; logo, essa conexão externa deve necessariamente incidir em um vértice de S. Sob a ótica da contabilidade de arestas, visto que os elementos do emparelhamento são disjuntos, cada componente ímpar consome um vértice exclusivo de S. A conclusão lógica, ilustrada na Figura 10, é que se existem k componentes ímpares, serão necessários no mínimo k vértices distintos em S, demonstrando que $|S| \geq o(G-S)$.

Demonstração da Suficiência ($\Leftarrow$): A prova de que a validade da condição garante o emparelhamento é construída por redução ao absurdo. Inicialmente, supomos que a condição de Tutte [14] é verdadeira para o grafo G (ou seja, $o(G-S) \leq |S|$ para todo subconjunto S), mas, contraditoriamente, G não possui um emparelhamento perfeito.

Para explorar as falhas dessa suposição, maximizamos a estrutura do grafo adicionando arestas fictícias até formarmos um grafo G^*, que representa o limite máximo de conexões possível sem que se crie um emparelhamento perfeito. Vale ressaltar que, se encontrarmos uma violação da condição de Tutte em G^*, ela prova a falha no grafo original. Neste grafo saturado G^*, definimos S como o conjunto de vértices universais. A análise do resultado revela que a quantidade de componentes ímpares geradas pela remoção de S supera o número de vértices disponíveis em S. Isso implica diretamente que $o(G^*-S) > |S|$, um resultado que viola a nossa hipótese inicial.

Além da análise de existência proposta por Tutte, a teoria dos grafos se conecta diretamente à Teoria da Ordem. Essa relação é fundamental para a otimização combinatória, pois permite tratar problemas de ordenação sob uma ótica algorítmica eficiente. Frequentemente, problemas práticos de agendamento, hierarquia e dependência de tarefas — modelados matematicamente como Conjuntos Parcialmente Ordenados (Posets) — não aparentam, à primeira vista, possuir relação direta com a geometria de vértices e arestas. No entanto, o Teorema de Dilworth [3] é o exemplo central dessa relação, estabelecendo um vínculo formal entre estruturas de ordem e grafos. Essa equivalência com o emparelhamento bipartido foi explorada por Kameda e Munro [8] para o desenvolvimento de algoritmos. Ao provarem que a decomposição de conjuntos ordenados pode ser reduzida ao problema de emparelhamento, eles viabilizaram o uso de soluções polinomiais eficientes para resolver problemas de ordenação.

Dilworth: O número mínimo de cadeias necessárias para cobrir todos os elementos de um conjunto parcialmente ordenado P é igual ao tamanho máximo de uma anticadeia em P.

Para demonstrar este teorema e sua aplicação computacional, utilizamos a técnica de Redução, transformando o problema de "Posets" em "Casamento Bipartido". A Figura 11 detalha visualmente essa transformação. O processo consiste em tomar os elementos do conjunto ordenado $P = \{x_1, \ldots, x_n\}$ e duplicá-los para criar um grafo bipartido $G = (U \cup V, E)$. Conforme detalhado na Figura 11, o lado U (nós azuis) representa os elementos atuando como "início de uma relação", enquanto o lado V (nós vermelhos) representa os mesmos elementos como "fim". A regra de construção é direta: desenhamos uma aresta direcionada (u_i, v_j) se, e somente se, o elemento x_i precede o elemento x_j na ordem original.

A validade desta construção reside em demonstrar uma equivalência estrutural estrita, da qual existe uma bijeção entre um emparelhamento válido em G e uma decomposição em cadeias em P. Observe que a definição de emparelhamento exige que arestas sejam disjuntas, ou seja, cada vértice tenha grau no máximo 1. No contexto do poset, isso traduz-se na regra de linearidade das cadeias: um elemento x_i não pode ter múltiplos sucessores imediatos (o que violaria o grau em U) nem múltiplos antecessores imediatos (o que violaria o grau em V). Portanto, um conjunto de arestas é um emparelhamento se, e somente se, ele une elementos formando sequências lineares válidas e disjuntas.

Algebraicamente, a prova se estabelece pela contagem de componentes. Iniciamos com n cadeias triviais (cada elemento isolado). Cada aresta (u_i, v_j) adicionada ao emparelhamento junta o final de uma cadeia ao início de outra, reduzindo o número total de cadeias em exatamente uma unidade. Assim, estabelecemos a identidade fundamental: o número de cadeias $|\mathscr{C}|$ resultantes de um emparelhamento M é dado por $|\mathscr{C}| = n - |M|$. Para minimizar o número de cadeias $|\mathscr{C}|$, o objetivo de Dilworth, somos matematicamente forçados a maximizar $|M|$. Desta forma, demonstramos que

ISSN: 2675-3588

encontrar a Decomposição Mínima de Cadeias é equivalente à busca pelo Emparelhamento Máximo no grafo bipartido. Como cada aresta do emparelhamento une dois vértices em uma mesma cadeia, o número mínimo de cadeias será igual ao número total de vértices subtraído da cardinalidade deste emparelhamento máximo.

Com a validação desta equivalência e o encerramento das demonstrações formais, finalizamos a análise técnica das estruturas propostas. Torna-se pertinente, então, avaliar não apenas a corretude matemática dos teoremas apresentados, mas também o impacto pedagógico que a construção dessas provas exerce sobre o domínio da disciplina.

VI. Resultados e Reflexões

A elaboração deste artigo cumpriu o papel de consolidar o entendimento sobre teoremas fundamentais da Teoria dos Grafos, funcionando como uma ferramenta de fixação de conceitos matemáticos complexos para estudantes de Ciência da Computação. Ao explorarmos as provas de Tutte [14] e Dilworth [3], os resultados apontam para a eficácia de uma abordagem que privilegia a intuição geométrica aliada ao rigor formal, facilitando a assimilação de estruturas abstratas. A análise desenvolvida confirma que o domínio de grafos gerais é significativamente ampliado quando se adota um pensamento estrutural e holístico.

Diferente da verificação local de vizinhanças — comum no estudo introdutório do Teorema de Hall [5] — a compreensão de Tutte beneficia-se imensamente da visualização global do grafo e de suas componentes conexas. O uso de metáforas visuais, onde o conjunto S é representado como um "hub" ou ponto de articulação, mostrou-se uma estratégia pedagógica poderosa para tangibilizar a abstração da remoção de vértices ($G - S$). Essa representação concreta permite que o estudante visualize imediatamente como a estrutura se fragmenta, tornando a condição de paridade uma consequência lógica observável, e não apenas uma regra algébrica.

Da mesma forma, a exploração do Teorema de Dilworth [3] revelou-se uma oportunidade excelente para clarificar a dualidade Min-Max. A abordagem adotada permitiu demonstrar positivamente como a adição de arestas (emparelhamento) atua como um mecanismo de otimização que funde cadeias disparatadas, oferecendo uma intuição robusta sobre como problemas de ordem podem ser resolvidos eficientemente através de grafos. Essa transposição didática resulta no desenvolvimento de competências críticas, como a modelagem por redução, onde o discente aprende a converter um problema de ordem parcial em um desafio geométrico de emparelhamento bipartido. Essa capacidade de simplificação topológica é um ganho conceitual que prepara o aluno para enfrentar problemas de alta complexidade em disciplinas como Teoria da Computação e Análise de Algoritmos, onde a técnica de redução é o alicerce para o entendimento de classes de complexidade.

Além disso, ao priorizar a centralidade teórica de Tutte [14] e Dilworth [3], este trabalho promove uma forte interdisciplinaridade ao conectar a Álgebra com a Algoritmia de Grafos. Demonstrou-se que problemas práticos de agendamento, hierarquia e dependência de tarefas — comuns em áreas como Sistemas Operacionais e Engenharia de Software — são, em sua essência, problemas de desenho estrutural. O exercício de construir provas matemáticas utilizando diagramas como parte do argumento lógico desenvolve no estudante um rigor demonstrativo visual, permitindo identificar gargalos em redes de forma intuitiva. Assim, o texto consolida-se como um material de apoio didático que transforma o rigor dos livros-texto em uma ponte acessível para o sucesso em disciplinas avançadas de Otimização Combinatória, validando a premissa de que a compreensão profunda da estrutura do problema é o primeiro passo para a eficiência algorítmica..

VII. Considerações Finais

O objetivo central deste trabalho foi revisitar os fundamentos teóricos do emparelhamento em grafos, transcendendo a abordagem tradicional focada apenas na execução de algoritmos. Buscou-se preencher a lacuna didática existente entre o entendimento intuitivo de grafos bipartidos e a complexidade abstrata dos grafos gerais e estruturas de ordem. Ao analisar as demonstrações clássicas, o artigo propôs uma narrativa visual que facilita a assimilação de conceitos difíceis por estudantes de graduação.

A síntese dos resultados aponta para duas conclusões teóricas maiores. Primeiramente, na análise de existência, entendemos que o Teorema de Tutte é uma fórmula abrangente, que explica tanto os casos simples (bipartidos) quanto os complicados (gerais), olhando para a estrutura completa do grafo. Demonstramos que a "barreira de paridade" (componentes ímpares isolados) é o mecanismo universal de obstrução, englobando os casos restritos de Hall [5] e König [9]. Em segundo lugar, a exploração do Teorema de Dilworth [3] ratificou a equivalência estrita entre problemas de ordenação (posets) e o emparelhamento bipartido. Essa conexão provou que a complexidade de problemas de agendamento pode ser reduzida polinomialmente, validando as estratégias algorítmicas de Kameda e Munro [8].

No que tange às contribuições pedagógicas, este estudo oferece uma metodologia de ensino baseada na visualização de "obstáculos estruturais". A principal contribuição reside na formalização da técnica de Redução, ao invés de apenas apresentar o Teorema de Dilworth como uma fórmula, detalhamos o processo de transformação topológica que converte um problema desconhecido em um conhecido. Além disso, o uso de analogias concretas fornece aos estudantes um vocabulário visual para identificar falhas de emparelhamento, superando a dificuldade comum de visualizar a remoção de conjuntos arbitrários.

Como limitação, este estudo concentrou-se nas condições de existência e nas provas estruturais, sem aprofundar-se na implementação computacional dos algoritmos de busca, como o *Blossom* de Edmonds. Como perspectiva para trabalhos futuros, sugere-se a expansão desta base didática para o domínio da implementação computacional interativa. A criação de ferramentas de software que permitam a visualização dinâmica dos algoritmos — especificamente a simulação passo a passo da contração de ciclos ímpares no algoritmo *Blossom* de Edmonds — representaria um avanço significativo. Ao utilizar as mesmas metáforas de “fusão de componentes” e “gargalos” estabelecidas neste artigo, seria possível demonstrar aos estudantes não apenas

por que o emparelhamento falha sob a ótica teórica, mas *como* o computador manipula e reduz essas estruturas em tempo de execução. Encerramos, portanto, com a convicção de que a união entre a base matemática sólida e a visualização intuitiva se mostrou uma estratégia de ensino extremamente eficaz. Acreditamos que essa abordagem facilita significativamente o aprendizado, permitindo que os estudantes compreendam o conteúdo com muito mais clareza e segurança.

REFERÊNCIAS

[1] Claude Berge. Two theorems in graph theory. *Proceedings of the National Academy of Sciences of the United States of America*, 43(9):842–844, 1957.

[2] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications*. Elsevier, New York, 1976.

[3] R. P. Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics*, pages 161–166, 1950.

[4] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[5] P. Hall. On representatives of subsets. *Journal of the London Mathematical Society*, 10:26–30, 1935.

[6] John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.

[7] Shang-En Huang and Hsin-Hao Su. (1-ε)-Approximate Maximum Weighted Matching in Poly(1/ε, log n) Time in the Distributed and Parallel Settings. *Proceedings of the 2023 ACM Symposium on Principles of Distributed Computing*, pages 42–52, 2023.

[8] T. Kameda and I. Munro. A o(|v|·|e|) algorithm for maximum matching of graphs. *Computing*, 12(1):91–98, 1974.

[9] D. König. Grafok es matrixok. *Matematikai és Fizikai Lapok*, 38:116–119, 1931.

[10] Y. M. Lassance Di Vilhena y Cantañede, G. B. Bianchini, and T. D. Dos Santos. Reflexões e práticas pedagógicas no escopo da disciplina de teoria da computação. *Academic Journal on Computing, Engineering and Applied Mathematics*, 6(2):10–17, 2025.

[11] Paul A. Peterson and Michael C. Loui. The general maximum matching algorithm of Micali and Vazirani. *Algorithmica*, 3:511–533, 1988.

[12] Jean-Charles Régin. A filtering algorithm for constraints of difference in csps. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI)*, pages 362–367, 1994.

[13] Tamir Tassa. Addendum to "finding all maximally-matchable edges in a bipartite graph". *Theoretical Computer Science*, 491:136, 2013.

[14] W. T. Tutte. The factorization of linear graphs. *Journal of the London Mathematical Society*, 22(2):107–111, 1947.

 ISSN: 2675-3588

Teoria dos Grafos: O Problema de Coloração de Arestas em Grafos

Graph Theory: The Edge Coloring Problem in Graphs

Lean de Albuquerque Pereira[1], Tiago Castro Barbosa[1], Daniel Martins da Silva[1] e Tanilson Dias dos Santos[1]

[1] *Universidade Federal do Tocantins, Ciência da Computação, Tocantins, Brasil*

Data de recebimento do manuscrito: 03/12/2025
Data de aceitação do manuscrito: 30/01/2026
Data de publicação: 10/02/2026

Resumo—A teoria dos grafos, impulsionada historicamente pela conjectura de Francis Guthrie em 1852 e pela subsequente prova do Teorema das Quatro Cores, evoluiu de curiosidades topológicas para ferramentas essenciais de modelagem. Este trabalho foca especificamente no Problema de Coloração de Arestas, abordando-o sob uma perspectiva histórica e rigorosamente formal. Inicialmente, o texto contextualiza a transição dos problemas de coloração de mapas para a coloração de arestas, destacando sua relevância prática em otimização de redes e agendamento. O núcleo da discussão aprofunda-se na análise do Teorema de Vizing, que estabelece limites precisos para o índice cromático de grafos simples, situando-o entre o grau máximo e o grau máximo acrescido de uma unidade. Serão dissecados os principais lemas e as condições estruturais que determinam se um grafo pertence à Classe 1 ou Classe 2. Ao explorar a complexidade inerente a essa classificação, o artigo serve como uma referência pedagógica, elucidando como restrições locais de adjacência ditam o comportamento global em sistemas complexos.

Palavras-chave—Coloração de Grafos, Problema das Quatro Cores, Teorema de Vizing, Otimização Combinatória, Modelagem.

Abstract—*Graph theory, historically propelled by Francis Guthrie's 1852 conjecture and the eventual proof of the Four Color Theorem, has evolved from a collection of topological curiosities into a set of essential modeling tools. This work specifically targets the Edge Coloring Problem, addressing it through a lens that is both historical and rigorously formal. Initially, the text contextualizes the conceptual shift from map coloring to edge coloring, emphasizing its practical applicability in critical areas such as network optimization and scheduling. The core discussion deepens into an analysis of Vizing's Theorem, which establishes precise boundaries for the chromatic index of simple graphs, positioning it strictly between the maximum degree and the maximum degree plus one. Key lemmas and structural conditions determining whether a graph falls into Class 1 or Class 2 are dissected. By exploring the inherent complexity of this classification, this article serves as a pedagogical reference, clarifying how local adjacency constraints dictate global behavior in complex systems.*

Keywords—*Graph Coloring, Four Color Problem, Vizing's Theorem, Combinatorial Optimization, Modeling.*

I. Introdução

A Teoria dos Grafos é uma ferramenta de modelagem versátil, oriunda da matemática, mas de escopo fundamental para a ciência da computação. Sua capacidade de representar e modelar relações complexas em sistemas diversos – desde redes neurais e clusters de computadores até a otimização logística de trabalhadores e rotas aéreas – a torna fascinante e diretamente aplicável a problemas cotidianos. Ao traduzir situações reais para uma linguagem matemática precisa, os grafos permitem abstrair a complexidade do mundo físico, revelando a estrutura lógica subjacente aos problemas de conexão e conflito.

Nesse contexto, o estudo de coloração em grafos remonta ao século XIX, originado pelo que pode ser visto como o problema gerador da área: o famoso "Problema das Quatro Cores". A história inicia-se em 1852 com o matemático e botânico sul-africano Francis Guthrie. Ao tentar colorir mapas de condados da Inglaterra, Guthrie observou que talvez fosse possível colorir qualquer mapa plano utilizando apenas quatro cores, de modo que regiões vizinhas não compartilhassem a mesma cor. Embora a conjectura tenha sido formulada em correspondências privadas naquela época, ela foi formalmente apresentada à comunidade científica por Cayley em 1879 [1] e discutida pelo próprio Guthrie em nota posterior [2]. A curiosidade inicial deflagrou uma das mais longas e produtivas buscas por uma prova matemática.

Dados de contato: Lean de Albuquerque Pereira, lean.albuquerque@uft.edu.br

A primeira prova da conjectura surgiu apenas em 1879, apresentada pelo matemático inglês Alfred Kempe [3]. Aceita por uma década, a demonstração foi refutada em 1890, quando erros estruturais foram encontrados. Diversas soluções foram propostas subsequentemente, mas a confirmação definitiva da conjectura ocorreu somente em 1976, pelos matemáticos Kenneth Appel e Wolfgang Haken, da Universidade de Illinois [4, 5]. Contudo, parte dessa prova utilizava computadores para verificar milhares de casos, fato que gerou resistência na comunidade matemática da época, que ansiava por uma demonstração puramente analítica.

Embora o "Problema das Quatro Cores" trate essencialmente da coloração de vértices (ou faces), ele pavimentou o caminho para variantes igualmente profundas, como o Problema de Coloração de Arestas, foco central deste trabalho. Diferente de colorir regiões, colorir arestas busca atribuir rótulos às conexões de um grafo de tal forma que arestas incidentes a um mesmo vértice não compartilhem a mesma cor. Esse tipo de modelagem é vital para cenários onde o conflito não está nos objetos (vértices), mas na utilização simultânea de canais de comunicação ou horários, sendo o índice cromático o parâmetro que define a eficiência máxima dessa alocação.

Para além do panorama histórico internacional, aparece com prestígio também a contribuição brasileira no desenvolvimento da Teoria dos Grafos. O Brasil consolidou-se como um pólo de excelência mundial nesta área, impulsionado por pesquisadores cujos trabalhos são referência na literatura contemporânea. Dentre eles, destacam-se as contribuições de Jayme Luiz Szwarcfiter [6], fundamental na estruturação da pesquisa em algoritmos e grafos no país; Cláudio L. Lucchesi [7], renomado por seus trabalhos seminais, incluindo o célebre Teorema de Lucchesi-Younger em grafos direcionados; e Nelson Maculan [8], uma referência global em otimização combinatória. Contextualizar o problema de coloração de arestas envolve, portanto, reconhecer essa robusta tradição acadêmica nacional que alia rigor teórico a aplicações computacionais de ponta.

Neste artigo queremos portanto demonstrar de forma pedagógica o problema de coloração de arestas, assegurando ao leitor compreender a evolução desses conceitos, culminando na análise de dois pilares teóricos fundamentais, o Teorema de Kőnig [9], que soluciona o problema para grafos bipartidos relacionando-o ao grau máximo, e o Teorema de Vizing [10], que estabelece os limites estritos para grafos simples. Ao detalhar essas condições, busca-se absorver a robustez matemática que sustenta a classificação dos grafos e suas aplicações contemporâneas.

A seguir abordaremos o tópico por entre quatro seções subsequentes. A *Seção II* estabelece as definições preliminares e a notação fundamental, introduzindo conceitos estruturais como grau máximo e emparelhamento. Na *Seção III*, exploramos a natureza do problema, discutindo intuitivamente os limites cromáticos e apresentando os lemas auxiliares de Bondy e Murty que fundamentam a otimização de cores. A *Seção IV* é dedicada à demonstração formal dos dois pilares da teoria: o Teorema de Kőnig para grafos bipartidos e o Teorema de Vizing para grafos simples. Por fim, a *Seção V* apresenta as conclusões e uma síntese dos resultados obtidos.

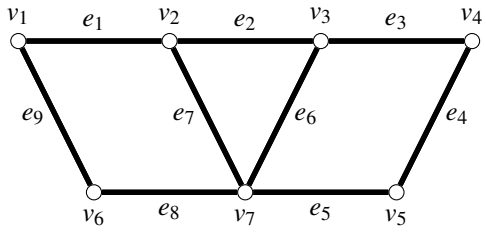


Figura 1: Representação Gráfica do Grafo G_1.

II. PRELIMINARES

Para compreender a profundidade do Problema de Coloração de Arestas, é necessário primeiro estabelecer a linguagem comum da Teoria dos Grafos. Nesta seção, definimos as estruturas fundamentais, as propriedades de conectividade e os parâmetros que governam a complexidade desses sistemas. As definições foram extraídas de [6, 11].

Formalmente, um *grafo* $G = (V,E)$ é uma estrutura matemática composta por dois conjuntos fundamentais: um conjunto não vazio V de *vértices* e um conjunto E de pares não-ordenados de vértices, denominados *arestas*. Denotamos uma aresta qualquer e, como $e = (a,b)$, onde a e b são vértices do grafo e dizemos que a e b são *extremos*(ou *extremidades*) da aresta e. Ainda, a aresta e é dita *incidente* aos vértices a e b [6].

Neste contexto, os vértices (V) representam os objetos ou entidades do sistema, como computadores, pessoas ou interseções, enquanto as arestas (E) representam as conexões ou relações diretas entre esses objetos.

Um grafo qualquer, digamos G_1, pode ser representado de várias maneiras, por exemplo, de forma geométrica como pode ser visto na *Figura 1*. Cada vértice é simbolizado com um círculo, e os segmentos de retas que os conectam são as arestas do grafo. Denotamos como $V(G)$ e $E(G)$ o conjunto de vértices e arestas do grafo G, respectivamente. Por exemplo, para o grafo G_1, denotamos sua estrutura como:

$$V(G_1) = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$$
$$E(G_1) = \{(v_1,v_2),(v_2,v_3),(v_3,v_4),(v_4,v_5),(v_6,v_1),$$
$$(v_2,v_7),(v_3,v_7),(v_5,v_7),(v_6,v_7)\}$$

Dois vértices são *adjacentes* (ou vizinhos) se existe uma aresta que incide em ambos os vértices. Analogamente, duas arestas são *adjacentes* se possuem uma extremidade em comum [6].

Por exemplo, em G_1, v_1 e v_2 são vértices adjacentes, pois existe uma aresta que incide (conecta) ambos os vértices: $e_1 = (v_1,v_2)$. Por outro lado, os vértices v_1 e v_5 não são adjacentes, uma vez que não existe aresta que os conectam. Similarmente, considerando a aresta e_2, temos que e_1 e e_2 são adjacentes, pois possuem uma extremidade em comum (v_2). No entanto, as arestas e_1 e e_4 não são adjacentes.

Seguindo as definições clássicas de Bondy e Murty [11], estabelecemos duas propriedades essenciais para o escopo deste trabalho. Primeiramente, um grafo G é dito *finito* se o seu conjunto de vértices e arestas é finito. Em segundo lugar, um grafo é classificado como *simples* se ele não possui *laços* (uma aresta com início e fim no mesmo vértice) e não possui duas ou mais arestas que incidem no mesmo par de vértices

ISSN: 2675-3588

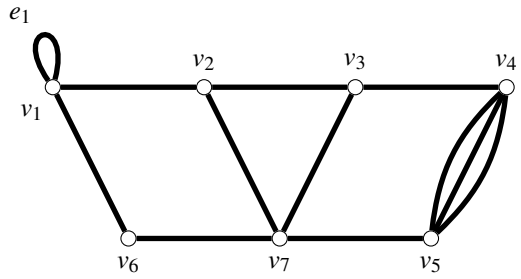


Figura 2: Grafo G_2 (exemplo de grafo não simples).

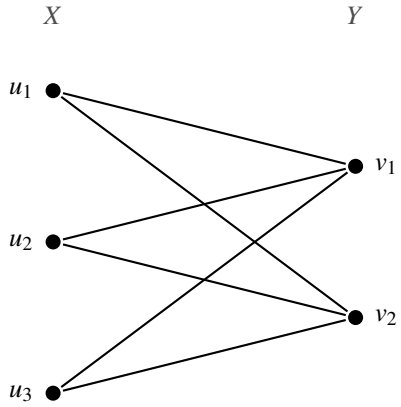


Figura 3: Grafo bipartido $K_{3,2}$ ilustrando a partição de vértices em X e Y.

(arestas múltiplas). Por exemplo, o grafo G_2 (Figura 2) não é simples, pois possui três arestas que conectam v_4 e v_5 e ainda possui um laço, representado pela aresta e_1. Por outro lado, o grafo G_1 é simples.

Um grafo $G = (V,E)$ é dito *bipartido* se o seu conjunto de vértices V pode ser particionado em dois subconjuntos disjuntos, digamos X e Y, de tal forma que toda aresta de G conecta um vértice de X a um vértice de Y. Consequentemente, não existem arestas com ambas as extremidades no mesmo subconjunto. A Figura 3 exemplifica esta propriedade através do grafo completo $K_{3,2}$.

Um dos conceitos mais críticos para problemas de coloração é o *grau* de um vértice. O grau de um vértice v, denotado por $d(v)$, é definido como o número de arestas incidentes a ele. Na *Figura 4*, destacamos dois exemplos importantes: o vértice v_1 possui apenas duas arestas incidentes (destacadas em azul), logo $d(v_1) = 2$; já o vértice v_7 comporta-se como o elemento de maior conectividade (destacadas em vermelho). A partir dessa definição local, derivamos o parâmetro global mais importante para este trabalho: o *Grau Máximo* ($\Delta(G)$). Ele representa o maior valor de grau encontrado entre todos os vértices. No nosso exemplo, como nenhum vértice supera v_7, temos que $\Delta(G_1) = 4$.

Em um grafo definimos *Caminho* (P_n) como sendo uma sequência de vértices adjacentes sem repetição. Na Figura 5, a sequência $v_1v_6v_7v_5$ (destacada em *azul*) constitui um caminho válido (P_4), conectando o vértice v_1 ao v_5 através do interior do grafo. Em contrapartida, a sequência $v_1v_2v_1v_6$ não forma um caminho, pois o vértice v_1 se repete. Um *Ciclo* (C_n) consiste em um caminho cujo vértice de início é igual ao vértice de fim, fechando um circuito. O destaque em *vermelho* exemplifica um ciclo C_3, de tamanho 3 (um triângulo): $v_2v_3v_7v_2$. [6]

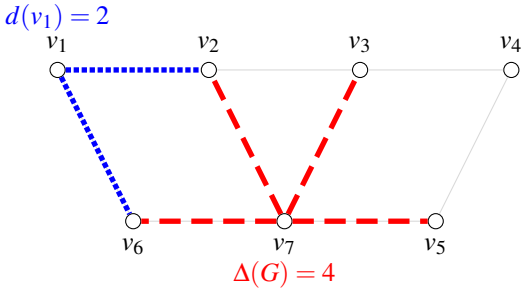


Figura 4: Visualização dos graus do grafo G_1.

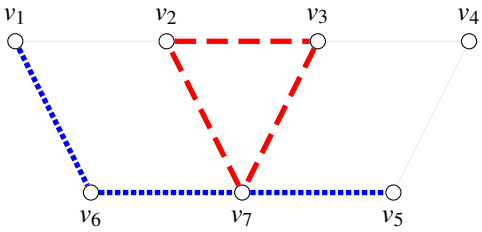


Figura 5: Exemplos de subestruturas em G_1: um Caminho aberto (azul) e um Ciclo fechado (vermelho).

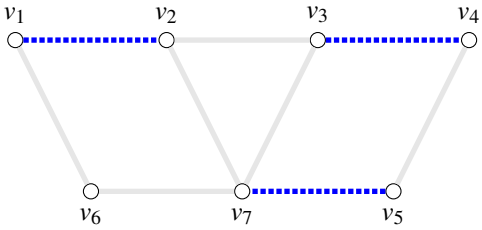


Figura 6: Um emparelhamento no grafo G_1 (em azul).

Um grafo G é denominado *conexo* quando existe caminho para cada par de vértices; do contrário, o grafo é dito ser *desconexo* [6]. Por exemplo, o grafo G_1 é conexo uma vez que para cada dois vértices quaisquer sempre existe um caminho que os conecta. Em contrapartida, considere que as arestas verdes da figura 6 juntamente com os seus vértices formem um grafo. Este, seria desconexo já que, por exemplo, não existe um caminho que conecta os vértices v_1 e v_7.

Finalmente, chegamos ao conceito de *emparelhamento* (*matching*). Um emparelhamento em um grafo G é um conjunto de arestas $M \subseteq E$ tal que nenhuma aresta de M é adjacente a outra; em outras palavras, nenhum vértice do grafo incide em mais de uma aresta de M [11].

Na *Figura 6*, destacamos em azul um emparelhamento formado pelas arestas $\{(v_1,v_2),(v_3,v_4),(v_5,v_7)\}$. Note a característica visual mais importante: essas três arestas são totalmente independentes e não compartilham nenhum vértice comum (elas "não se tocam"). Esse conceito é a base estrutural da coloração de arestas, pois em uma coloração válida, todas as arestas pintadas com uma mesma cor formam, obrigatoriamente, um emparelhamento.

III. Trabalhos Relacionados

A Teoria dos Grafos constitui uma importante área tanto no âmbito teórico e prático. No campo teórico sua importância é um reflexo da existência de muitos problemas ainda em estudo ou mesmo sem solução, o que incentiva a escrita de trabalhos acadêmicos na área e formação de

grupos de pesquisas na universidade [6]. Por outro lado, o tema é também extremamente relevante do ponto de vista prático com aplicações que surgem nas mais diversas áreas como na química (modelagem de estrutura de moléculas); no planejamento de rotas de tráfego aéreo com menor distância; na engenharia e obviamente na computação [12]. Portanto, são muitos os trabalhos que buscam contribuir pedagogicamente no ensino Teoria dos Grafos de formas mais acessível, haja vista sua importância prática e teórica.

Silva [12], em sua dissertação, tem como proposta de trabalho introduzir a Teoria dos Grafos no ensino fundamental. Para isso ele propõe uma abordagem evidentemente mais lúdica para assim motivar os alunos ao aprendizado, e escolhe problemas que sejam mais próximos ao codiano dos alunos como o problema de caminhos. Segundo o autor, o trabalho não só contribui para professores que desejam lecionar o conteúdo, mas também para qualquer pessoa que tem interesse no assunto.

Csóka, Lippner e Pikhurko [**?**], em seu estudo investigaram o problema de coloração de arestas em *Graphings*, segundo os autores: "Um graphing é uma generalização analítica de um grafo de grau limitado que aparece em várias áreas, como limites de grafos esparsos e teoria de equivalência de órbitas." Eles mostraram tanto o Teorema de König e o Teorema de Vizing poderiam ser generalizados para essa classe de grafos.

Em seu artigo Müller e Bayer [13] apresentam um possibilidade pedagógica para a abordagem de Teoria dos Grafos nos anos finais do ensino fundamental, através de um desafio lúdico adaptado por eles. Tal atividade além de divertir os alunos faz uma exposição branda sobre a estrutura de um grafo (vértices, arestas, grau) e conceitos relacionados como conexidade e planaridade.

Soares [14] em seu trabalho, apresenta três teoremas em Teoria dos Grafos e suas respectivas provas em detalhes e estruturadamente, com o intuito de encorajar a inclusão de Tópicos de Grafos no Ensino Médio. Os teoremas apresentados: Teorema das Cinco Cores, Teorema da Galeria de Arte, e Teorema da Amizade foram escolhidos ainda por possuírem um certo apelo estético a auxiliar na conclusão do objetivo de seu estudo.

Finalmente, um trabalho feito por Yasser e Bianchii [15] que, apesar de ser da área de Teoria da Computação, se propõe a fazer uma reflexão e discutir sobre Práticas Pedagógicas no escopo da disciplina de Teoria da Computação. Conforme os autores, o uso de abordagens alternativas como seminários, auxiliou na compreensão dos conceitos que são expostos tradicionalmente de maneira mais abstrata e gerou um maior índice de satisfação na disciplina.

Na próxima seção introduziremos definições e conceitos que serão utilizadas nesta pesquisa para abordar o Problema de Coloração de Arestas.

IV. Coloração de Arestas

Intuitivamente, como o próprio nome sugere, uma *coloração* de arestas consiste em atribuir k rótulos às arestas de um grafo qualquer onde cada rótulo pode ser interpretada como uma cor.

Formalmente, uma k-coloração de arestas de um grafo G sem laços, pode ser descrita não apenas como uma atribuição de rótulos, mas estruturalmente como uma partição do conjunto de arestas E em k subconjuntos $(E_1, E_2, \ldots, E_k)$. Desta forma, cada subconjunto E_i representa as arestas de uma mesma cor. Se as arestas de cada subconjunto E_i forem não adjacentes, dizemos que a coloração é *própria* [11]. Se um grafo G admite uma coloração própria com k cores, dizemos que *G é k-colorível*.

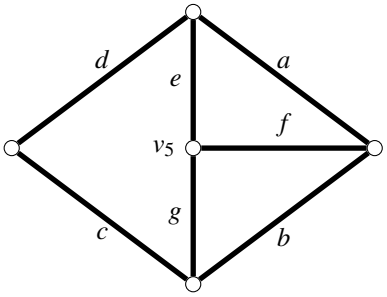


Figura 7: Grafo G_2 Fonte: Bondy e Murty(1976

Sob a ótica da Teoria dos Grafos, nota-se que esse conjunto de arestas independentes corresponde exatamente à definição de *emparelhamento* (*matching*) vista na seção anterior. Portanto, colorir as arestas de um grafo G equivale a particionar sua estrutura em uma coleção de emparelhamentos distintos $(M_1, M_2, \ldots, M_k)$.

Por exemplo, considere o grafo G_2(Figura 7). Podemos definir uma coloração $\mathscr{C} = (\{a,b,c,d\},\{e,f\},\{g\})$. Podemos interpretar essa partição da seguinte forma: as arestas a, b, c e d colorimos com uma cor qualquer, digamos c_1; as arestas e e f recebem a cor c_2 e a aresta g recebe a cor c_3. Obviamente essa coloração não é própria uma vez que existem arestas adjacentes que receberam a mesma cor(por exemplo, as arestas a, b, c, d).

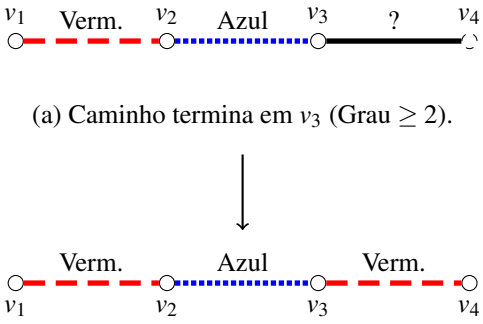
Por outro lado, considere a coloração $\mathscr{C}' = (\{a,g\},\{b,e\},\{c,f\},\{d\})$. Novamente, isso pode ser interpretado como uma atribuição de cores da seguinte forma: a arestas a e g recebem a cor c'_1; as arestas b e e recebem a cor c'_2; as arestas c e f recebem a cor c'_3 e a aresta d recebe a cor c'_4. Dessa vez, note que não existe arestas adjacentes com uma mesma cor. Portanto, $\mathscr{C}'$ é uma coloração própria e perceba que cada conjunto de arestas dessa partição forma um *emparelhamento*.

Dizemos ainda que uma determinada *cor c é representada em um vértice v, se existe alguma aresta incidente a v que possua a cor c*. Por exemplo, para o grafo G_1 e considerando a coloração $\mathscr{C}'$ as cores c_1, c_2, c_3 são representadas no vértice v_5, uma vez que, as arestas e, f e g, incidem em v_5 e possuem as cores c_1, c_2, c_3.

Dessa perspectiva, surge um questionamento natural: "Qual a menor quantidade de cores necessária para pintar as arestas deste grafo sem gerar conflitos de incidência?". A resposta define um dos parâmetros fundamentais da área: o *índice cromático*, denotado por $\chi'(G)$. Este parâmetro representa o número mínimo de emparelhamentos distintos necessários para cobrir todas as arestas de um grafo de forma válida. No exemplo do grafo G_2, o leitor pode conferir que 4 é o *menor números de cores possível para realizar uma coloração própria* em G_2. Portanto $\chi'(G_2) = 4$.

Ao buscarmos o índice cromático, deparamo-nos imediatamente com uma restrição física imposta pela própria estrutura do grafo. Considere o vértice mais "congestionado" do sistema, isto é, aquele que possui o maior número de conexões

 ISSN: 2675-3588



(a) Caminho termina em v_3 (Grau ≥ 2).

(b) Extensão: v_3 agora tem as duas cores.

Figura 8: Ilustração esquemática da propriedade de extensibilidade.

(o *grau máximo*, denotado por $\Delta(G)$).

A lógica é trivial: se olharmos novamente para o vértice v_7 da *Figura 1* (Seção 2), vemos que ele possui 4 conexões. É fisicamente impossível colorir essas 4 arestas incidentes com apenas 3 cores sem que duas delas compartilhem a mesma cor e causem um conflito. Esse "gargalo" local impõe, portanto, um limite inferior global para todo o sistema:

$$\chi'(G) \geq \Delta(G) \tag{1}$$

Essa desigualdade estabelece que são necessárias, *pelo menos*, tantas cores quanto o grau máximo. A questão central que a teoria busca responder é: será que esse mínimo é suficiente? Para alguns para alguns tipos de grafos, como os *grafos bipartidos*, a resposta é afirmativa, como veremos a seguir.

Para avançarmos da intuição para a prova formal de que grafos bipartidos atingem o limite inferior $\Delta(G)$, necessitamos de uma ferramenta auxiliar que garanta a distribuição equilibrada de cores. Bondy e Murty apresentam um resultado técnico fundamental, conhecido no livro como *Lema 6.1.1* [11]. Para fins didáticos, chamaremos este resultado de *Lema das Duas Cores*.

A intuição por trás deste lema é uma questão de paridade. Sabemos que ciclos ímpares são as únicas estruturas que impedem uma 2-coloração perfeita (onde arestas alternam cores). Se removermos essa restrição, ganhamos controle sobre a incidência de cores nos vértices.

> "*Lema das Duas Cores:* Seja G um grafo conexo que não é um ciclo ímpar. Então, G possui uma 2-coloração de arestas na qual ambas as cores estão representadas em cada vértice de grau pelo menos dois."

Para visualizar a ideia construtiva deste lema, imagine que nosso objetivo é traçar um caminho pelo grafo, pintando as arestas alternadamente em *Vermelho* e *Azul*, conforme ilustrado esquematicamente na *Figura 8*.

Ao passarmos por um vértice intermediário (como o vértice v_2 na Figura 8-a), necessariamente entramos por uma cor e saímos pela outra. Isso garante que v_2 possui ambas as cores representadas. O problema surge apenas nos vértices que habitam a *extremidade* do caminho (como o vértice v_3), pois eles estariam em contato com apenas uma aresta colorida neste trajeto.

A genialidade do lema reside na *extensibilidade*, demonstrada na parte (b) da Figura 8. Se o caminho termina em um vértice que ainda tem outras arestas não coloridas (ou seja, grau ≥ 2), podemos simplesmente *expandir* o caminho por essa nova aresta usando a cor alternada.

Podemos repetir esse processo até que o caminho termine em um vértice sem saída ou feche um ciclo. O lema garante que, exceto no caso específico do ciclo ímpar (onde a alternância de cores trava ao fechar o ciclo), sempre conseguimos ajustar caminhos para que nenhum vértice de grau igual ou maior que dois fique com uma cor só.

Além da existência de colorações, é importante definir uma forma para compará-las. Dada uma k-coloração $\mathscr{C}$ de G, denotamos por $c(v)$ o número de cores distintas representadas no vértice v.

Intuitivamente, um vértice não pode "ver" mais cores do que o número de arestas que chegam a ele. Portanto, temos a desigualdade trivial:

$$c(v) \leq d(v) \tag{2}$$

A igualdade $c(v) = d(v)$ ocorre se, e somente se, a coloração é *própria* em torno de v (ou seja, todas as arestas incidentes têm cores diferentes). Com base nisso, definimos o conceito de *melhoria* (*improvement*). Dizemos que uma coloração $\mathscr{C}'$ é uma melhoria sobre $\mathscr{C}$ se a soma global de cores distintas observadas pelos vértices aumenta:

$$\sum_{v \in V} c'(v) > \sum_{v \in V} c(v) \tag{3}$$

Uma k-coloração é dita *ótima* se ela não pode ser melhorada. Esse conceito de "otimalidade" é a chave para as provas construtivas que virão a seguir: a ideia é começar com uma coloração qualquer e "melhorá-la" iterativamente até atingir uma coloração onde a regra de adjacência seja satisfeita para o maior número possível de vértices.

Com o conceito de otimização já estabelecido, podemos finalmente analisar quais os fatores que impedem uma coloração de ser perfeita.

Suponha que atingimos uma k-coloração ótima $\mathscr{C}$. Agora, imagine que essa coloração ainda não é a "ideal" em um vértice u: a cor i está faltando em u, mas a cor j aparece repetida. Isso indica um desequilíbrio local.

Intuitivamente, gostaríamos de trocar algumas arestas da cor j por i para equilibrar a distribuição. O *Lema 6.1.2* de Bondy e Murty, aqui chamado de *Lema do Obstáculo em Ciclos Ímpares*, nos diz exatamente quando isso *não é possível*.

> "*Lema do Obstáculo em Ciclos Ímpares:* Seja $\mathscr{C}$ uma k-coloração ótima de G. Se existe um vértice u onde a cor i não aparece, mas a cor j aparece pelo menos duas vezes, então a componente conexa formada apenas pelas arestas dessas duas cores (i e j) que contém u é, necessariamente, um *ciclo ímpar*."

A prova dessa afirmação conecta-se diretamente ao Lema das Duas Cores e pode ser visualizada na *Figura 9*.

Perceba que No vértice u, temos duas arestas azuis (j) e nenhuma vermelha (i), tentar consertar isso alterando as cores ao longo do ciclo apenas deslocaria o problema para v_1 ou v_2, sem resolver o conflito globalmente. Se a componente contendo u *não* fosse um ciclo ímpar, poderíamos aplicar a

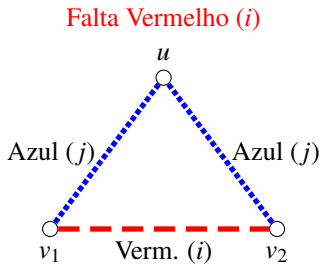


Figura 9: O obstáculo do ciclo ímpar.

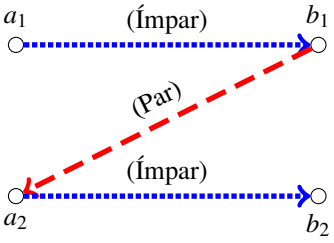


Figura 10: Qualquer caminho de comprimento ímpar termina necessariamente no time oposto.

lógica da extensibilidade vista anteriormente para re-colorir essa componente de modo que u passasse a ter ambas as cores (i e j). Isso faria com que o número de cores distintas em u aumentasse ($c(u)$ subiria em 1), sem prejudicar os outros vértices, criando uma coloração "melhor".

Como partimos da premissa de que a coloração original já era *ótima* (impossível de melhorar), essa re-coloração é impossível. Logo, a única explicação geométrica que trava essa melhoria é que estamos presos na estrutura rígida mostrada na Figura 9: um ciclo ímpar.

V. O Teorema de Kőnig

Com base na fundamentação estabelecida, alcançamos o ponto de convergência desta primeira parte. Os lemas anteriores construíram uma narrativa clara: a otimização de uma coloração só é bloqueada estruturalmente pela presença de *ciclos ímpares*.

Para contextualizar a importância do que vem a seguir, vale ressaltar que, quando o matemático húngaro Dénes Kőnig publicou este resultado em 1916 [9], a Teoria dos Grafos ainda nem existia como disciplina autônoma. Kőnig, que mais tarde escreveria o primeiro livro-texto da área, chegou a este teorema estudando a decomposição de matrizes e determinantes. Ele percebeu que certas estruturas algébricas poderiam ser traduzidas geometricamente para o que hoje chamamos de grafos bipartidos, provando que, nessas estruturas "bem-comportadas", a complexidade do problema desaparece.

A elegância da sua conexão reside no fato de que, por definição, a propriedade fundamental de um *grafo bipartido* é a ausência completa de ciclos de comprimento ímpar. Se a única barreira topológica para a otimização perfeita é o ciclo ímpar, e os grafos bipartidos são desprovidos dessa estrutura, a conclusão lógica é inevitável.

> "*Teorema de Kőnig (1916):* Se G é um grafo bipartido, então seu índice cromático é exatamente igual ao seu grau máximo, ou seja, $\chi'(G) = \Delta(G)$."

Para visualizar o porquê deste teorema funcionar, imagine que os vértices do grafo estão divididos em dois times rivais, *Time A* e *Time B*, e as arestas representam partidas entre eles. Em um grafo bipartido, um time nunca joga contra si mesmo; as arestas sempre ligam A a B.

Se tentarmos colorir as arestas (agendar os jogos) e encontrarmos um conflito que exige uma troca de cores em cadeia, essa cadeia de trocas funcionaria como um movimento de "ping-pong", ilustrado na *Figura 10*.

Para que um conflito seja insolúvel (como vimos no Lema anterior), essa cadeia precisaria fechar um ciclo ímpar mas se observarmos o movimento na figura veremos que *Passo 1 (Ímpar):* Sai de A $\rightarrow$ Chega em B, *Passo 2 (Par):* Sai de B $\rightarrow$ Volta para A, *Passo 3 (Ímpar):* Sai de A $\rightarrow$ Chega em B. Isso mostra que se nosso intuito for fechar um ciclo e voltar ao vértice de origem (que está em A), é necessário, obrigatoriamente, um número *par* de passos pois é impossível sair de A e voltar para A com um número ímpar de movimentos, pois estaríamos fisicamente no lado do Time B.

Podemos concluir portanto que o "curto-circuito" cromático do ciclo ímpar nunca acontece, sempre conseguimos resolver os conflitos locais e organizar as arestas em exatamente Δ rodadas (cores) perfeitas.

O Teorema de Kőnig representa, como vimos, o cenário ideal na coloração de arestas: uma classe de grafos onde a topologia colabora perfeitamente com a alocação de recursos, garantindo que o limite inferior natural (Δ) seja sempre suficiente. Nesses casos, não há desperdício e a estrutura bipartida assegura a inexistência dos conflitos cíclicos que impediriam a otimização.

Contudo, a modelagem de sistemas complexos frequentemente nos confronta com grafos que não possuem essa propriedade. O que acontece quando a restrição é levantada e os ciclos ímpares, como um simples triângulo, são reintroduzidos na estrutura? A intuição poderia sugerir que, sem a garantia de Kőnig, o número de cores necessárias poderia crescer descontroladamente acima do grau máximo.

A resposta para o caso geral foi descoberta quase cinquenta anos depois e revela um resultado surpreendente: mesmo na presença de ciclos ímpares e estruturas complexas, o "caos" cromático é extremamente limitado. O índice cromático nunca se afasta muito do ideal estabelecido por Kőnig, oscilando em um intervalo restrito de apenas dois valores possíveis.

VI. O Teorema de Vizing

O Teorema de Vizing [10] constitui um outro resultado clássico no problema de coloração de arestas. Em seu trabalho, ele mostrou que existia um limite superior para o índice cromático de um multigrafo. Um multigrafo é um grafo que possui mais de uma aresta que conecta um mesmo par de vértices (veja a figura 2). Contudo, o presente trabalho trata de grafos simples, então para cada par de vértices há somente uma aresta que os conecta. Sob essas hipóteses, o Teorema de Vizing possui o seguinte enunciado:

> "Seja G um grafo simples. Então vale a desigualdade: $\Delta(G) \leq \chi'(G) \leq \Delta(G) + 1$"

ISSN: 2675-3588

Uma vez que o índice cromático de um grafo é um número inteiro, o teorema diz que para qualquer grafo G simples, o seu índice cromático ou é igual ao grau máximo de G ou então é maior por uma unidade apenas. Esse é um resultado extremamente útil na construção de algoritmos para coloração própria e mínima, pois implica que precisamos apenas examinar dois conjuntos de colorações, o que reduz bastante o espaço de busca [16]. A primeira desigualdade é trivial e pode ser vista na seção *IV* em (1). A dificuldade reside na segunda desigualdade.

Na literatura existem vários tipos de provas da segunda parte: por indução e construção por exemplo. Neste texto usaremos o tipo de demonstração por *contradição*.

No enunciado do teorema temos duas proposições: p = "G é um grafo simples" e q = "$\Delta(G) \leq \chi'(G) \leq \Delta(G)+1$". Queremos mostrar que a ocorrência de p *implica* q. A estrutura da prova é a seguinte: assumimos que p ocorre mas *negamos a proposição* q, ou seja, assumimos o contrário daquilo que queremos provar, e assim teremos uma nova proposição diferente:

"Seja G um grafo simples. Então $\chi'(G) > \Delta(G)+1$"

Com base nessa proposição, nós faremos uma série de deduções lógicas válidas, e eventualmente chegaremos à conclusão de ser falso um fato já provado ser verdade. Mas isso é uma contradição, então a única explicação para essa incoerência lógica é ter suposto inicialmente que $\chi'(G) > \Delta(G)+1$. Então se essa proposição é falsa, sua negação é verdadeira e portanto: $\chi'(G) \leq \Delta(G)+1$. [17]

O teorema de Vizing divide os grafos que satisfazem as hipóteses do teorema em duas classes, de acordo com seu índice cromático; se um grafo G satisfaz $\chi'(G) = \Delta(G)$ ele é dito ser de *classe 1*, se $\chi'(G) = \Delta(G)+1$ ele é de *classe 2* [18].

Para compreender melhor essa classificação, é útil visualizar como a topologia do grafo impõe restrições locais. A distinção fundamental reside na capacidade da estrutura em acomodar emparelhamentos sem gerar conflitos insolúveis. Conforme ilustrado na Figura 11, um ciclo par (C_4), por ser um grafo bipartido, permite uma alternância perfeita de índices (representados pelos números 1 e 2), satisfazendo $\chi'(G) = \Delta(G) = 2$ e classificando-se como Classe 1. Em contrapartida, um ciclo ímpar (C_5) apresenta um impasse estrutural: ao tentar alternar os índices 1 e 2, a última aresta conecta vértices que já possuem incidências de ambos, obrigando o uso de um terceiro índice (número 3). Isso resulta em $\chi'(G) = 3 = \Delta(G)+1$, caracterizando o grafo como Classe 2.

Uma curiosidade interessante é que Holyer [19] mostrou que, dado um grafo G, decidir se ele é de Classe 1 ou de Classe 2 é um *problema NP-completo*. De forma simplificada, problemas NP-completos são problemas de decisão cuja solução, uma vez proposta, pode ser verificada de forma fácil, porém encontrar sua solução é difícil [6]. Por exemplo, dada uma coloração de arestas $\mathscr{C}$ qualquer sobre um grafo G, é fácil verificar se essa coloração é de Classe 1: basta checar se a coloração é própria e se utiliza exatamente $\Delta(G)$ cores. No entanto, decidir se existe tal coloração entre o enorme número de possibilidades é um problema computacionalmente difícil.

Na seção a seguir, veremos as demonstrações dos dois teoremas propostos.

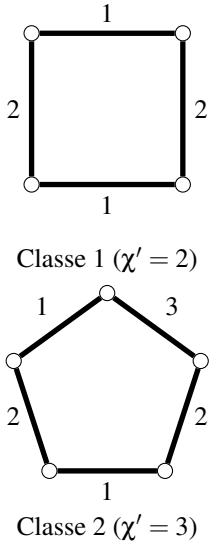


Figura 11: Comparação visual utilizando numeração nas arestas: O ciclo par (C_4) usa apenas rótulos 1 e 2, enquanto o ciclo ímpar (C_5) necessita do rótulo 3.

VII. Demonstração e Contribuições

Estabelecidas as condições estruturais e os lemas auxiliares sobre a distribuição de cores, o texto avança para a formalização dos dois pilares centrais da coloração de arestas.

Demonstra-se inicialmente o *Teorema de Kőnig*, provando que a ausência de ciclos ímpares em grafos bipartidos garante que o índice cromático atinja seu limite inferior natural (Δ). Subsequentemente, a análise expande-se para a classe dos grafos simples gerais. Mediante o método de redução ao absurdo e argumentos de recoloração, demonstra-se o célebre *Teorema de Vizing*. Este estabelece que, mesmo na presença de estruturas cíclicas ímpares, o índice cromático excede o grau máximo em no máximo uma unidade.

Ambas as demonstrações fundamentam-se na estrutura lógica apresentada por Bondy e Murty [11], utilizando os conceitos de otimização cromática e emparelhamentos definidos preliminarmente.

O Teorema de Kőnig enuncia-se da seguinte maneira.

Teorema de König VII.1. *Seja G um grafo bipartido. Então a igualdade abaixo se verifica*

$$\chi'(G) = \Delta(G). \tag{4}$$

Proof. A demonstração da igualdade utiliza o método de redução ao absurdo. Assume-se a falsidade da tese para obter uma contradição estrutural.

Seja G um grafo bipartido e suponha-se, por contradição, que $\chi'(G) > \Delta(G)$. Considere $\mathscr{C} = (E_1, E_2, \ldots, E_\Delta)$ uma Δ-coloração ótima das arestas de G. Como o grafo não é Δ-colorível propriamente, existe necessariamente um vértice u onde o número de cores presentes é inferior ao grau do vértice, satisfazendo a condição

$$c(u) < d(u) \tag{5}$$

Essa desigualdade implica uma falha na distribuição das cores em u. Especificamente, existem cores i e j tais que a cor

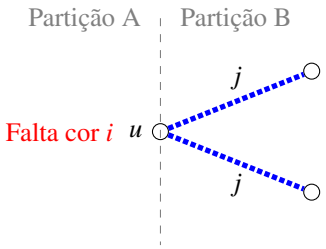


Figura 12: Defeito cromático em u: repetição da cor j (azul) e ausência da cor i (vermelha).

i não está representada em u (falta), enquanto a cor j aparece pelo menos duas vezes (repetição). A Figura 12 ilustra essa configuração local, onde o vértice u possui duas arestas azuis (j) e nenhuma aresta vermelha (i).

Aplica-se neste ponto o *Lema das duas cores* (Lema 6.1.2 de Bondy & Murty). Segundo este resultado, se tal falha ocorre em uma coloração ótima, a componente conexa do subgrafo induzido apenas pelas cores i e j que contém u deve ser, obrigatoriamente, um *ciclo ímpar*.

Entretanto, tal conclusão gera uma contradição topológica imediata com a natureza do grafo G. A definição de grafo bipartido exige que o conjunto de vértices possa ser dividido em dois subconjuntos disjuntos, A e B, onde toda aresta conecta um vértice de A a um de B.

Para que um ciclo exista, é necessário sair de uma partição e retornar a ela. Como cada passo na trilha alterna de partição ($A \to B \to A \to \ldots$), retornar ao vértice de origem exige necessariamente um número par de passos. Consequentemente, *é impossível existir um ciclo de comprimento ímpar em um grafo bipartido*.

Visto que a existência do ciclo ímpar exigido pelo lema é impossível, a suposição inicial de que $\chi'(G) > \Delta(G)$ revela-se falsa. Portanto, conclui-se que $\chi'(G) = \Delta(G)$. □

.

Vejamos a seguir a demonstração de um outro importante teorema relacionado ao Problema de Coloração de Arestas em Grafos.

Teorema de Vizing VII.2. *Seja G um grafo simples. Então:*

$$\Delta(G) \leq \chi'(G) \leq \Delta(G)+1. \tag{6}$$

Proof. Seja G um grafo simples. Suponha por absurdo que $\chi'(G) > \Delta(G)+1$. Seja $\mathscr{C} = (E_1, E_2, ..., E_{\Delta(G)+1})$ uma $(\Delta+1)$-coloração das arestas de G e seja u um vértice tal que:

$$c(u) < d(u) \tag{7}$$

Note que o vértice u existe, pois assumimos(por contradição) que o grafo G não é $(\Delta+1)$-colorível. Assim deve existir pelo menos um vértice tal que uma mesma cor esteja representada nele, exatamente o que o item (7) exprime(Lembre que $c(v)$ denota o número de cores distintas representadas em um dado vértice v). Então, existem certas cores i_0 e i_1 tais que: i_0 não está representado em u, e i_1 está representado pelo menos duas vezes em u. Esse fato ocorre porque, para todo vértice de G, em particular para u, segue que $d(u) < \Delta(G)+1$ e a coloração que estamos usando possui $\Delta(G)+1$ cores, logo pelo menos uma cor não é usada em u, o que justifica a existência de i_0. A existência da cor i_1 é consequência de (5). Seja uv_1 uma aresta que possua a cor i_1. Agora considere a seguinte asserção:

> (i): "Uma vez que $d(v_1) < \Delta(G)+1$ existe uma cor i_2 que não é representada em v_1. Note que i_2 necessariamente deve estar representada em u. Do contrário, poderíamos recolorir a aresta uv_1 com a cor i_2 e assim obter uma melhoria na nossa coloração, o que contradiz a hipótese dela ser ótima."

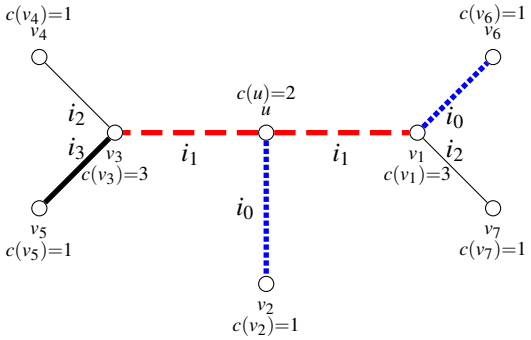


Figura 13: Grafo G com coloração c_0

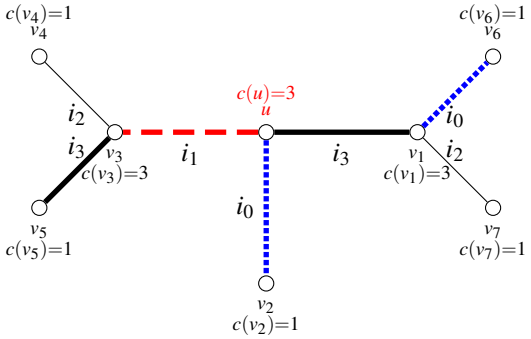


Figura 14: Grafo G com coloração c_0'

Vejamos um exemplo para compreender melhor o item (i). Considere o grafo G(figura 13), cujo grau máximo de vale 3(Naturalmente, o argumento também é válido se G fosse um grafo maior, o que importa é olhar localmente para o vértice u que sabemos que existe). Assim, a coloração denotada por c_0 é composta por 4 cores, denotadas por i, onde:

$$c_0 = \{i_0, i_1, i_2, i_3\}$$

Note que, o somatório do número de cores distintas que são representadas em cada vértice é 13, isto é, $\sum c(v) = 13$. A cor preta não é representada no vértice em u e nem em v_1. Crie uma nova coloração c_0' onde aresta uv_1 se torna da cor preta e o restante das arestas permanecem inalteradas. Para essa nova coloração, o somatório do número de cores distintas representadas em cada vértice é 14: $\sum c'(v) = 14$ (Figura 14). Ou seja, conseguimos melhorar a nossa coloração c_0, o que é uma contradição pois supomos que tal coloração era ótima. Portanto o item (i) é de fato verdadeiro.

Então sabemos que existe alguma aresta diferente de uv_1 que possui a cor i_2, chamemos ela de uv_2. Novamente, temos que $d(v_1) < \Delta(G)+1$, logo existe uma cor i_3 que não é representada em v_2. Por um raciocínio totalmente análogo ao feito em (i), a cor i_3 deve necessariamente estar representada em u, do contrário, seria possível fazer uma melhoria na

ISSN: 2675-3588

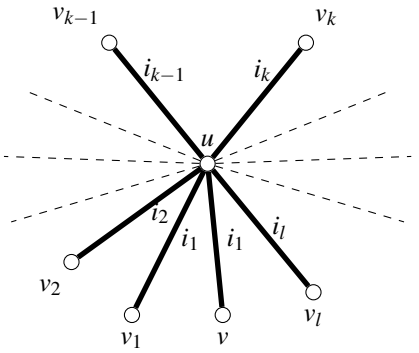


Figura 15: Estrutura gerada em u. Fonte: Bondy e Murty(1976)

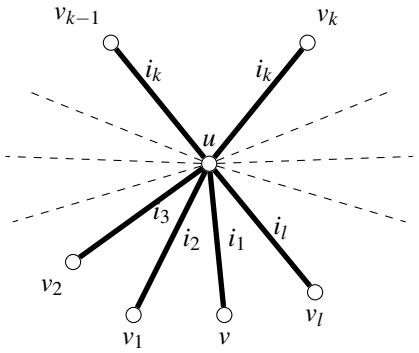


Figura 16: 1º recoloração. Fonte: Bondy e Murty(1976)

coloração φ atribuindo a cor i_2 à aresta uv_1 e a cor i_3 à aresta uv_2. Assim deve existir um aresta uv_3 com a cor i_3.

Continuando com esse procedimento, construiremos uma sequência de vértices $v_1, v_2, ...$ e uma sequência de cores $i_1, i_2, ...$ que possui as seguintes propriedades:

(a) A aresta uv_j possui a cor i_j.

(b) A cor i_{j+1} não aparece na aresta uv_j.

(c) Como estamos considerando um grafo simples finito, u possui grau finito, e em algum momento as cores começarão a se repetir nas arestas de u.

Graficamente, podemos ver a estrutura construída(Figura 15). Nela podemos ver todas as arestas adjacentes a u, sendo que a cor i_1 é representada duas vezes e a cor i_0 não é representada nenhuma vez. A cor i_2 não é representada em v_1 mas é representada em v_1, a cor i_3 não é representada em v_2 porém é representada em i_3 do contrário, como já discutimos, seria possível reatribuir cores às arestas do grafo de modo a obter uma melhoria para a $\Delta(G)+1$-coloração, o que geraria um contradição pois supomos que tal coloração é máxima(e assim por diante para as demais cores). Agora faremos recolorações no grafo G de forma a manter sua otimalidade.

A primeira coloração se dará da seguinte forma: a aresta uv_j receberá a cor da aresta uv_{j+1} com $1 \leq j \leq k-1$. Aqui, as cores das arestas de uv_1 até uv_k são todas distintas, e depois da aresta v_k as cores começam a se repetir seguindo a propriedade (c) dessa estrutura construída. Na prática estamos apenas deslocando as cores uma unidade no sentido anti-horário: a aresta uv_{k-1} receber a cor i_k, a aresta uv_{k-2} recebe a cor i_{k-1} ..., a aresta uv_2 recebe a cor i_3, a aresta uv_1 recebe a cor i_2 (Figura 16).

Note que a nova coloração $\mathscr{C}' = (E'_1, E'_2, ..., E'_{\Delta(G)+1})$ também é uma $\Delta(G)+1$-coloração ótima, pois na estrutura que construímos, a cor i_{j+1} não aparece na aresta u_j(propriedade (b)). Ou seja a quantidade de cores distintas representadas em cada vértice permanece inalterada. Por exemplo, observe a aresta uv_1. Antes a cor i_2 não era representada nela(do contrario teríamos uma contradição), isto é, nenhuma outra arestas diferente uv_1 possuia a cor i_2 em v_1. Agora surge a questão: e se v_1 tiver uma aresta wv_1 com a cor i_1? Se isso ocorresse, ao colorir uv_1 com i_2 haveria uma melhoria na coloração(uma contradição). Logo estamos trocando *uma cor(i_1) que só aparece uma vez em v_1 por outra(i_2) que irá aparecer apenas uma vez em v_1*. O

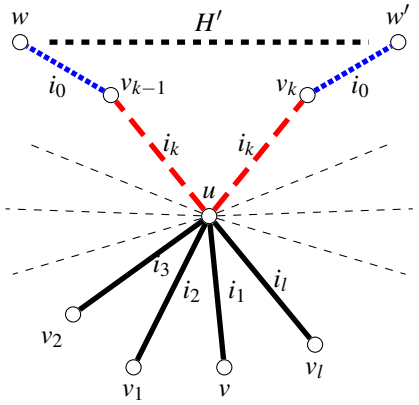


Figura 17: Ciclo ímpar formado pela componente H'. Fonte: Adaptado Bondy e Murty(1976).

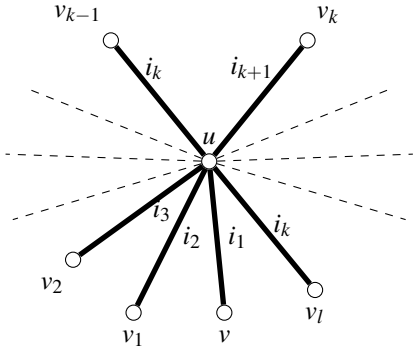


Figura 18: 2º recoloração. Fonte: Bondy e Murty(1976)

mesmo vale para os demais vértices. Veja que a coloração só altera essa região especifica, o restante do grafo permanece da mesma forma.

Então estamos diante de uma $\Delta(G)+1$-coloração ótima, onde o vértice u possui uma cor que não é representada nele(i_0 por hipótese) e um outra cor que é representada pelo menos 2 vezes(i_k). Assim pelo *Lema do Obstáculo em Ciclos Ímpares* visto na sessão IV, a componente conexa H' formada pelas arestas das cores i_0 e i_k, isto é, $H' = G[E'_{i_0} \cup E'_{i_k}]$ é um ciclo ímpar e contém o vértice u(Figura 17).

Agora faremos uma segunda recoloração. Cada aresta uv_j receberá a cor i_{j+1} com $k \leq j \leq l-1$, e para aresta uv_l atribuímos a cor i_k(Figura 18). A lógica aqui é muito semelhante à primeira parte, só que agora estamos considerando as arestas que não foram coloridas na primeira fase. Assim a aresta uv_k receberá a cor i_{k+1}, a aresta uv_{k+1} receberá a cor i_{k+2}..., a aresta uv_{l-1} receberá a cor i_l. O

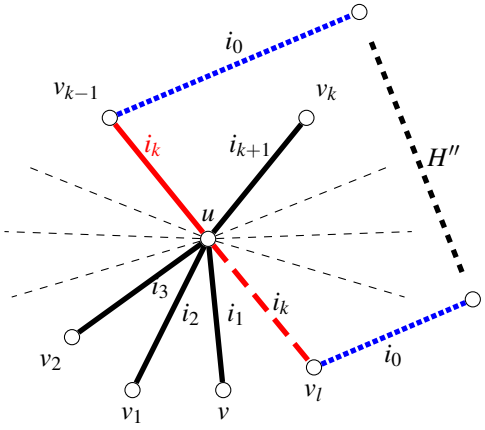


Figura 19: Ciclo ímpar formado pela componente H''. Fonte: Bondy e Murty(1976)

único ponto a se atentar é que diferença é cor i_k à aresta uv_l, isso ocorreu porque uma vez que o grau de u é finito, haverá uma repetição de cor, isto é, haverá um l tal que $i_{l+1} = i_k$, onde i_k é justamente uma cor que já ocorreu em algum vértice anterior. Perceba que essa segunda coloração $\mathscr{C}'' = (E''_1, E''_2, ..., E''_{\Delta(G)+1})$ também é uma $(\Delta(G)+1)$-coloração ótima por um argumento análogo ao aquele usado na primeira coloração. Novamente, pelo *Lema do Obstáculo em Ciclo Ímpares*, a componente: $H'' = G[E''_{i_0} \cup E''_{i_k}]$ é um ciclo ímpar e contém u(Figura 19).

Note que como a componente H' é conexa, então sempre existirá um caminho de u até v_{k-1} e de v_{k-1} até v_k. No nosso exemplo esse caminho é: $uv_{k-1}ww'v_k$. Esse caminho continuará existindo na segunda coloração, pois nela nós consideramos apenas as arestas com as cores de k até $l-1$, e o caminho citado surge nas arestas de cores 1 até $k-1$. É importante notar que a cada recoloração, nós modificamos apenas partes localizadas do grafo, o restante se mantém inalterado. Desse modo, a componente H'' contém o vértice v_k e seu grau é um. Temos portanto uma contradição, pois H'' é um ciclo ímpar e não pode ter vértices com grau um. Essa falha lógica surgiu porque supomos inicialmente que $\chi'(G) > \Delta(G)+1$. Então segue que: $\chi'(G) \leq \Delta(G)+1$, o que encerra a demonstração.

□

Como foi dito na seção VI, o Teorema possui uma grande importância para algoritmos de coloração de arestas. Apesar desse tópico principal deste trabalho, será interessante tecer alguns comentários.

A estrutura que construímos na demonstração(15), onde temos um vértice central(u) e demais outros vértices adjacentes que seguem algumas propriedades(VII) é conhecida na literatura como *Fan* ou *Vizing's Fan*(Fan de Vizing) [20]. Alguns autores definem essa estrutura explícitamente e outros não(como no caso do Bondy e Murty). Contudo, a vantagem de definir essa estrutura e juntamente realizar a demonstração por construção do Teorema de Vizing é que obtemos um *algoritmo para coloração própria de arestas* que utiliza no máximo $\Delta(G)+1$ cores.

O algoritmo de coloração de arestas baseado na prova construtiva do Teorema de Vizing é conhecido como *Algoritmo de Coloração de Arestas de Mista Gries*, e leva o nome dos autores que propuseram a rotina [21]. Uma

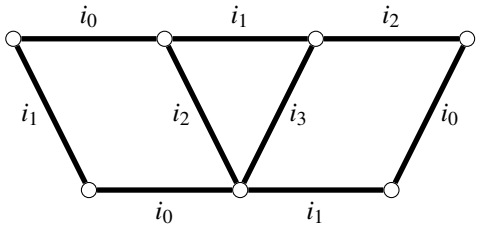


Figura 20: Grafo G_1.

implementação do algoritmo na linguagem de programação *Python* pode ser vista em [22].

Exemplo:

Vamos utilizar um exemplo para melhor visualizar a aplicação prática do Teorema de Vizing. Suponha que desejamos colorir propriamente o grafo G_1 da seção I(figura 20)

Perceba que nesse grafo, $\Delta(G_1) = 4$. Portanto, pelo *Teorema de Vizing*, o número de cores necessárias para colorir o grafo propriamente não será maior que $\Delta(G_1)+1 = 4+1 = 5$. De fato, o leitor poderá verificar que não é possível colorir esse grafo com menos de 5 cores. Definamos a coloração $c_0 = \{i_0, i_1, i_2, i_3\}$, onde cada $i_j (0 \leq j \leq 3)$ representa uma cor distinta. A coloração pode ser vista na figura:

Uma observação importante é que poderíamos ter um grafo G cujo índice cromático fosse menor que $\Delta(G)+1$(seção VIII). Contudo isso não invalida o teorema pois o valor $\Delta(G)+1$ é um limite superior, ou seja, a garantia é que não será preciso mais que $\Delta(G)+1$ cores para colorir um grafo G simples e finito qualquer.

VIII. APLICAÇÕES

A Coloração de Arestas de Grafos possui uma série de aplicações práticas, tais como planejamento de rotas, tráfego em redes e muitas outras [23]. Nesta seção trataremos de um problema bastante interessante: O Problema de Programação de Tabelas Esportivas. Utilizaremos como referência o trabalho de Januário (2015) [24].

Um torneio do tipo *round a robin*(todos contra todos), é um competição que envolve t times diferentes que disputam entre si uma quantidade j de jogos. Por exemplo, para $t = 4$ e $j = 1$ teremos um torneio em que, cada time disputa contra os demais 3 uma vez. Nesse cenário 2 questionamentos poderiam surgir: como elaborar uma agenda de jogos de modo que, os times não disputem não mais que uma partida em uma mesma rodada e quantas rodadas seriam necessárias? Podemos responder essas pergunta utilizando os conhecimentos aprendidos até aqui sobre grafos.

Em primeiro lugar, note que podemos facilmente modelar a estrutura do torneio da seguinte forma: defina um grafo $G = (V, E)$, onde os vértices representam os times do torneio e as a arestas representam a as partidas que devem ocorrer entre eles. Consideremos $t = 4$ e $j = 1$, ou seja, 4 times t_1, t_2, t_3 e t_4 que disputam uma partida entre si. O grafo que representa a estrutura desse torneio pode ser vista na figura 21.

Agora devemos encontrar uma forma de garantir que nas rodadas que se seguirão cada time jogue apenas uma partida. Isso pode ser feito utilizando as técnicas de Coloração de Arestas. Definimos então uma coloração $\mathscr{C}$ em que cada cor

ISSN: 2675-3588

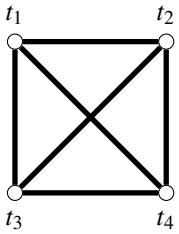


Figura 21: Grafo que representa o torneio

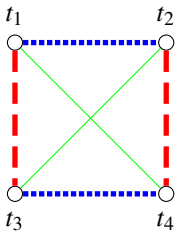


Figura 22: Grafo que representa o torneio, colorido

representa uma rodada do torneio. Observe que buscamos uma coloração do grafo G que seja própria. A coloração sendo própria, garantimos que nenhuma aresta adjacente a qualquer um dos vértices tenha a mesma cor, ou seja, cada partida ocorrerá em uma rodada diferente, e não haverão conflitos.

Sendo assim, estamos aptos a resolver a questão acerca do número de rodadas necessárias para realizar esse torneio. Responder isso é equivalente a reponder: quantas cores são necessárias para obter uma coloração própria de G? Pelo teorema de Vizing, sabemos que o melhor valor e que funciona para todos os casos é $\Delta(G)+1 = 3+1 = 4$ cores. Obviamente, para valores maiores também é possível obter uma coloração própria, mas é de interesse usar a menor número de cores, pois implica que teremos um menor número de rodadas. Note também que o grafo do torneio usado como exemplo pode ser colorido propriamente com 3 cores, pois como ele é bipartido, pelo Teorema de Kőnig, $\chi'(G) = \Delta(G)$. Porém o limite superior fornecido pelo Teorema de Vizing é melhor no sentido de que, funciona para todos os casos possíveis, mesmo se o grafo do torneio não for bipartido.

O segundo passo seria então aplicar algum algoritmo de coloração própria de arestas no grafo G(sabendo que será preciso não mais que 4 cores) obtendo portanto, o agendamento das partidas. Uma solução pode ser vista na figura 22. Note que, a coloração resolve o problema pois, nunhum vértice(time) possui mais de uma aresta(partida) cuja cor(horário) é o mesmo.

IX. Resultados e Reflexões

A base teórica da coloração de arestas encontra-se bem consolidada na literatura, todavia a complexidade dos argumentos construtivos e das técnicas de recoloração iterativa impõe frequentemente barreiras ao aprendizado em nível de graduação.

O mérito central deste trabalho reside não apenas na demonstração formal, mas na sistematização visual desses raciocínios. Ao decompor as restrições estruturais e os impeditivos topológicos em diagramas sequenciais, evidenciou-se a natureza local do problema. A análise permitiu demonstrar que, enquanto certas classes de grafos com propriedades específicas permitem uma alocação ótima de recursos garantindo que o índice cromático iguale o grau máximo ($\chi' = \Delta$), a generalização para estruturas mais complexas acarreta, no pior caso, o incremento de apenas uma cor adicional ($\chi' \leq \Delta + 1$).

A expectativa é que este material atue como um instrumento pedagógico facilitador, permitindo que estudantes das áreas de Computação e Matemática transitem da intuição geométrica para o rigor analítico das provas formais com maior fluidez e compreensão.

X. Considerações Finais

Este estudo revisitou o Problema de Coloração de Arestas, partindo de suas raízes históricas no Problema das Quatro Cores até a formalização contemporânea. A análise comparativa entre a estrutura rígida dos grafos bipartidos e a flexibilidade dos grafos simples permitiu compreender como propriedades topológicas (como a paridade de ciclos) ditam os limites de alocação de recursos.

Conclui-se que a abordagem geométrica e iterativa é essencial para a compreensão profunda do Índice Cromático. A dificuldade inerente em conciliar o rigor matemático com a clareza didática foi mitigada pelo uso extensivo de representações visuais, que serviram como âncoras cognitivas para as abstrações lógicas.

Como trabalhos futuros, sugere-se a expansão desta análise para o Teorema de Vizing generalizado para multigrafos, onde a multiplicidade das arestas introduz novas variáveis à desigualdade cromática. Espera-se que este material sirva como referência pedagógica, facilitando o ensino de Otimização Combinatória e Teoria dos Grafos em cursos de Computação e Matemática.

Referências

[1] A. Cayley, "On the colouring of maps," *Proceedings of the Royal Geographical Society and Monthly Record of Geography*, vol. 1, no. 4, pp. 259–261, 1879.

[2] F. Guthrie, "Note on the colouring of maps," *Proceedings of the Royal Society of Edinburgh*, vol. 10, pp. 727–728, 1880.

[3] A. B. Kempe, "On the geographical problem of the four colours," *American Journal of Mathematics*, vol. 2, p. 193, 1879.

[4] K. Appel and W. Haken, "Every planar map is four colorable," *Bulletin of the American Mathematical Society*, vol. 82, no. 5, pp. 711–712, 1976.

[5] ——, "Every planar map is four colorable. part i: Discharging," *Illinois Journal of Mathematics*, vol. 21, no. 3, pp. 429–490, 1977.

[6] J. L. Szwarcfiter, *Teoria computacional de grafos: os algoritmos*. Rio de Janeiro: Elsevier, 2018.

[7] C. L. Lucchesi and D. H. Younger, "A minimax theorem for directed graphs," *Journal of the London Mathematical Society*, vol. 2, no. 3, pp. 369–374, 1978.

[8] N. Maculan, "The steiner tree problem," *Annals of Operations Research*, vol. 13, no. 1, pp. 53–70, 1987.

[9] D. König, "Über graphen und ihre anwendung auf determinantentheorie und mengenlehre," *Mathematische Annalen*, vol. 77, pp. 453–465, 1916.

[10] V. G. Vizing, "On an estimate of the chromatic class of a p-graph," *Diskretnyi Analiz*, vol. 3, pp. 25–30, 1964.

[11] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*. New York: Elsevier, 1976.

[12] L. P. da Silva, "Aplicação da teoria dos grafos no ensino médio à luz das contribuições do PROFMAT," Dissertação (Mestrado Profissional em Matemática), Universidade Federal de Sergipe, São Cristóvão, 2016, orientador: Dr. Fábio dos Santos.

[13] J. G. Müller and T. Baier, "Teoria dos grafos: uma possibilidade pedagógica para o ensino fundamental," *Revista de Educação Matemática e Tecnologia Iberoamericana*, vol. 12, no. 2, 2021.

[14] F. V. S. Soares, "Três teoremas interessantes em teoria dos grafos," Dissertação (Mestrado Profissional em Matemática), Universidade Federal do Ceará, Fortaleza, 2017, orientador: Prof. Dr. Antonio Caminha Muniz Neto.

[15] Y. M. L. D. V. Y. Cantañede, G. de Barros Bianchini, and T. D. dos Santos, "Reflexões e práticas pedagógicas no escopo da disciplina de teoria da computação," *Academic Journal on Computing, Engineering and Applied Mathematics*, vol. 6, no. 2, pp. 10–17, oct 2025.

[16] S.-i. Nakano, X. Zhou, and T. Nishizeki, *Edge-coloring algorithms*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 172–183.

[17] D. J. Velleman, *How to Prove It: A Structured Approach*, 3rd ed. Cambridge University Press, 2019.

[18] R. Diestel, *Graph Theory*, 5th ed. Berlim: Springer Publishing Company, Incorporated, 2017.

[19] I. Holyer, "The np-completeness of edge-coloring," *SIAM Journal on Computing*, vol. 10, no. 4, pp. 718–720, 1981.

[20] M. Stiebitz, D. Scheide, B. Toft, and L. Favrholdt, *Graph Edge Coloring: Vizing's Theorem and Goldberg's Conjecture*, ser. CourseSmart. Wiley, 2012.

[21] J. Misra and D. Gries, "A constructive proof of vizing's theorem," *Information Processing Letters*, vol. 41, no. 3, pp. 131–133, 1992. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0020019092900041S

[22] M. A. Maldonado, "An implementation of the misra & gries edge coloring algorithm and its integration on sagemath," Curitiba, 2023.

[23] P. Formanowicz and K. Tanaś, "A survey of graph coloring - its types, methods and applications," *Foundations of Computing and Decision Sciences*, vol. 37, 09 2012.

[24] T. Januario, S. Urrutia, C. C. Ribeiro, and D. de Werra, "Edge coloring: A natural model for sports scheduling," *European Journal of Operational Research*, vol. 254, no. 1, pp. 1–8, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221716301667

 ISSN: 2675-3588

Demonstração prática e pedagógica do Teorema de Tutte-Berge e o Teorema de Tutte

A Practical and Pedagogical Demonstration of the Tutte-Berge and Tutte Theorems

João Pedro Felix Veloso[1], Artur Anderson Alves Corrêa[1], Daniel Martins da Silva[2] e Tanilson Dias dos Santos[1]

[1] *Universidade Federal do Tocantins, Tocantins, Brasil*
[2] *Universidade Federal do Norte do Tocantins, Tocantins, Brasil*

Data de recebimento do manuscrito: 01/12/2025
Data de aceitação do manuscrito: 22/01/2026
Data de publicação: 10/02/2026

Resumo—Este artigo representa o relato de uma experiência pedagógica desenvolvido na disciplina de Teoria dos Grafos no curso de Ciência da Computação, ofertada no semestre 2025/2 na Universidade Federal do Tocantins. A aplicação prática dos conceitos de grafos aprendidos na disciplina partirá de uma reprodução da demonstração dos Teoremas de Tutte-Berge e de Tutte, os quais representaram grandes avanços na pesquisa de emparelhamentos em grafos. Mais especificamente, o estudo da condição de existência de emparelhamento máximo e perfeito em um grafo qualquer. Estes estudos, por sua vez, abriram as portas para a resolução de problemas cada vez mais complexos, e a versatilidade de seus usos pode ser interpretada como complemento das conquistas trazidas pelo Teorema de Hall. A explicação de tais conceitos será feita com base nas principais dificuldades encontradas pelo corpo estudantil, demonstrando de forma didática e ilustrativa, por meio de imagens, a fim de reduzir a abstração inerente ao tema.

Palavras-chave—Teoria dos grafos, Teorema de Tutte-Berge, Teorema de Tutte, grafos máximos, barreiras, Seminários Acadêmicos, Experiência Pedagógica.

Abstract—*This paper reports on a pedagogical experience developed during the Graph Theory course within the Computer Science program, offered in the second semester of 2025 at the Federal University of Tocantins. The practical application of the graph concepts learned in the course involves reproducing the proofs of the Tutte-Berge and Tutte theorems, which represented major advancements in graph matching research. More specifically, it focuses on the study of the existence conditions for maximum and perfect matchings in arbitrary graphs. These studies, in turn, paved the way for solving increasingly complex problems, and the versatility of their applications can be interpreted as a complement to the achievements brought by Hall's Theorem. The explanation of these concepts is based on the primary difficulties encountered by the student body, employing a didactic approach illustrated with images to reduce the inherent abstraction of the subject matter.*

Keywords—*Graph Theory, Tutte-Berge Theorem, Tutte's Theorem, Maximum Matchings, Barriers, Academic Seminars, Pedagogical Experience.*

I. Introdução

A teoria dos Grafos é uma das grandes protagonistas que permeiam o mundo da computação, oferecendo uma linguagem universal para a modelagem de relacionamentos e estruturas complexas. O estudo de grafos não se limita apenas à abstração matemática; ele permeia soluções para problemas reais e contemporâneos, variando desde a otimização de rotas em sistemas de logística e o design de circuitos eletrônicos até a análise de redes sociais e a bioinformática. A capacidade de abstrair problemas do mundo real em vértices e arestas, aplicando sobre eles algoritmos eficientes de busca, fluxo e conexidade, é uma habilidade indispensável para o cientista da computação moderno.

Partindo deste contexto, o estudo dos emparelhamentos nos grafos remonta a dezenas de anos repletas de contribuições. Redes sociais, problemas de atribuições de postos de trabalho, alocações de recursos ,entre outros, são os problemas que os emparelhamentos enfrentaram, contudo, o foco deste artigo está no emparelhamento máximo, ou seja, o emparelhamento de maior cardinalidade possível em um grafo G. Como um dos maiores representantes do estudo do emparelhamento máximo, em 1935, Philip Hall apresenta o

Dados de contato: Artur Anderson Alves Corrêa, alves.artur@uft.edu.br

teorema de Hall, popularmente conhecido como o "Teorema do casamento". Esse nome popular adveio da natureza do problema que partia da seguinte metáfora: se todo grupo de meninas em uma vila gostar coletivamente de pelo menos tantos meninos quanto há meninas no grupo, então cada menina pode se casar com um menino de quem ela gosta.

Mais formalmente, temos que o teorema de Hall apresentou as ferramentas necessárias para as descobertas de emparelhamentos máximos em grafos bipartidos.O Teorema de Hall tem se mostrado uma ferramenta valiosa tanto na teoria dos grafos quanto em outras áreas da matemática. Ademais, em 1957 Claude Berge avançou o estudo do problema do emparelhamento máximo confirmando a relação crucial entre caminhos M-aumentantes e emparelhamentos máximos. Relação esta, já previamente apontada (mas não provada), por Konig em 1931 e Pettersen em 1891. Herdando estas contribuições, William Thomas Tutte avança com o teorema de Tutte-Berge e o teorema de Tutte, descobrindo uma fórmula do tamanho de um emparelhamento máximo em um grafo qualquer e também uma condição de existência para um emparelhamento perfeito.

Embora o Teorema de Hall tenha estabelecido um marco fundamental, sua aplicabilidade direta restringe-se aos grafos bipartidos, deixando uma lacuna significativa para estruturas mais complexas onde a bipartição não é garantida. É nesse cenário que a generalização proposta por Tutte se torna revolucionária. Ao introduzir o conceito de componentes ímpares resultantes da remoção de vértices, o Teorema de Tutte (1947) fornece uma condição necessária e suficiente para a existência de um emparelhamento perfeito em um grafo qualquer, superando as limitações impostas pela necessidade de bipartição. A fórmula de Tutte-Berge, consolidada posteriormente em 1958, expande essa visão ao quantificar a deficiência de um grafo, ou seja, determinar o tamanho exato do emparelhamento máximo baseando-se na estrutura topológica do grafo e na análise de seus subconjuntos críticos, conhecidos como barreiras.

Assim, partindo do reconhecimento da importância desses teoremas, o artigo se propõe à reprodução de resultados já adquiridos através de uma perspectiva pedagógica e a disseminação desse conhecimento aos alunos em escala pessoal. Além disso, promove-se a introdução de todos os conceitos necessários para o entendimento dos teoremas, facilitando o acesso às nomenclaturas utilizadas no artigo.

Partindo para a estrutura, o artigo está organizado da seguinte maneira: na *Seção 2 (Preliminares)*, são definidos os conceitos básicos, como grafo, conexidade, componentes e emparelhamento, estabelecendo a notação e o vocabulário necessários. Em seguida, a *Seção 3 (Trabalhos Relacionados)* apresenta uma revisão bibliográfica, situando este trabalho em relação a outras abordagens pedagógicas e técnicas existentes na literatura. Avançando para a definição do escopo, a *Seção 4 (Descrição do Problema)* detalha os teoremas de Tutte-Berge e Tutte, bem como sua importância histórica. Já na *Seção 5 (Demonstração e Contribuições)*, encontra-se o núcleo do trabalho, contendo as demonstrações passo a passo dos teoremas escolhidos. Posteriormente, a *Seção 6 (Resultados e Reflexões)* discute as dificuldades encontradas durante o estudo, as estratégias de superação e realiza discussões quanto aos resultados, e, por fim, a *Seção 7 (Considerações Finais)* sintetiza os aprendizados e conclui a temática.

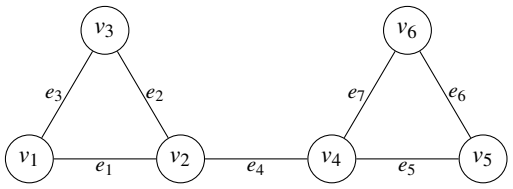
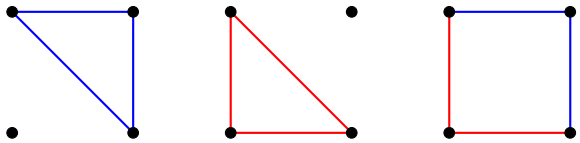


Figura 1: Grafo G ilustrando vértices (v_i), arestas (e_i), ciclos e conectividade.

Figura 2: Representação de um grafo trivial, composto por um único vértice isolado.



(a) G_1 (azul) **(b)** G_2 (vermelho) **(c)** $G_1 \Delta G_2$

Figura 3: Ilustração da diferença simétrica.

II. PRELIMINARES

Um grafo G(V,E) é uma estrutura de dados formada por dois conjuntos: um conjunto V chamado de vértices e um conjunto E de elementos chamados de arestas; cada aresta está associada a dois vértices: o primeiro é a ponta inicial da aresta e o segundo é a ponta final. Pode-se imaginar que um grafo é um mapa rodoviário idealizado: os vértices são cidades A e B e as arestas são estradas. Considere o grafo 1:

Chamamos de *subgrafo* um grafo formado por um conjunto de vértices e arestas do grafo original. Assim, considere um subgrafo H com os conjuntos de vértices $V = \{v_1, v_2, v_3\}$ e arestas $E = \{e_3, e_2, e_1\}$. A partir do subgrafo $H(V,E)$, podemos definir o conceito de *caminho*: um caminho em grafos é uma sequência de vértices interligados por arestas, onde o vértice final de uma aresta é o vértice inicial da próxima.

Ou seja, o conjunto $V = \{v_1, v_2\}$ é um caminho conectado pela aresta e_1. Como extensão dessa ideia, temos o conceito de *ciclo*: um ciclo em grafos é um caminho que começa e termina no mesmo vértice, sem repetir outros vértices no percurso. Ou seja, um exemplo de ciclo é $V = \{v_1, v_2, v_3\}$; partindo de v_1 pela aresta e_3, partindo de v_3 pela aresta e_2 e partindo de v_2 pela aresta e_1, temos um ciclo.

Continuamente, um grafo trivial é definido como um grafo que possui exatamente um vértice e nenhuma aresta.

Matematicamente, se $G = (V,E)$, então G é trivial se $|V| = 1$ e $E = \emptyset$.Também, outro conceito que deve ser explicado é a *diferença simétrica*. A diferença simétrica de dois grafos(denotado por $G_1 \Delta G_2$) é uma operação que resulta em um novo grafo contendo apenas as arestas que são exclusivas de cada um dos grafos originais.

O grafo resultante da figura 3 (c) contém apenas as arestas exclusivas de G_1 (topo) e exclusivas de G_2 (fundo). A aresta diagonal, presente em ambos, é removida.Além disso, um grafo é *conexo* se existir um caminho entre qualquer par de

ISSN: 2675-3588

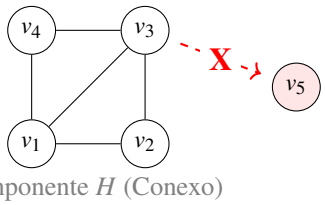
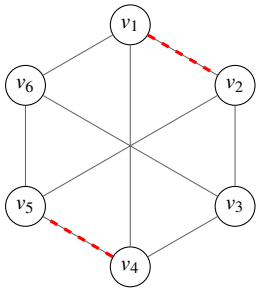
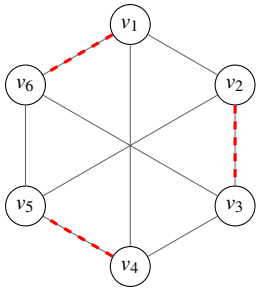


Figura 4: Exemplo de grafo desconexo. O componente H à esquerda é conexo internamente, mas o vértice v_5 está isolado.

Figura 5: Exemplos de emparelhamento em um mesmo grafo G:



(a) Emparelhamento não máximo (tamanho 2). Vértices v_3 e v_6 não foram emparelhados.



(b) Emparelhamento máximo (tamanho 3). Neste caso, é um emparelhamento perfeito.

vértices. Em outras palavras, é possível ir de qualquer vértice para qualquer outro vértice usando apenas as arestas do grafo. Se não for possível, o grafo é considerado desconexo.

Levando em conta o subgrafo H (a parte esquerda da figura), é possível ir de qualquer vértice a outro através de suas arestas; isso significa que o subgrafo H é conexo.

Porém, considerando a *Figura 4*, é impossível que v_3 alcance v_5. De fato, é impossível que qualquer vértice do componente H chegue até v_5, pois não há qualquer aresta que ligue o vértice v_5 aos outros vértices. Portanto, a figura representa um grafo *desconexo*.

Seguindo adiante, iremos para o conceito de *emparelhamento*. Um emparelhamento é um conjunto de arestas onde nenhuma delas compartilha o mesmo vértice. Em termos simples, é uma seleção de conexões onde cada vértice do grafo está ligado a, no máximo, um outro vértice.Isso pode ser entendido como a formação de pares exclusivos dentro de um grupo. Os vértices "selecionados", isto é, incidentes a uma aresta emparelhada são chamados de M-saturados. Caso não sejam, são chamados de M-insaturados. A partir deste princípio, podemos definir *emparelhamento máximo* que é a cardinalidade do maior emparelhamento possível no grafo.

As arestas em vermelho e tracejadas indicam os pares

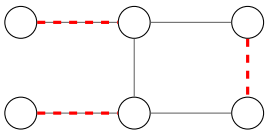
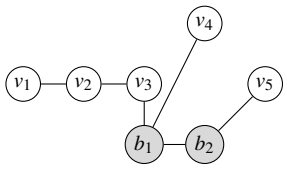
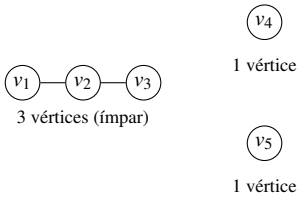


Figura 6: Emparelhamento máximo (e perfeito). As arestas em vermelho e tracejadas indicam os pares exclusivos formados.

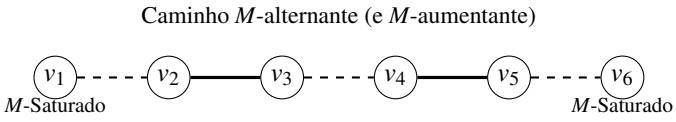


(a) Grafo G com $B = \{b_1, b_2\}$ destacado



(b) $G - B$: três componentes ímpares

Figura 7: Representação de uma barreira. A remoção de B produz mais componentes ímpares do que $|B|$.



exclusivos formados. Em 5(a) temos um conjunto válido, mas que poderia ser maior. Em 5(b) temos o maior conjunto possível para este grafo.Além disso, temos o conceito de *emparelhamento perfeito*. Diz-se que um emparelhamento M é *perfeito* se todo vértice do grafo estiver saturado por M. Naturalmente, todo emparelhamento perfeito é *máximo*, e todo emparelhamento máximo é *maximal* (isto é, não pode ser estendido adicionando-se arestas).

Avançando, um *vértice essencial* é aquele que todo emparelhamento máximo o cobre. Com a ideia de emparelhamento determinada, podemos partir para o conceito de *barreira*: Formalmente, dado um grafo G, um subconjunto de vértices B é chamado de barreira se a remoção de B divide o grafo em um número de componentes ímpares (componentes de um grafo com uma quantidade ímpar de vértices) maior que o tamanho do próprio conjunto B.

Ademais, deve-se introduzir conceito de caminho M-alternante e caminho M-aumentante. Seja G um grafo geral, E o conjunto de arestas de G e M um emparelhamento de G. Um *caminho M-alternante* em G é um caminho cujas arestas pertencem alternadamente a $E \setminus M$ e a M. Um caminho M-alternante cujos vértices extremos são ambos M-Saturados é chamado *caminho M-aumentante*. Observe que um caminho M-aumentante possui uma quantidade par de vértices.

Para a melhor compreensão das fórmulas apresentadas a seguir no artigo, partimos das seguintes denominações: A quantidade de arestas em um emparelhamento máximo será denotada por $\alpha'(G)$. Além disso, denotaremos por $o(G)$ como o número de componentes ímpares do grafo. Também, chamaremos de grafos hipoemparelháveis grafos que não possuem emparelhamentos perfeitos, contudo, qualquer subgrafo com qualquer vértice retirado possui

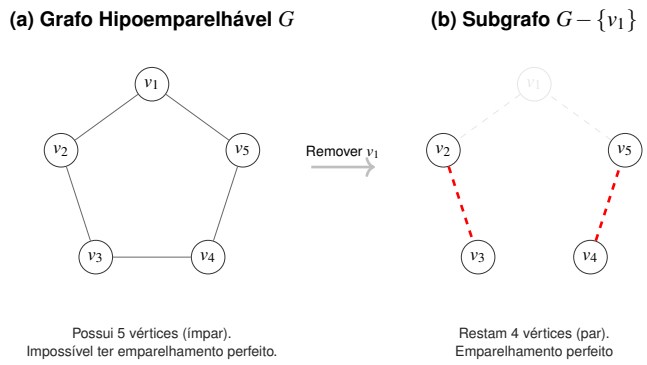


Figura 8: Ilustração de um grafo hipoemparelhável.

$$def(G) = (o(G-S) - |S|)$$

Figura 9: Representação matemática da deficiência

emparelhamento perfeito.

Em (a), o grafo original C_5 não tem emparelhamento perfeito devido à paridade. Em (b), após a remoção do vértice v_1, o subgrafo restante admite um emparelhamento perfeito (arestas vermelhas tracejadas).

O comportamento demonstrado se repete não importa qual vértice seja retirado. Ademais, devemos definir a ideia de *deficiência*. A deficiência mede quantos vértices não podem ser pareados, no pior caso, se tentarmos formar um emparelhamento. Note que o(G-S) é o número de componentes ímpares após a remoção de S. Sabemos que cada componente ímpar garante que pelo menos 1 vértice ficará sem par. Ou seja, a deficiência representa quantos vértices ficam inevitavelmente "solitários" depois que removemos S.

Com toda a introdução teórica feita, partiremos para uma pequena revisão de literatura quanto aos problemas do emparelhamento máximo e a evolução pedagógica do ensino dos grafos.

III. Trabalhos Relacionados

A literatura voltada ao ensino de Ciência da Computação e, especificamente, de Teoria dos Grafos, destaca que a complexidade e o nível de abstração dos conceitos exigem estratégias pedagógicas diversificadas. A pesquisa bibliográfica realizada para este artigo identificou duas frentes principais de trabalhos relacionados: (i) experiências didáticas e ferramentas de apoio ao ensino de grafos e computação teórica; e (ii) fundamentações teóricas modernas sobre emparelhamento e os teoremas de Tutte.

No contexto de metodologias ativas, Lassance [1] relata uma experiência similar à vivenciada na elaboração deste artigo, aplicada à disciplina de Teoria da Computação. Os autores destacam que a implementação de um Ciclo de Seminários, focando em tópicos de alta complexidade como NP-Completude, resultou na maximização da compreensão dos estudantes e no desenvolvimento da autonomia investigativa. Este artigo dá continuidade a essa visão, utilizando a metodologia de seminário para aprofundar o estudo de emparelhamentos.

Para mitigar as dificuldades de abstração, diversas abordagens visuais têm sido propostas. Santos et al. [2] discutem a validação do sistema *GraphViewer*, uma ferramenta de visualização de algoritmos focada no ensino de provas por indução em Teoria dos Grafos. Os autores evidenciam que a visualização passo a passo auxilia na compreensão de demonstrações matemáticas rigorosas. A ferramenta preenche uma lacuna específica ao focar em "demonstrações por indução", uma área onde os alunos historicamente têm grande dificuldade de visualização.

O estudo aplicou métricas rigorosas de Ganho de Aprendizagem Absoluto (*GAA*) e Normalizado (*GAN*) para medir a eficácia da ferramenta. Em realção ao aumento do desempenho, no primeiro experimento realizado, o grupo que utilizou o GraphViewer (grupo de teste) obteve um Ganho de Aprendizagem Normalizado (*GAN*) de 25,11% ,quase o dobro do ganho obtido pelo grupo de controle, e 13,92% que não usou a ferramenta. Na mesma linha, em um trabalho anterior, Santos et al. [3] apresentaram o ambiente TBC-GRAFOS, demonstrando que o uso de softwares gráficos reduz índices de reprovação e agiliza a compreensão de algoritmos clássicos, como os de busca e caminho mínimo.

Além de softwares, abordagens lúdicas também se mostram eficazes. Correa et al. [4] desenvolveram o jogo de tabuleiro "Formígrafo", que utiliza a temática de um formigueiro para motivar o aprendizado do Problema do Caminho Mínimo. O trabalho reforça que a contextualização lúdica facilita a introdução de conceitos abstratos de grafos ponderados.

No que tange à fundamentação teórica específica deste trabalho, a literatura apresenta evoluções nas demonstrações clássicas de emparelhamento. Qu e West [5] publicaram recentemente uma nova prova para a Fórmula Generalizada de Tutte-Berge aplicada a subgrafos f-limitados. O trabalho utiliza o Teorema do f-Fator de Tutte para estabelecer uma relação min-max, simplificando a compreensão da fórmula clássica quando $f(v) = 1$. Isso é, cada vértice pode estar conectado a, no máximo, uma aresta dentro desse subgrafo.

Complementarmente, o livro do Douglas B. West[6] parte de uma perspectiva mais tradicional através das técnicas de provas matemáticas puras. Contudo, encara os problemas de diversos ângulos, iniciando pelo mais tradicional, sendo ele o método da indução de vértices provando a suficiência do teorema de Tutte. Após isso, tomando uma via mais original a resolução do teorema, provando-o pelo Teorema de Hall. Aprofundando-se , a ideia geral é transformar o grafo original em um grafo bipartido adequado e então aplicar Hall.

Tais contribuições denotam a importância de encarar a teoria dos grafos com uma visão didática, a fim de facilitar a compreensão de temas abstratos. Não só isso, mas a exploração de novas maneiras de provar os teoremas mostra que o problema do emparelhamento máximo é relevante até hoje e a sua discussão e compreensão é necessária. Com isso em mente, a seguir mudaremos o enfoque para o problema em si, aprofundado nos teoremas de Tutte-Berge e Tutte.

IV. Descrição do Problema

Começando com o primeiro dos teoremas escolhidos, considerando um grafo G(S,E), sendo S o conjunto de

ISSN: 2675-3588

$$\alpha'(G) = \frac{1}{2}\min\{v(G) - (o(G-S) - |S|)\} \; tal \; que \; S \subset V$$

Figura 10: Fórmula de Tutte-Berge para emparelhamento máximo.

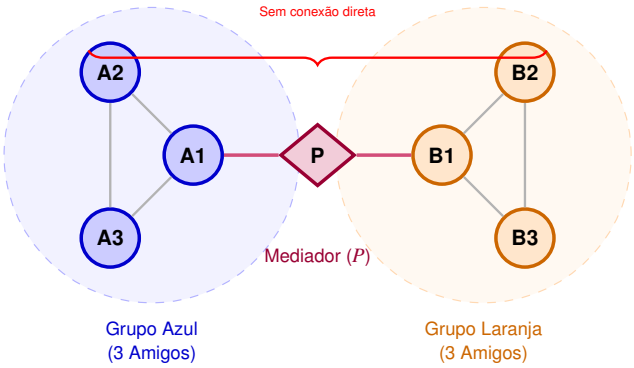


Figura 11: O Problema do Buddy System. A remoção de P cria dois componentes ímpares isolados.

vértices de G e |S| a cardinalidade do conjunto, a fórmula do teorema de Tutte-Berge é dada por:

A intuição por trás da Fórmula de Tutte-Berge baseia-se na barreira estrutural causada pela paridade dos componentes. Considere a remoção de um conjunto de vértices $U \subseteq V$. O grafo resultante $G-U$ se fragmenta em vários componentes conexos.

Com relação à natureza do problema, ele é classificado como um problema de otimização, pois determina o valor máximo de $\alpha'(G)$. O objetivo é encontrar o conjunto U que maximiza a deficiência para provar que o emparelhamento não pode ser maior.

Diferente de muitos problemas em grafos gerais (como Coloração ou Caminho Hamiltoniano) que são NP-Difíceis, o problema do Emparelhamento Máximo pertence à classe P (Tempo Polinomial). Um dos algoritmos mais eficientes conhecidos para emparelhamento máximo em grafos gerais é o algoritmo de Micali-Vazirani, cuja complexidade é $O(E\sqrt{V})$. Também, É possível resolver diversos problemas com o teorema de Tutte-Berge. Um exemplo clássico é o *Problema da Formação de Equipes* (*Buddy System*).

O problema consiste em formar o número máximo possível de duplas (emparelhamento máximo) em um grupo de pessoas, respeitando uma regra de compatibilidade: uma dupla só pode ser formada se houver uma aresta de "amizade" entre as duas pessoas.

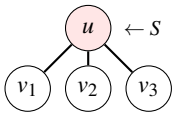
A Figura 11 ilustra um cenário onde um emparelhamento perfeito é impossível devido à estrutura social do grupo. Temos 7 pessoas divididas em: O *Grupo Azul* é composto por 3 pessoas (A1, A2 e A3), todas amigas entre si. O *Grupo Laranja* também possui 3 pessoas (B1, B2 e B3), que igualmente são amigas entre si. Por fim, há o *Mediador P*, uma pessoa central que possui uma relação de amizade com integrantes de ambos os grupos.

Há uma incompatibilidade total entre os grupos: ninguém do Azul é amigo de alguém do Laranja.

Análise via Tutte-Berge: Se removermos o conjunto $U = \{P\}$, o grafo se quebra em dois componentes conexos (os dois grupos), ambos com um número ímpar de vértices. O

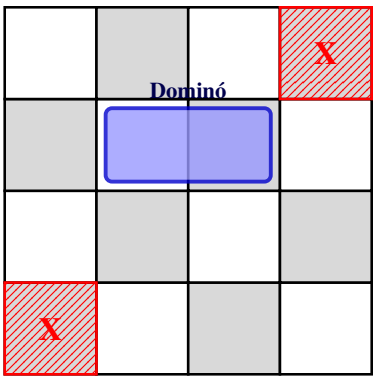
$$o(G-S) \leq |S|, \quad \forall S \subseteq V$$

Figura 12: A condição do Teorema de Tutte.



$o(G-S) = 3 > |S| = 1$
Sem Emparelhamento Perfeito

Figura 13: Grafo $K_{1,3}$ violando a condição de Tutte.



Contagem Final:
Brancas: 8 Pretas: 6
Impossível cobrir!

Figura 14: O Tabuleiro Mutilado. A remoção de dois cantos da mesma cor quebra a paridade necessária.

mediador único não é suficiente para cobrir a demanda desses componentes.

Agora, analisando o *Teorema de Tutte*, ele responde se o grafo possui um emparelhamento perfeito. A condição é apresentada na Figura 12:

Assim, um grafo possui um emparelhamento perfeito se e somente se a condição acima for cumprida para todo subconjunto S. A Figura 13 mostra uma violação simples.

Seguindo, o problema de encontrar um emparelhamento perfeito pertence à classe P. Um dos algoritmos fundamentais é o *Algoritmo de Blossom* (Edmonds, 1965), que lida com "ciclos ímpares" contraindo-os em super-vértices.

Para entender a importância da paridade, analisamos o problema do tabuleiro de xadrez 4×4 "mutilado". Removemos duas casas de cantos opostos (digamos, duas pretas). Restam 14 casas: 8 brancas e 6 pretas (Figura 14).

Para que um emparelhamento perfeito existisse (cobertura por dominós), precisaríamos de um número igual de casas brancas e pretas, pois cada dominó consome um par de cores diferentes. Como restaram 8 casas brancas e apenas 6 pretas, é impossível cobrir o tabuleiro. Com toda a formulação teórica dos teoremas explicada, podemos resumir e comparar ambos através das seguintes tabelas:

Concluindo, o teorema de Tutte e Tutte-Berge possuem diversos paralelos téoricos que serão a seguir, aprofundados e demonstrados.

TABELA 1: RESUMO ESTRUTURAL: TEOREMAS DE TUTTE E TUTTE–BERGE

Problema	Caracterização de Emparelhamentos Perfeitos (Tutte) e de Emparelhamentos Máximos (Tutte–Berge).
Input	Estrutura do grafo G: componentes ímpares após remoção de subconjuntos $S \subseteq V(G)$; método estrutural baseado em princípios Min–Max.
Output	**Tutte**: G possui emparelhamento perfeito se $o(G-S) \leq \lvert S\rvert$ para todo $S \subseteq V(G)$. **Tutte–Berge**: o tamanho máximo do emparelhamento é $\frac{1}{2}\big(\lvert V(G)\rvert - \mathrm{def}(G)\big)$.
Resumo	Caracterização Min–Max: relaciona componentes ímpares, barreiras e deficiência à estrutura de emparelhamentos; fornece condição necessária e suficiente para emparelhamentos perfeitos e fórmula exata para emparelhamentos máximos.

TABELA 2: COMPARATIVO ENTRE OS TEOREMAS DE TUTTE E TUTTE–BERGE

Aspecto	Tutte	Tutte–Berge
Objetivo principal	Determinar a existência de um emparelhamento perfeito.	Determinar o tamanho máximo de um emparelhamento em qualquer grafo.
Caracterização	Existencial: condições para a existência de emparelhamento perfeito.	Quantitativa: fornece a cardinalidade de um emparelhamento máximo.
Conceitos estruturais	Componentes ímpares e emparelhamentos perfeitos.	Componentes ímpares, emparelhamentos máximos e barreiras.

V. DEMONSTRAÇÃO E CONTRIBUIÇÕES

As demonstrações que serão apresentadas são aquelas descritas no livro *Graduate Texts in Mathematics*, conforme citam Bondy e Murty [7] a respeito do Teorema de Tutte-Berge e do Teorema de Tutte. Os teoremas em questão são a estrutura base para caracterizar emparelhamentos máximos e perfeitos apresentando condições necessárias e suficientes. Além disso, as demonstrações a seguir seguem um tratamento bem próximo do exposto por Bondy e Murty, mas apresentando-os de maneira mais clara.

O TEOREMA DE TUTTE-BERGE

Teorema V.1 (Bondy–Murty [7])**.** *O TEOREMA DE TUTTE-BERGE*
Todo grafo tem uma barreira.

Em um grafo bipartido, uma cobertura mínima constitui uma barreira do grafo. Porém, geralmente todo grafo tem uma barreira. Este fato é conhecido como o *Teorema de Tutte-Berge*. Entretanto, lembre-se de que um vértice v de um grafo G é essencial se todo emparelhamento máximo cobre v, e não essencial caso contrário. Assim, v é essencial se $\alpha'(G-v) = \alpha'(G) - 1$ e não essencial se $\alpha'(G-v) = \alpha'(G)$. Dessa maneira, temos os seguintes lemas auxiliares para o teorema:

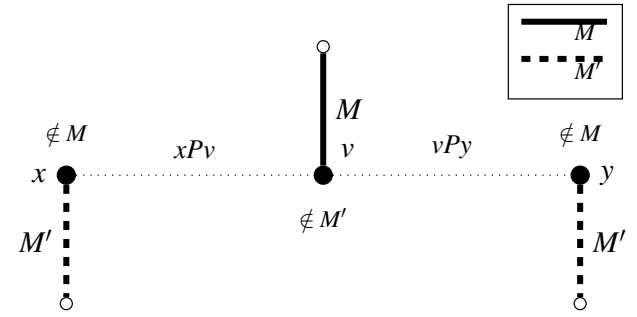


Figura 15: Ilustração da indução com o emparelhamento M' cobrindo tanto x quanto y

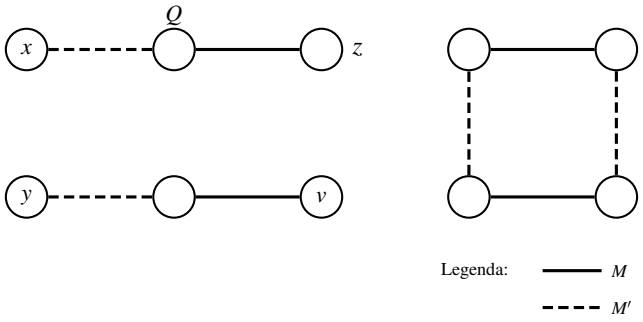


Figura 16: Ilustração dos componentes de $G[M\Delta M']$.

Lema 1 (16.8, Bondy–Murty [7])**.** O conjunto vazio é uma barreira de todo grafo hipoemparelhável.

Lema 2 (16.9, Bondy–Murty [7])**.** Seja v um vértice essencial de um grafo G e seja B uma barreira de $G-v$. Então $B \cup \{v\}$ é uma barreira de G.

Lema 3 (16.10, Bondy–Murty [7])**.** Seja G um grafo conexo no qual nenhum vértice é essencial. Então G é hipoemparelhável.

Prova Como nenhum vértice de G é essencial, G não tem um emparelhamento perfeito. Resta mostrar que todo subgrafo com um vértice removido tem um emparelhamento perfeito. Caso isso não ocorra, então cada emparelhamento máximo deixa pelo menos dois vértices descobertos. Assim, basta mostrar que para qualquer emparelhamento máximo e quaisquer dois vértices em Go emparelhamento cobre pelo menos um destes vértices. Estabelecemos isto por indução na distância entre estes dois vértices.

Considere um emparelhamento máximo M e dois vértices x e y em G. Seja xPy um caminho xy-mais curto em G. Suponha que nem x nem y são cobertos por M. Como M é máximo, P tem comprimento de pelo menos dois. Seja v um vértice interno de P. Como xPv é mais curto que P, o vértice v é coberto por M, por indução. Por outro lado, como v é não essencial, G tem um emparelhamento máximo M' que não cobre v. Além disso, como xPv e vPy são ambos mais curtos que P, o emparelhamento M' cobre tanto x quanto y, novamente por indução.

□

O vértice interno v é coberto por M (pois $xPv < P$), mas descoberto por M' (pois v não é essencial). Consequentemente, M' cobre x e y.Os componentes de $G[M\Delta M']$ são caminhos e ciclos pares cujas arestas pertencem alternadamente a M e M' .

ISSN: 2675-3588

Cada um dos vértices x, v, y é coberto por exatamente um dos dois emparelhamentos e, portanto, é uma extremidade de um dos caminhos. Como os caminhos são pares, x e y não são extremidades do mesmo caminho. Além disso, os caminhos que começam em x e y não podem ambos terminar em v.

Podemos, portanto, supor que o caminho Q que começa em x não termina nem em v nem em y. Mas então o emparelhamento $M'\Delta E(Q)$ é um emparelhamento máximo que não cobre nem x nem v, contradizendo a hipótese de indução e estabelecendo o lema.Também, deste teorema podemos deduzir um dos mais importantes Corolários dos teoremas de Tutte e Berge. Sendo ela, a *formula de Tutte-Berge*.

Antes de entrar em cálculos, serão retomadas as definições já dadas anteriormente e também, considere que o número total de vértices de um grafo G, denotado por $|V(G)|$, pode ser particionado em dois conjuntos: os vértices que são cobertos por um emparelhamento máximo M e os vértices que permanecem descobertos.

Seja $\alpha'(G) = |M|$ o tamanho do emparelhamento máximo. O número de vértices cobertos é, portanto, $2\alpha'(G)$. O número de vértices não cobertos é definido como a deficiência do grafo, denotada por $\text{def}(G)$. Assim, temos a identidade fundamental:

$$|V(G)| = 2\alpha'(G) + \text{def}(G) \tag{1}$$

Como já dito anteriormente, O Teorema de Tutte-Berge estabelece que a barreira para um emparelhamento perfeito reside na existência de um subconjunto $S \subseteq V(G)$ cuja remoção cria mais componentes ímpares do que o próprio $|S|$ consegue cobrir. Cada componente ímpar em $G-S$ deve, necessariamente, ter pelo menos um vértice não emparelhado internamente ou conectado a um vértice de S.

No pior caso (o que maximiza os vértices descobertos), a deficiência é dada por:

$$\text{def}(G) = \max_{S \subseteq V(G)} \{o(G-S) - |S|\} \tag{2}$$

Substituindo a equação (2) em (1), obtemos:

$$|V(G)| = 2\alpha'(G) + \max_{S \subseteq V(G)} \{o(G-S) - |S|\}$$

Para isolar $\alpha'(G)$, reorganizamos a equação. Note que subtrair o valor máximo de um conjunto é equivalente a somar o valor mínimo do termo negativo:

$$2\alpha'(G) = |V(G)| - \max_{S \subseteq V(G)} \{o(G-S) - |S|\}$$
$$2\alpha'(G) = \min_{S \subseteq V(G)} \{|V(G)| - (o(G-S) - |S|)\}$$

Finalmente, dividindo por 2, chegamos à fórmula do Corolário:

$$\alpha'(G) = \frac{1}{2} \min_{S \subseteq V(G)} \{|V(G)| - (o(G-S) - |S|)\} \tag{3}$$

Esta formulação confirma que o tamanho do emparelhamento máximo é determinado pela "barreira" S que minimiza a perda de vértices que não podem ser emparelhados devido à estrutura topológica do grafo.

Isso nos leva ao seguinte colário:

Corolário 1 (Bondy–Murty [7])**.** Para qualquer grafo G,

$$\alpha'(G) = \frac{1}{2} \min_{S \subseteq V(G)} \{|V(G)| - (o(G-S) - |S|)\}.$$

O resultado em questão fornece a fórmula de Tutte-Berge que caracteriza o tamanho de qualquer emparelhamento máximo em relação as barreiras correspondentes que sintetiza o *Teorema de Tutte-Berge*.

TEOREMA DE TUTTE

Teorema V.2 (Bondy–Murty [7])**.** *Um grafo G tem um emparelhamento perfeito se, e somente se,*

$$o(G-S) \leq |S| \quad \textit{para todo } S \subseteq V(G). \tag{4}$$

Prova Demonstraremos ambas as direções da equivalência.

Necessidade ($\Rightarrow$):

Suponha que G possui um emparelhamento perfeito M. Devemos mostrar que $o(G-S) \leq |S|$ para todo $S \subseteq V(G)$.

Seja $S \subseteq V(G)$ um subconjunto arbitrário de vértices. Considere o grafo $G-S$ obtido após a remoção de todos os vértices de S e suas arestas incidentes. Seja $C_1, C_2, \ldots, C_k$ o conjunto de componentes ímpares de $G-S$.

Para cada componente ímpar C_i, o número de vértices $|V(C_i)|$ é ímpar. Como M é um emparelhamento perfeito em G, todos os vértices devem estar cobertos por M. Portanto, cada componente ímpar C_i deve ter pelo menos uma aresta de M conectando um vértice interno de C_i a um vértice em S (já que um número ímpar de vértices não pode ser perfeitamente emparelhado internamente).

Formalmente: como $|V(C_i)|$ é ímpar e M é perfeito em G, existe pelo menos um vértice $v_i \in V(C_i)$ tal que v_i está emparelhado com algum vértice $s_i \in S$.

Como as arestas de M são disjuntas nos vértices (cada vértice aparece em no máximo uma aresta), os vértices $s_1, s_2, \ldots, s_k \in S$ que estão emparelhados com vértices das componentes ímpares devem ser distintos. Portanto, precisamos de pelo menos k vértices em S para cobrir todas as componentes ímpares.

Logo, $k = o(G-S) \leq |S|$, como queríamos demonstrar.

Suficiência ($\Leftarrow$):

Suponha, por contradição, que $o(G-S) \leq |S|$ para todo $S \subseteq V(G)$, mas G *não* possui emparelhamento perfeito.

Como G não possui emparelhamento perfeito, seja M^* um emparelhamento máximo de G. Seja $U \subseteq V(G)$ o conjunto de vértices não cobertos por M^*. Por hipótese, $|U| \geq 1$ (se $|U| = 0$, então M^* seria perfeito, contradição).

Pelo Teorema de Tutte-Berge (Corolário 1), existe uma barreira $B \subseteq V(G)$ tal que

$$\text{def}(G) = o(G-B) - |B| = |U|. \tag{5}$$

Como $|U| \geq 1$, temos:

$$o(G-B) - |B| \geq 1 \quad \Rightarrow \quad o(G-B) \geq |B| + 1 > |B|.$$

Tomando $S = B$, obtemos que $o(G-S) > |S|$, o que **contradiz** diretamente a hipótese de que $o(G-S) \leq |S|$ para todo $S \subseteq V(G)$.

Portanto, nossa suposição inicial estava errada, e G deve possuir um emparelhamento perfeito. □

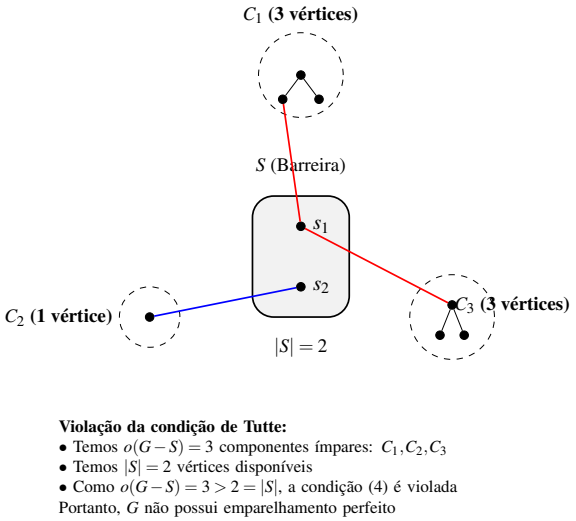


Figura 17: Ilustração da violação da condição de Tutte.

A condição de Tutte (4) é uma caracterização *necessária e suficiente* para a existência de emparelhamentos perfeitos em grafos gerais, generalizando o Teorema de Hall para grafos não-bipartidos. A desigualdade $o(G-S) \leq |S|$ captura uma restrição estrutural fundamental: cada componente ímpar resultante da remoção de S necessita de pelo menos um vértice de S para completar o emparelhamento, e não pode haver mais componentes ímpares do que vértices disponíveis em S.

A Figura 17 ilustra um caso concreto onde a condição de Tutte falha. Os componentes C_1, C_2, C_3 são componentes ímpares de $G-S$, cada um contendo respectivamente 3, 1 e 3 vértices.

Para que exista um emparelhamento perfeito em G, cada componente ímpar precisaria estar conectada a pelo menos um vértice distinto em S (pois um conjunto com número ímpar de vértices não pode ser perfeitamente emparelhado internamente). Entretanto, como temos 3 componentes ímpares mas apenas $|S| = 2$ vértices em S, é matematicamente impossível satisfazer todos os emparelhamentos necessários.

Este exemplo demonstra que a desigualdade $o(G-S) > |S|$ constitui uma *obstrução estrutural* à existência de emparelhamentos perfeitos. Finalmente, é provada o teorema de Tutte-Berge e o teorema de Tutte, que terão seus resultados novamente avaliados e pensados na próxima sessão.

VI. Resultados e Reflexões

O trabalho realizado nesse artigo considerou grafos com ênfase na análise de suas propriedades de emparelhamento. Nos casos apresentados, foram mostrados cenários que não admitem emparelhamento perfeito, de tal forma descobrindo as condições que impedem sua ocorrência. Nesse tema, conceitos como componentes ímpares, barreiras e remoção de vértices constituíram elementos fundamentais para a formulação dos critérios analisados. De maneira específica, observou-se de que forma a remoção de um subconjunto $S \subseteq V(G)$ influencia o número de componentes ímpares de $G-S$, fornecendo elementos fundamentais para a compreensão estrutural do grafo.

O principal resultado identificado corresponde à caracterização dos grafos que permitem a existência de um emparelhamento perfeito, conforme estabelecido pelo Teorema de Tutte. Esse teorema determina que um grafo G possui emparelhamento perfeito se, e somente se, para todo subconjunto $S \subseteq V(G)$, o número de componentes ímpares de $G-S$ satisfaz a relação:

$$o(G-S) \leq |S|$$

O estudo comprova que a condição de Tutte possui natureza necessária e suficiente, servindo como uma conexão entre características globais (como a presença de um emparelhamento máximo) e as propriedades locais que resultam da partição do grafo em seus componentes ímpares.

Adicionalmente, o Teorema de Tutte–Berge permitiu quantificar o tamanho de um emparelhamento máximo mesmo em situações nas quais o grafo não admite emparelhamento perfeito. Essa quantidade é dada por:

$$\alpha'(G) = \frac{1}{2} \min_{S \subseteq V(G)} \{|V(G)| - (o(G-S) - |S|)\}.$$

e a deficiência do grafo é definida por:

$$\mathrm{def}(G) = \max_{S \subseteq V(G)} (o(G-S) - |S|)$$

Esse resultado complementa o Teorema de Tutte ao oferecer uma medida precisa do grau de impossibilidade estrutural que impede o grafo de possuir um emparelhamento perfeito. A análise desses teoremas também permitiu identificar aspectos adicionais cruciais para a compreensão da teoria de emparelhamentos. Notou-se que as barreiras desempenham um papel essencial na caracterização dos grafos hipoemparelháveis, contribuindo para a análise de estruturas que impossibilitam a construção de emparelhamentos perfeitos.

Partindo para a perspectiva pedagógica, o caráter altamente figurativo das explicações do teorema e os exemplos como o Buddy System,tabuleiro multilado foram propostos como um material pedagógico para facilitar a compreensão dos alunos. Estes exemplos são feitos trazendo objetos e situações cotidianas como metáforas para a lógica dos teoremas, retirando do aluno a carga teórica que os livros didáticos possuem.

No caso do buddy system, o problema poderia ser demonstrado de forma prática dividindo a sala de aula entre times azul e laranja, também escolhendo um aluno como mediador. Dessa forma, o problema engajaria os alunos a aprofundarem seus pensamentos com a camada da experiência, escapando dos limite da teoria. Assim, o mesmo tipo de atividade pode ser feita com o tabuleiro multilado, dividindo a sala em grupos e distribuindo tabuleiros, o que motiva o aluno a ver como os grafos estão presentes no dia a dia. Isso não significa porém, o completo abandono da teoria, pois ferramentas como o *GraphViewer*[2] já citada anteriormente, permite a visualização da prova dos algoritmos o que aumenta a capacidade de aprendizado dos alunos, como demonstrado no próprio estudo.

Com tudo isso posto , a seção de considerações finais apresentará o resumo dos resultados, cumprimento dos objetivos, contribuições do estudo, limitações da pesquisa e sugestões futuras para pesquisas.

ISSN: 2675-3588

VII. Conclusões Finais

Os resultados obtidos permitiram caracterizar de maneira precisa as condições estruturais que determinam a existência ou inexistência de emparelhamentos perfeitos em grafos. O Teorema de Tutte mostrou-se fundamental nesse processo, uma vez que estabelece a relação entre o número de componentes ímpares e o tamanho dos subconjuntos de vértices removidos.

Constatou-se, além disso, que o Teorema de Tutte–Berge complementa essa análise ao quantificar o tamanho de um emparelhamento máximo mesmo quando o grafo não admite emparelhamento perfeito. Assim, verificou-se que ambos os teoremas fornecem uma descrição abrangente e operacional da estrutura dos emparelhamentos em grafos gerais.

Todos os objetivos traçados no início do estudo foram alcançados. O objetivo geral, que consistia em entender as condições que asseguram a existência de emparelhamentos perfeitos, foi cumprido por meio do exame detalhado das demonstrações e implicações dos Teoremas de Tutte e Tutte–Berge.

Os objetivos específicos também foram atendidos: o papel dos componentes ímpares foi elucidado, a noção de deficiência foi analisada como medida estrutural relevante, o conceito de barreira foi discutido no contexto de grafos hipoemparelháveis e a relação entre esses elementos e a formação de emparelhamentos máximos foi cuidadosamente explorada. Esses resultados demonstram que a investigação se desenvolveu de acordo com o que havia sido proposto.

O estudo apresenta contribuições teóricas ao sistematizar dois resultados centrais da teoria de emparelhamentos, destacando as relações entre componentes ímpares, barreiras e deficiência.A discussão reforça a relevância das formulações de Tutte para a compreensão estrutural dos grafos e evidencia a profundidade de suas implicações matemáticas.

Sob uma perspectiva prática, os resultados discutidos fornecem ferramentas analíticas importantes para problemas de alocação, otimização, modelagem combinatória e desenho de redes. A aplicabilidade dos teoremas de Tutte em diferentes áreas, como ciência da computação e pesquisa operacional, demonstra a utilidade das formulações estudadas.

Este estudo apresenta como principal limitação seu foco estritamente teórico, não abordando algoritmos computacionais para o cálculo de emparelhamentos máximos nem implementações práticas relacionadas. Além disso, não foram consideradas generalizações contemporâneas dos resultados de Tutte, como fatores k-regulares ou formulações baseadas em programação linear. Tais escolhas restringiram deliberadamente o escopo do trabalho, mantendo-o alinhado aos objetivos propostos, embora reduzam sua abrangência aplicada.

Com base nas limitações observadas, temos algumas possíveis pesquisas futuras. Uma possibilidade é explorar algoritmos eficientes para encontrar emparelhamentos máximos e perfeitos, analisando seu desempenho em grafos grandes ou específicos. Outra vertente, envolve estudar a aplicação dos teoremas em classes particulares de grafos, como bipartidos por exemplo.

Além disso, futuras pesquisas podem aprofundar as generalizações dos resultados apresentados, investigando fatores k-regulares, decomposições estruturais e conexões com métodos algébricos e combinatórios modernos. Esses caminhos podem ampliar tanto o alcance teórico quanto a aplicabilidade prática dos conceitos discutidos.

Referências

[1] Y. Lassance, G. de Barros Bianchini, and T. Dias dos Santos, "Reflexões e práticas pedagógicas no escopo da disciplina de teoria da computação," *Academic Journal on Computing, Engineering and Applied Mathematics*, vol. 6, no. 2, p. 10–17, out. 2025. [Online]. Available: https://sistemas.uft.edu.br/periodicos/index.php/AJCEAM/article/view/21791

[2] M. O. Santos, J. C. J. de Freitas, G. F. Silva, and E. L. Bispo Jr, "Validaçao de um sistema de visualizaçao de algoritmos no ensino de provas por induçao em teoria dos grafos," in *Simpósio Brasileiro de Informática na Educação (SBIE)*. SBC, 2020, pp. 1613–1622.

[3] R. P. Santos, H. A. Costa, A. M. Resende, and J. M. Souza, "O uso de ambientes gráficos para ensino e aprendizagem de estruturas de dados e de algoritmos em grafos," in *Anais do XVI Workshop sobre Educação em Computação, XXVIII Congresso da Sociedade Brasileira de Computação*. sn, 2008, pp. 157–166.

[4] A. C. Correa, A. Lyra, Y. B. Lima, and G. Xexéo, "Formígrafo: um jogo para motivar ao aprendizado de teoria de grafos," in *Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames)*. SBC, 2022, pp. 1096–1100.

[5] Z. Qu and D. B. West, "Another proof of the generalized tutte–berge formula for f-bounded subgraphs," *arXiv preprint arXiv:2307.01324*, 2023.

[6] D. B. WEST, *INTRODUCTION TO GRAPH THEORY SECOND EDITION(p.174)*. Patparganj, India: Prentice Hall, 1999.

[7] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*. London: Macmillan, 1976.

 ISSN: 2675-3588

Problemas de Agendamento: Contribuições Pedagógicas para o Aprendizado no Escopo da Teoria da Computação

Scheduling Problems: Pedagogical Contributions to Learning within the Scope of Computation Theory

Heloisa Rolins Ribeiro [1], Neci Oneides da Silva Fialho Neta [1], Daniel Martins da Silva [2] e Tanilson Dias dos Santos [1]

[1] *Universidade Federal do Tocantins, Curso de Ciência da Computação*
[2] *Universidade Federal do Norte do Tocantins,Curso de Logística*

Data de recebimento do manuscrito: 03/12/2025
Data de aceitação do manuscrito: 05/01/2026
Data de publicação: 10/02/2026

Resumo—Os problemas de agendamento constituem uma classe essencial de desafios em otimização e computação, especialmente em sistemas operacionais, processamento paralelo e aplicações em tempo real. Apesar de sua ampla utilização prática, diversas variantes permanecem computacionalmente intratáveis, mesmo sob fortes restrições estruturais. Este artigo investiga a NP-completude de duas versões específicas do problema de escalonamento: o agendamento com tempo de execução unitário e o agendamento com dois processadores em tarefas de duração igual a uma ou duas unidades. A fundamentação teórica baseia-se em reduções polinomiais clássicas, em particular a partir do problema 3-SAT, que permite codificar atribuições lógicas diretamente nas restrições de precedência e capacidade dos processadores. Além disso, transformações adicionais entre versões restritas do problema são utilizadas para preservar a equivalência estrutural das soluções. As contribuições incluem uma reconstrução didática das provas originais, a análise dos mecanismos que geram dureza computacional e uma discussão sobre as implicações práticas desses resultados em sistemas reais de escalonamento. Os resultados apresentados na literatura reforçam que mesmo cenários aparentemente simples apresentam comportamento NP-completo.

Palavras-chave—NP-completude; Agendamento; Redução polinomial; 3-SAT; Complexidade computacional.

Abstract—*Scheduling problems constitute a fundamental class of challenges in optimization and computing, particularly in operating systems, parallel processing, and real-time applications. Despite their wide practical use, many variants remain computationally intractable, even under strong structural restrictions. This article investigates the NP-completeness of two specific versions of the scheduling problem: scheduling with unit processing time and scheduling on two processors with tasks of duration one or two time units. The theoretical foundation relies on classical polynomial-time reductions, especially from the 3-SAT problem, which allows logical assignments to be encoded directly into precedence constraints and processor-capacity limitations. Furthermore, additional transformations between restricted versions of the problem are employed to preserve the structural equivalence of solutions. The contributions include a didactic reconstruction of the original proofs, an analysis of the mechanisms that give rise to computational hardness, and a discussion of the practical implications of these results in real scheduling systems. The results presented in the literature reinforce that even seemingly simple scenarios exhibit NP-complete behavior.*

Keywords—*NP-completeness; Scheduling; Polynomial reduction; 3-SAT; Computational complexity.*

I. Introdução

A Teoria da Computação estabelece os fundamentos formais para compreender os limites do que pode ser calculado de maneira eficiente. Nesse contexto, a teoria da complexidade computacional desempenha papel central ao classificar problemas quanto ao custo de suas soluções, destacando as classes $\mathcal{P}$, $\mathcal{NP}$ e NP-completo. Problemas NP-completos são aqueles para os quais não se conhece algoritmo polinomial e, ao mesmo tempo, qualquer problema em $\mathcal{NP}$ pode ser reduzido a eles em tempo polinomial. Assim, demonstrar que um problema pertence a essa classe significa evidenciar sua provável intratabilidade.

Nesse contexto, os problemas de agendamento (scheduling problems) ocupa posição de destaque. Eles modelam situações onde tarefas devem ser distribuídas ao longo do

Dados de contato: Neci Oneides da Silva Fialho Neta, neci.silva@uft.edu.br

tempo ou entre múltiplos processadores, respeitando restrições de precedência, limites de duração e capacidade. Tais problemas surgem em sistemas operacionais, manufatura, computação paralela, arquiteturas multinúcleo e otimização industrial. Entretanto, mesmo versões altamente restritas do escalonamento podem exibir comportamento computacional complexo.

O presente artigo aborda duas variantes específicas: (i) o agendamento em que todas as tarefas possuem tempo de execução unitário e (ii) o agendamento em dois processadores com tarefas de duração igual a 1 ou 2 unidades. Apesar da simplicidade aparente dessas restrições, ambas as versões são NP-completas.

O propósito deste trabalho é apresentar uma análise das provas de NP-completude desses dois problemas, contextualizando-as dentro da Teoria da Computação e explicando, passo a passo, como reduções polinomiais, especialmente a partir do problema 3-SAT, permitem codificar instâncias lógicas dentro de modelos de escalonamento. Além disso, discute-se como restrições de precedência, janelas de execução e limitações de processadores funcionam como dispositivos para simular atribuições booleanas.

As principais contribuições deste artigo são a reconstrução didática das demonstrações clássicas, tornando-as mais acessíveis a estudantes e pesquisadores; a análise conceitual dos mecanismos responsáveis pela complexidade computacional dos problemas estudados; a integração entre teoria e prática, discutindo implicações para sistemas reais de escalonamento e algoritmos modernos; e a organização clara e sistemática das relações entre as variantes do problema, destacando cadeias de reduções e interdependências.

Com isso, o artigo busca não apenas demonstrar formalmente a NP-completude das variantes analisadas, mas também oferecer uma compreensão mais profunda sobre por que tais problemas permanecem intratáveis mesmo em cenários simples.

Para organizar a discussão, o artigo está estruturado da seguinte forma: na Seção II (Preliminares), apresentam-se os conceitos preliminares necessários para compreender a complexidade dos problemas estudados, incluindo definições formais, modelos de agendamento e a cadeia de reduções utilizada. A Seção III (Trabalhos Relacionados) revisa trabalhos clássicos e contemporâneos relacionados ao tema, situando P2 e P3 no contexto mais amplo da teoria de escalonamento. A Seção IV (Descrição do Problema) descreve formalmente as variantes analisadas e suas aplicações, ilustrando seus aspectos combinatórios. Na Seção V (Demonstração e Contribuições) são desenvolvidas as provas de NP-completude de P2 e P3, com ênfase nas reduções polinomiais que conectam esses problemas ao 3-SAT. A Seção VI (Resultados e Reflexões) apresenta reflexões e interpretações sobre os resultados obtidos, destacando implicações teóricas e pedagógicas. Por fim, a Seção VII (Considerações Finais) reúne as considerações finais e aponta possíveis direções para investigações futuras.

II. PRELIMINARES

As preliminares apresentadas nesta seção têm o objetivo de estabelecer todas as definições, notações e convenções formais utilizadas ao longo deste trabalho. Como as demonstrações de NP-completude reconstruídas aqui envolvem cadeias de reduções, relações de precedência, funções de escalonamento e estruturas lógicas, é importante que os símbolos e conceitos empregados sejam apresentados de modo claro e unificado antes de aparecerem nas seções posteriores.

TABELA 1: DESCRIÇÃO FORMAL DO PROBLEMA 3-SAT.

3-SAT

Entrada: Uma fórmula booleana φ em forma normal conjuntiva,

$$\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_k,$$

onde cada cláusula possui exatamente três literais:

$$C_i = (\ell_1 \vee \ell_2 \vee \ell_3), \qquad \ell_j \in \{x, \neg x\}.$$

Objetivo: Decidir se existe uma atribuição de valores verdade às variáveis que satisfaça todas as cláusulas de φ.
Saída: SIM, se φ é satisfatível; NÃO, caso contrário.

Para iniciar, adotamos as classes de complexidade usuais da Teoria da Computação. A classe $\mathcal{P}$ contém todos os problemas de decisão solucionáveis por algoritmos determinísticos cujo tempo de execução é polinomial no tamanho da entrada. A classe $\mathcal{NP}$ reúne problemas cujas soluções podem ser verificadas em tempo polinomial por um verificador determinístico, dado um certificado apropriado. Um problema π é dito *NP-completo* se satisfaz duas condições: (i) $\pi \in \mathcal{NP}$; e (ii) para todo problema π' já conhecido por ser NP-completo, existe uma redução polinomial de π' para π. Denotamos tal redução pela notação:

$$\pi' \leq_p \pi,$$

que indica que qualquer instância de π' pode ser transformada, em tempo polinomial, em uma instância equivalente de π. Essa notação será empregada repetidas vezes ao longo deste artigo.

Como ponto de partida das reduções, utilizamos o problema 3-SAT, cuja importância histórica foi estabelecida por Cook em 1971 [1]. Empregamos as notações padrão: *variáveis booleanas* $x_1, \ldots, x_m$, que são entidades que podem assumir os valores verdadeiro ou falso; *literais* $\ell \in \{x_i, \neg x_i\}$, onde cada literal representa uma variável booleana ou sua negação; *cláusulas* $C_j = (\ell_1 \vee \ell_2 \vee \ell_3)$, que são disjunções de exatamente três literais; e *fórmulas booleanas* em forma normal conjuntiva (CNF) da forma

$$\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_n,$$

que consistem em conjunções de múltiplas cláusulas. Uma fórmula é *satisfatível* quando existe uma atribuição de valores às *variáveis booleanas* que torna todas as *cláusulas* verdadeiras. Como o problema 3-SAT é o ponto de origem da cadeia de reduções analisada, apresentamos a seguir sua definição formal apresentada na tabela 1.

Passamos agora ao modelo formal de agendamento adotado em todas as variantes do problema. Uma instância é composta por um conjunto de tarefas $S = \{J_1, \ldots, J_n\}$, uma relação parcial $<$ indicando precedências (por exemplo, $J < J'$ significa que J deve terminar antes do início de

 ISSN: 2675-3588

J'), uma função duração $W : S \to \mathbb{Z}^+$ que associa a cada tarefa seu tempo de execução em unidades de tempo, um número de processadores k, e um horizonte de tempo t. Um escalonamento é descrito por uma função

$$f : S \to \{0, 1, \ldots, t-1\},$$

que define para cada tarefa seu instante de início dentro do intervalo de tempo disponível, onde $f(J)$ determina o instante de início de J. Esse escalonamento é válido quando satisfaz: (i) $f(J) + W(J) \leq f(J')$ sempre que $J < J'$; (ii) em cada unidade de tempo, no máximo k tarefas executam simultaneamente; e (iii) todas as tarefas terminam antes do tempo limite t. Essa notação será usada constantemente nas definições e nas construções das reduções.

As variantes específicas de escalonamento reconstruídas neste trabalho são as mesmas introduzidas por Ullman (1975) [2]. O problema P2 consiste em escalonar tarefas com duração unitária sobre k processadores, sob um conjunto arbitrário de precedências. O problema P3 envolve dois processadores e tarefas cujas durações pertencem ao conjunto $\{1,2\}$. O problema P4 é semelhante a P2, exceto pelo fato de que, em vez de um número fixo de processadores, cada unidade de tempo possui uma capacidade própria $c_0, c_1, \ldots, c_{t-1}$. Já o problema P5 corresponde a uma versão onde todos os processadores devem permanecer ocupados durante toda a execução, isto é, exatamente k tarefas devem estar em execução em cada instante.

Como as reduções de Ullman empregam estruturas visuais e padrões temporais específicos, adotamos também notações auxiliares internas às construções. Cadeias verticais de tarefas representam literais ou suas negações; barras sobre variáveis, como $\bar{x}_i$, indicam negação; blocos D_{ij} agrupam tarefas associadas a cláusulas ou a estruturas auxiliares; e, na redução para P3, os termos *banda* e *quebra* designam segmentos longos e curtos de execução no segundo processador, respectivamente. O par de tarefas J' e J, ambas de duração 2, será utilizado para preservar precedências ao converter instâncias de P5 para P3. Finalmente, o termo *certificado* será entendido sempre como um escalonamento candidato cuja verificação é realizada em tempo polinomial.

Com todas essas convenções estabelecidas, apresentamos a cadeia de reduções que estrutura a demonstração da NP-completude dos problemas estudados:

$$\text{3-SAT} \leq_p P4 \leq_p P5 \leq_p P2, \qquad P5 \leq_p P3.$$

Cada ocorrência do símbolo $\leq_p$ será detalhada nas seções subsequentes, com construções explícitas e demonstrações de validade. Assim, esta seção reúne todo o aparato matemático necessário para sustentar as provas desenvolvidas ao longo do artigo.

III. Trabalhos Relacionados

Os estudos sobre a complexidade de problemas de escalonamento possuem uma trajetória consolidada na literatura, e o presente trabalho se insere nesse contexto ao analisar variantes restritas que permanecem NP-completas. O trabalho de referência fundamental é o de Ullman [2], cujo objetivo foi demonstrar formalmente que versões simplificadas do problema de scheduling continuam a exibir dureza combinatória. Por meio de reduções formais iniciadas em 3-SAT, o autor constrói progressivamente instâncias dos problemas P4, P5, P2 e P3, utilizando cadeias de tarefas, precedências rígidas e janelas de execução que representam diretamente a lógica das fórmulas booleanas. Seu principal resultado é estabelecer que tanto o agendamento com tempos unitários quanto o agendamento em dois processadores com tarefas de duração 1 ou 2 são NP-completos, servindo como base teórica direta para as análises reconstruídas neste artigo.

O survey clássico de Graham, Lawler, Lenstra e Rinnooy Kan [3] também está intimamente relacionado a este trabalho. Seu objetivo foi organizar e classificar modelos determinísticos de escalonamento, descrevendo algoritmos, limites de complexidade, estruturas de precedência e resultados de aproximação. A metodologia consiste em sistematizar o campo usando a notação de três campos $(\alpha|\beta|\gamma)$, além de situar diversos problemas dentro de categorias de tratabilidade ou NP-dificuldade. O survey demonstra que a interação entre precedências e múltiplas máquinas é uma das principais fontes de intratabilidade, o que contextualiza de maneira abrangente os problemas P2 e P3 analisados aqui.

Outro trabalho relevante é o de Brucker e Kravchenko [4], cujo foco é o escalonamento em máquinas paralelas quando todos os tempos de processamento são iguais. Seu objetivo foi investigar como a presença de precedências e janelas temporais afeta a complexidade do problema. Por meio de reduções polinomiais baseadas em problemas clássicos de particionamento, os autores demonstram que mesmo instâncias homogêneas tornam-se NP-difíceis quando combinadas com dependências. Essa conclusão reforça diretamente o caso de P2 estudado neste artigo.

Também se destaca a obra de Pinedo [5], cujo objetivo é oferecer uma visão abrangente dos modelos de escalonamento utilizados em sistemas industriais, computacionais e de produção. É uma referência técnica essencial para compreender como modelos com múltiplas máquinas, precedências e janelas temporais se comportam na prática, contextualizando os cenários teóricos tratados neste trabalho.

Por fim, o trabalho de Baptiste, Leung e Smith [6] aprofunda limites de complexidade em modelos de escalonamento com restrições de precedência, janelas de disponibilidade e múltiplas máquinas. Seu objetivo é mapear rigorosamente a fronteira entre casos polinomiais e NP-difíceis, empregando técnicas de construção temporal similares às utilizadas por Ullman. Seus resultados mostram que até variantes aparentemente simples tornam-se NP-completas quando precedências e tempos variados interagem, conectando-se diretamente às reduções que caracterizam P3.

Coletivamente, esses estudos situam claramente P2 e P3 dentro do panorama teórico do escalonamento, reforçando que tais variantes representam casos emblemáticos na fronteira entre tratabilidade e intratabilidade na Teoria da Computação.

Em complemento a esses estudos específicos de escalonamento, a monografia clássica de Garey e Johnson [7] fornece o pano de fundo teórico geral sobre NP-completude e técnicas de redução polinomial. Embora trate de uma ampla variedade de problemas e não se concentre exclusivamente

TABELA 2: DESCRIÇÃO FORMAL DO PROBLEMA P2.

P2 – Escalonamento com tempo de execução unitário
Entrada: Um conjunto de n tarefas, cada uma levando exatamente 1 unidade de tempo para ser concluída. Há relações de precedência entre algumas tarefas, existem m processadores idênticos disponíveis e um tempo máximo total D para executar todas elas.
Objetivo: Decidir se existe uma forma de agendar todas as tarefas nos m processadores de modo que todas sejam concluídas até o tempo limite D.
Saída: SIM, se existe um escalonamento que termina todas as tarefas dentro de D; NÃO, caso contrário.

TABELA 3: DESCRIÇÃO FORMAL DO PROBLEMA P3.

P3 – Escalonamento com tempos de execução variados
Entrada: Um conjunto de n tarefas, cada uma com um tempo de execução definido. Há relações de precedência entre certas tarefas. Existem m processadores idênticos disponíveis e um tempo limite total D para concluir todas as tarefas.
Objetivo: Determinar se existe um escalonamento válido que aloque todas as tarefas aos m processadores de forma a respeitar as precedências e terminar tudo até o tempo D.
Saída: SIM, se existe tal escalonamento dentro de D; NÃO, caso contrário.

em modelos de escalonamento como P2 e P3, essa obra é uma referência útil para o enquadramento conceitual deste trabalho, especialmente no que diz respeito à definição formal das classes $\mathcal{P}$, $\mathcal{NP}$ e dos problemas NP-completos.

Além da literatura técnica sobre escalonamento, este trabalho também se apoia em produções pedagógicas da área de Teoria da Computação. O artigo de Lassance et al. [8] discute práticas de ensino envolvendo decidibilidade, NP-completude e transformações polinomiais, oferecendo uma base didática que auxiliou na organização conceitual dos fundamentos teóricos utilizados, ainda que não trate diretamente dos problemas de scheduling analisados aqui.

IV. DESCRIÇÃO DO PROBLEMA

Os problemas de agendamento tratam da organização de um conjunto de tarefas ao longo do tempo ou entre múltiplos recursos, respeitando restrições estruturais como precedência, duração e capacidade de processamento. Em sua formulação clássica, busca-se determinar em que momento cada tarefa deve ser executada, de modo a cumprir dependências e limitações de recursos, garantindo que todas sejam concluídas antes de um tempo máximo permitido. As variantes analisadas neste trabalho — o agendamento com tempo de execução unitário (P2) e o agendamento em dois processadores com tarefas de duração igual a uma ou duas unidades (P3) — representam versões restritas desse modelo geral, mas preservam a complexidade combinatória presente em cenários mais amplos.

A definição formal do problema de escalonamento com tempo unitário é apresentada na tabela 2.

Embora o problema P2 trate exclusivamente de tarefas com duração unitária, o que permite certas simplificações em sua análise estrutural, muitas aplicações práticas exigem considerar tarefas com tempos distintos de execução. Essa generalização leva naturalmente à formulação do problema P3, apresentada a seguir na tabela 3.

Enquanto P2 e P3 representam variantes fundamentais do modelo de escalonamento com restrições de precedência e limite global de tempo, a análise de sua complexidade costuma recorrer a versões intermediárias mais expressivas. Entre elas destacam-se os problemas P4 e P5, que introduzem novos elementos — como tempos de liberação e prazos individuais — permitindo construir reduções mais detalhadas e modularizadas ao longo da prova de NP-completude.

TABELA 4: DESCRIÇÃO FORMAL DO PROBLEMA P4.

P4 – Escalonamento com tempos de liberação
Entrada: Um conjunto de n tarefas, cada uma com um tempo de duração e um instante mínimo no qual está autorizada a começar; algumas tarefas devem ocorrer antes de outras; há m processadores idênticos disponíveis; e existe um limite total D para finalizar todas as tarefas.
Objetivo: Determinar se há uma forma de escalonar todas as tarefas nos m processadores, respeitando os tempos de liberação, as precedências e o tempo máximo permitido.
Saída: SIM, se existe um escalonamento válido dentro de D; NÃO, caso contrário.

TABELA 5: DESCRIÇÃO FORMAL DO PROBLEMA P5.

P5 – Escalonamento com precedências arbitrárias
Entrada: Um conjunto de n tarefas, cada uma com tempo de duração e um prazo individual; um conjunto de dependências indicando quais tarefas devem anteceder outras; e m processadores idênticos disponíveis.
Objetivo: Determinar se existe um escalonamento que respeite tanto os prazos individuais como todas as dependências entre as tarefas.
Saída: SIM, se existe um escalonamento válido que satisfaça todos os prazos e dependências; NÃO, caso contrário.

Para estabelecer a complexidade computacional dos problemas P2 e P3, utilizamos dois problemas intermediários nas reduções, conforme proposto por Ullman [2]: P4 (escalonamento com tempos de liberação),definido formalmente na tabela 4, e P5 (escalonamento com precedências arbitrárias),apresentado na tabela 5. Estes servem como etapas intermediárias na cadeia de reduções que parte do problema 3-SAT e culmina na demonstração de NP-completude de P2 e P3.

Embora P4 e P5 compartilhem a estrutura básica de problemas de escalonamento, diferenciam-se pelas restrições específicas que impõem. Enquanto P4 introduz *tempos de liberação* como restrições adicionais ao início das tarefas, P5 generaliza as relações de precedência e incorpora *prazos individuais* para cada tarefa. Essa progressão na complexidade das restrições é fundamental para a cadeia de reduções, permitindo que se estabeleça a NP-dificuldade de P2 através de transformações sucessivas partindo do 3-SAT.

Uma forma intuitiva de visualizar esses problemas é por

ISSN: 2675-3588

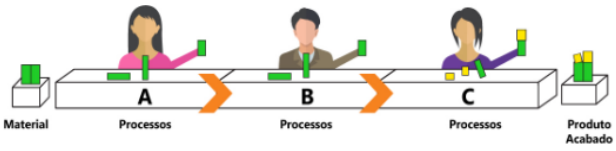


Figura 1: Linha de produção ilustrando dependências, capacidade limitada e fluxo sequencial — elementos que caracterizam os problemas de escalonamento P2 e P3.

meio de uma linha de produção simplificada, como ilustrado na Figura 1. Nessa representação, o material bruto entra pela esquerda e percorre três etapas de processamento (A, B e C) até tornar-se produto acabado. Cada estação representa um conjunto de tarefas que deve ser executado em uma ordem específica, pois A precisa concluir sua parte antes que B possa começar, e o mesmo vale para a transição de B para C. Essa metáfora captura precisamente a ideia de restrições de precedência presentes nos problemas de escalonamento. Além disso, cada operador da linha só consegue manipular uma peça por vez, analogamente ao limite de capacidade dos processadores ou máquinas em um modelo computacional. Ao observarmos a linha em funcionamento, percebemos que, mesmo que as peças tenham tamanhos semelhantes, pequenas dependências ou variações de duração podem provocar bloqueios, esperas desnecessárias ou gargalos — efeitos que modelam diretamente a complexidade de P2 e P3.

No contexto dessa metáfora, o problema P2 corresponde a uma situação em que todas as tarefas duram exatamente uma unidade de tempo. Isso seria equivalente a imaginar que cada operador leva sempre o mesmo tempo para processar qualquer peça que receba. Ainda assim, dependências rígidas entre etapas podem impedir um fluxo contínuo, e o desafio consiste em verificar se existe uma forma de organizar essas execuções dentro de um limite global de tempo. Já o problema P3 se aproxima de uma linha de produção com dois operadores trabalhando simultaneamente, mas com tarefas que podem durar uma ou duas unidades de tempo. Nesse cenário, algumas peças exigem mais trabalho em uma etapa específica, o que gera desequilíbrios e requer um planejamento cuidadoso para evitar que o segundo operador fique sobrecarregado ou ocioso em momentos críticos.

Esses modelos, embora simples, surgem naturalmente em sistemas operacionais, computação paralela, engenharia industrial e processamento em tempo real. A analogia da linha de produção evidencia de forma clara como neles coexistem dois fatores cruciais: a necessidade de obedecer a dependências estritas e a limitação de recursos. Mesmo exemplos cotidianos, como essa sequência organizada de operações A→B→C, são suficientes para ilustrar como a ordem de execução e a duração das tarefas influenciam diretamente a viabilidade de um cronograma. Basta imaginar um operador ficando sem peças para trabalhar devido ao atraso na etapa anterior — um fenômeno equivalente à espera imposta pela precedência entre tarefas — ou dois operadores disputando o processamento simultâneo de peças, representando o conflito pela capacidade dos processadores.

Assim, a metáfora da linha de produção ajuda a visualizar por que os problemas P2 e P3, apesar de parecerem simples, capturam estruturas lógicas suficientemente ricas para simular decisões combinatórias complexas. Em particular, a interação entre tempos de execução, dependências e recursos limitados cria padrões que não apenas se aproximam do fluxo real de sistemas industriais, mas também constituem o núcleo das dificuldades teóricas que tornam esses problemas NP-completos.

V. Demonstração e Contribuições

Nesta seção apresentamos, as provas de NP-completude das duas variantes de escalonamento estudadas: o problema de tempo de execução unitário (P2) e o problema de dois processadores com tarefas de duração 1 ou 2 (P3). A demonstração é organizada em duas etapas principais para cada problema: (i) prova de NP-pertinência (isto é, mostrar que o problema pertence à classe $\mathcal{NP}$) e (ii) prova de NP-dificuldade (isto é, mostrar que o problema é ao menos tão difícil quanto um problema base conhecido como NP-completo).

a. P2 ∈ NP e P3 ∈ NP

Para demonstrar que os problemas P2 e P3 pertencem à classe $\mathcal{NP}$, utilizamos o conceito de verificador polinomial. Em ambos os casos, o certificado natural é um possível escalonamento das tarefas: para cada tarefa J, o certificado descreve o instante de início $f(J)$ e, no caso de P3, também o processador onde ela é executada.

Dado esse certificado, o verificador deve apenas conferir se o escalonamento obedece a todas as restrições impostas pelo problema. O procedimento consiste em:

1. verificar se cada tarefa termina antes do limite de tempo t;
2. verificar todas as relações de precedência, confirmando que, para cada $J < J'$, o término de J ocorre antes ou no momento do início de J';
3. para cada unidade de tempo, contar quantas tarefas estão sendo executadas simultaneamente, garantindo que esse valor não ultrapassa o número de processadores disponíveis (em P3, exatamente dois).

Cada uma dessas verificações pode ser feita em tempo polinomial no número de tarefas e no número de relações de precedência. Especificamente, a validação das precedências requer verificar cada relação individualmente, com complexidade $O(|<|)$, onde $|<|$ é o número de relações de precedência. A verificação de sobrecarga por unidade de tempo pode ser feita em $O(n \cdot t)$, mantendo-se contadores para cada instante. Não é necessário explorar todas as possíveis execuções, mas apenas validar a execução proposta no certificado. Portanto, existe um verificador determinístico polinomial tanto para P2 quanto para P3, o que implica:

$$P2 \in NP \qquad \text{e} \qquad P3 \in NP.$$

b. P2 ∈ NP-Difícil

A prova de NP-dificuldade de P2 segue a estratégia de reduzir um problema clássico NP-completo, o 3-SAT, a uma instância de escalonamento com tempo de execução unitário.

A ideia é construir um conjunto de tarefas, precedências e limites de tempo tais que:

3-SAT é satisfatível $\iff$ P2 admite escalonamento válido.

Como abordado na Seção IV (Descrição do Problema), a demonstração completa é feita em etapas, por meio de P4 e P5.

Etapa 1: 3-SAT $\leq_p$ P4

Define-se inicialmente um problema intermediário, P4, que é uma versão do escalonamento com tempo unitário, mas com número de processadores variando ao longo do tempo. Em vez de um k fixo, P4 recebe uma sequência de capacidades $c_0, c_1, \ldots, c_{t-1}$, indicando quantas tarefas podem ser executadas em cada unidade de tempo.

A redução 3-SAT $\leq_p$ P4 constrói um conjunto de tarefas cuja viabilidade de escalonamento reflete diretamente a satisfatibilidade da fórmula. Para cada variável x_i são criadas duas cadeias disjuntas, uma associada a x_i e outra a $\neg x_i$, ligadas por restrições de precedência internas. A escolha de qual cadeia iniciar primeiro, e de qual literal será "verdadeiro", é forçada por tarefas auxiliares y_i e $\neg y_i$ que, via precedências e pela ocupação precisa dos slots de capacidade impostos pela sequência c_t, garantem que exatamente uma das duas cadeias progrida.

Em seguida, para cada cláusula C_j com três literais, introduz-se um bloco de sete tarefas $D_{j1}, \ldots, D_{j7}$ e arestas de precedência que as conectam às cadeias de variáveis. O instante crítico de execução dessas tarefas só pode ser alcançado se pelo menos uma das cadeias correspondentes a literais verdadeiros já estiver sido escalonada; caso contrário, a capacidade disponível naquele momento torna-se insuficiente e o escalonamento quebra. A sequência de capacidades $c_0, \ldots, c_{t-1}$ é escolhida de forma a apertar o espaço de processamento: em cada unidade de tempo o número de posições é exatamente o necessário para comportar as tarefas "verdadeiras" e as auxiliares, de modo que qualquer desvio da codificação correta — isto é, qualquer tentativa de satisfazer simultaneamente x_i e $\neg x_i$ ou de falsificar todas as cláusulas — impede a conclusão de todas as tarefas dentro do horizonte dado.

A complexidade desta construção é polinomial: para uma instância de 3-SAT com m variáveis e n cláusulas, o número total de tarefas geradas é da ordem de $O(m^2+n)$, assim como o número de relações de precedência. A construção pode ser implementada por algoritmos que percorrem variáveis e cláusulas com laços aninhados de profundidade constante, resultando em tempo polinomial no tamanho da fórmula original.

Comentário: a construção faz com que "rodar" certas tarefas em tempos específicos corresponda exatamente a atribuir verdadeiro ou falso às variáveis. Se a fórmula é satisfatível, existe um modo de encaixar todas as tarefas dentro do limite de tempo; se não é, faltará espaço em algum instante, e o escalonamento será impossível.

A Figura 2 ilustra a estrutura típica utilizada na redução 3-SAT $\rightarrow$ P4. Cada literal é convertido em uma cadeia vertical de tarefas — cadeias sem barra representam literais positivos, enquanto cadeias com barra representam negativas. Os blocos D_{ij} funcionam como pontos de verificação para cada

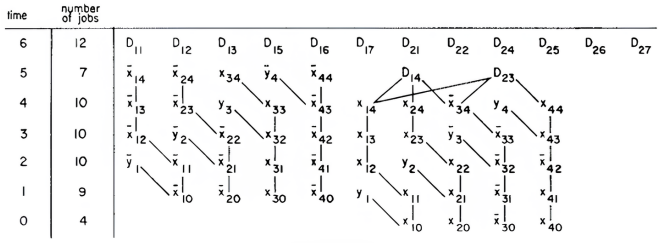


Figura 2: Estrutura geral da redução de uma instância de 3-SAT para o problema P4. Figura retirada de Ullman [2].

cláusula, garantindo que apenas escalonamentos compatíveis com uma atribuição satisfatória permitam preencher os instantes críticos impostos pelas capacidades temporais. Embora vários blocos apareçam na figura, apenas um bloco D_{ij} por cláusula desempenha o papel de validar a cláusula; os demais são estruturas auxiliares introduzidas para forçar o alinhamento temporal da construção.

Para tornar a construção mais intuitiva, a Figura 2 apresenta um exemplo concreto de como uma fórmula 3-SAT é convertida em cadeias de tarefas no problema P4. A fórmula booleana correspondente ao diagrama é:

$$\varphi = (x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_3 \vee x_4).$$

Nessa representação, cada literal aparece como uma cadeia vertical de tarefas. Cadeias sem barra correspondem ao literal positivo (como x_{14}, x_{24}, x_{34}), enquanto cadeias com barra representam o literal negado (como $\bar{x}_{33}, \bar{x}_{32}, \bar{x}_{31}$). As relações de precedência ligam os elementos de cada cadeia, garantindo que a posição temporal em que uma tarefa pode ser executada codifica a escolha "verdadeiro" ou "falso" para cada variável.

Além das cadeias de literais, o diagrama inclui vários blocos D_{ij}. Esses blocos têm funções distintas dentro da redução. Apenas alguns deles correspondem diretamente às cláusulas da fórmula; os demais fazem parte da estrutura geral da construção e servem para controlar capacidade temporal, sincronizar cadeias ou criar janelas de execução obrigatórias. Assim, embora muitos blocos apareçam no diagrama, somente dois deles representam efetivamente as cláusulas da fórmula de exemplo.

Para cada cláusula C_j, Ullman [2] insere exatamente um bloco D_{ij} que atua como ponto de verificação: esse bloco só pode ser executado caso pelo menos um dos três literais da cláusula tenha sido marcado como verdadeiro pela estrutura de precedência construída. A indexação segue o padrão usado no artigo: o índice i refere-se à variável principal associada ao bloco, enquanto j indica a cláusula da qual aquele bloco participa.

No exemplo da figura, as duas cláusulas da fórmula aparecem como: D_{14} (cláusula 1), D_{23} (cláusula 2).

Os demais blocos, como $D_{11}, D_{12}, \ldots, D_{27}$, não representam cláusulas. Eles compõem apenas a estrutura auxiliar da redução e não têm correspondência com fórmulas booleanas; funcionam como "slots de tempo" usados para forçar a organização correta das cadeias de variáveis.

Dessa forma, a Figura 2 ilustra como cada literal, cada cláusula e cada restrição temporal são traduzidos para tarefas do problema P4, permitindo que a satisfatibilidade de φ

ISSN: 2675-3588

seja refletida diretamente na existência de um escalonamento viável.

Etapa 2: P4 $\leq_p$ P5

O problema P5 é definido como uma versão de P2 com a restrição adicional de que todos os processadores devem estar ocupados em todas as unidades de tempo; isto é, o número de tarefas é exatamente $n = kt$ e o escalonamento deve preencher completamente a capacidade disponível.

Para reduzir P4 a P5, transforma-se o cronograma de capacidades variáveis $c_0, \ldots, c_{t-1}$ em um quadro de k processadores constantes simplesmente completando, em cada instante i, as $k - c_i$ posições ociosas com tarefas "de preenchimento". Essas tarefas são introduzidas em número exato para que, em cada unidade de tempo, o total de tarefas ativas seja exatamente k; além disso, elas são conectadas por uma única cadeia de precedências lineares que as obriga a executar sequencialmente, impedindo que sobreponham ou conflitem com as tarefas originais. Como suas durações são unitárias e suas janelas de execução são rigidamente controladas, qualquer escalonamento válido de P5 descarta automaticamente as tarefas de preenchimento e recupera, nos instantes restantes, um escalonamento válido para P4; reciprocamente, todo escalonamento de P4 pode ser estendido a um de P5 incluindo as tarefas de preenchimento nos slots vazios.

Esta transformação é polinomial: o número de tarefas de preenchimento adicionadas é proporcional à diferença entre a capacidade máxima e o número de tarefas já previstas em P4. Como o horizonte de tempo t e as capacidades c_i são limitados por funções polinomiais no tamanho da instância de P4, o tempo de construção é polinomial.

Comentário: essa etapa padroniza a capacidade ao longo do tempo, transformando capacidades variáveis em um número fixo de processadores que precisam estar sempre ocupados.

Etapa 3: P5 $\leq_p$ P2

Por fim, observa-se que P5 é apenas um caso particular de P2: trata-se do mesmo problema de escalonamento com tempo de execução unitário, mas com a condição adicional de $n = kt$. Portanto, qualquer instância de P5 é uma instância de P2 com uma restrição extra, e a redução é imediata, com complexidade linear no tamanho da instância de P5.

Juntando as etapas, temos:

$$\text{3-SAT} \leq_p P4 \leq_p P5 \leq_p P2,$$

onde cada redução é computável em tempo polinomial, o que implica que P2 é NP-difícil. Como já foi mostrado que P2 pertence a NP, conclui-se que P2 é NP-completo.

c. P3 $\in$ NP-Difícil

Para demonstrar que P3 é NP-difícil, utiliza-se P5 como problema intermediário. A ideia é reduzir uma instância de P5 para uma instância de P3, de forma que:

$$\text{P5 é solucionável} \iff \text{P3 construída é solucionável.}$$

A construção explora a presença de dois processadores e tarefas com pesos 1 ou 2 para criar um padrão de *bandas* e *quebras* no segundo processador.

Etapa: P5 $\leq_p$ P3

Dada uma instância de P5 com tempo limite t, número de processadores k e conjunto de tarefas S de tamanho kt parcialmente ordenado por $<$, constrói-se uma instância de P3 sobre dois processadores de velocidade $s = 2$ da seguinte forma.

Primeiro, cria-se uma sequência contínua de tarefas X_i de peso 1 que ocupam exclusivamente o primeiro processador durante todo o horizonte t, impedindo qualquer outra tarefa de executar nele. Em seguida, no segundo processador, dispõem-se tarefas Y_{ij} também de peso 1 de modo a gerar um padrão regular de "quebras" e "bandas": após cada intervalo de $2k$ unidades de tempo livres (banda), insere-se uma única unidade de tempo ocupada (quebra), repetindo esse ciclo até cobrir o horizonte total.

Cada tarefa original $J \in S$ é substituída por um par (J', J), ambas de peso 2. A precedência $J' \prec J$ garante que J só possa iniciar após J' terminar, e as relações $J \prec K$ do conjunto original tornam-se $J \prec K'$ na nova instância, preservando a ordem parcial. O peso 2 impede que qualquer dessas tarefas seja executada durante uma quebra (apenas uma unidade de tempo livre); portanto, J' e J são forçadas a se alinharem inteiramente dentro de uma banda de $2k$ unidades consecutivas. Como cada banda oferece exatamente $2k$ unidades de capacidade e o número total de tarefas de peso 2 é $2kt$, o preenchimento completo de todas as bandas corresponde biunivocamente a um escalonamento válido de P5: cada par (J', J) posicionado numa banda representa a execução da tarefa original J num dos k processadores de P5, enquanto as quebras funcionam como divisores naturais entre os k instantes de tempo.

Esta construção é polinomial: o número total de tarefas em P3 é limitado por uma função polinomial no número de tarefas e no horizonte de tempo da instância de P5. Especificamente, são criadas $O(kt)$ tarefas, e a construção pode ser implementada por laços que percorrem o conjunto de tarefas originais e o intervalo de tempo sem recursão excessiva.

O efeito desta construção é o seguinte: as tarefas X_i garantem que o primeiro processador esteja sempre ocupado, enquanto as tarefas Y_{ij} consomem parte do tempo do segundo processador, de modo que restam segmentos de tempo contínuos (bandas) suficientemente longos apenas para acomodar as tarefas de peso 2 (J) e pequenos espaços (quebras) onde se encaixam as tarefas J'. Adicionalmente, devido às dependências, cada tarefa J deve ser executada na banda correspondente ao instante em que seria processada na solução de P5, enquanto J' ocupa a quebra associada.

Com isso, qualquer solução para a instância de P3 construída induz um escalonamento válido para a instância original de P5 (interpretando cada banda como uma unidade de tempo de P5). Reciprocamente, qualquer solução de P5 pode ser "expandida" para uma solução de P3, alocando J' nas quebras e J nas bandas apropriadas.

Comentário: as bandas funcionam como "janelas compactadas" que representam cada unidade de tempo da

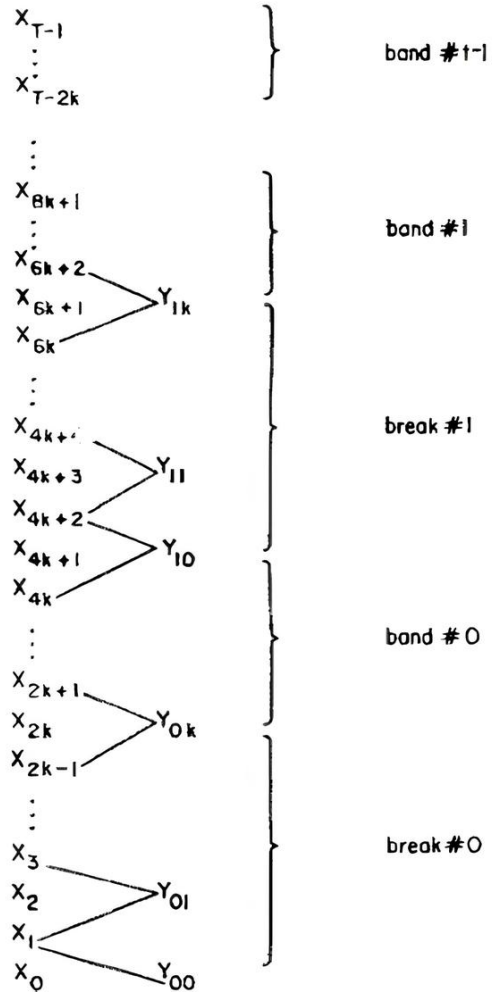


Figura 3: Organização das tarefas na redução do problema P5 para P3. Figura retirada de Ullman [2].

instância original de P5. Preencher corretamente essas bandas com tarefas de peso 2 equivale a decidir, para cada unidade de tempo, quais tarefas estão sendo executadas no modelo com k processadores.

A Figura 3 mostra como a redução de P5 para P3 utiliza dois processadores para reproduzir o comportamento temporal de uma instância original. O primeiro processador permanece ocupado continuamente pelas tarefas X_i, enquanto o segundo alterna entre segmentos curtos (*quebras*), que acomodam as tarefas J', e segmentos longos (*bandas*), nos quais são escalonadas as tarefas J de duração 2. Cada banda corresponde exatamente a uma unidade de tempo da instância de P5, preservando precedências e garantindo que o escalonamento resultante em P3 reflita corretamente a execução original em P5.

Como P5 é NP-completo e P5 $\leq_p$ P3 com complexidade polinomial, segue que P3 é NP-difícil. Já demonstramos anteriormente que P3 pertence a NP, portanto P3 é NP-completo.

d. Comentários Finais sobre a Demonstração

As provas apresentadas mostram que tanto P2 quanto P3 não são apenas variantes artificiais, mas modelos ricos o suficiente para simular a lógica de um problema canônico como o 3-SAT. As construções utilizadas exploram intensivamente a codificação de variáveis e atribuições como escolhas de tarefas executadas em tempos específicos, o uso de precedência para impor dependências lógicas, e o controle do número de processadores (ou da capacidade por unidade de tempo) para forçar o preenchimento exato de janelas de execução.

Esses mecanismos fazem com que qualquer tentativa de encontrar um algoritmo geral e eficiente para P2 ou P3 esbarre na mesma dificuldade encontrada para 3-SAT e outros problemas NP-completos. Assim, as demonstrações de NP-completude justificam, do ponto de vista teórico, o uso de heurísticas e algoritmos aproximados em problemas práticos de escalonamento.

VI. Resultados e Reflexões

A análise das variantes P2 e P3 permitiu confirmar formalmente sua classificação como problemas NP-completos, por meio da reconstrução detalhada das reduções apresentadas por Ullman (1975). A replicação dessas provas evidenciou como estruturas aparentemente simples (tarefas de duração unitária, dois processadores e precedências básicas) são suficientes para simular o comportamento lógico de fórmulas booleanas. Esse resultado reforça um dos princípios fundamentais da Teoria da Computação: a dificuldade computacional não depende apenas da complexidade aparente do modelo, mas da capacidade de representar decisões combinatórias por meio das restrições do problema.

Do ponto de vista metodológico, o processo revelou desafios relevantes. A cadeia de reduções 3-SAT $\rightarrow$ P4 $\rightarrow$ P5 $\rightarrow$ P2 exigiu um entendimento cuidadoso das construções intermediárias, especialmente na definição das capacidades variáveis de P4 e no uso das tarefas auxiliares que forçam a codificação de variáveis e cláusulas. Já a redução P5 $\rightarrow$ P3 mostrou-se particularmente difícil devido à alternância entre "bandas" e "quebras", que exige atenção à sincronização temporal e ao mapeamento entre segmentos de tempo e precedências. Esses aspectos demonstram que provas de NP-completude vão além de manipulações algébricas: tratam-se de construções conceituais sofisticadas que demandam rigor, visualização estrutural e compreensão profunda dos modelos envolvidos.

As contribuições deste trabalho situam-se tanto no campo da compreensão teórica quanto no campo pedagógico. Ao reorganizar e explicar as reduções de forma sistemática, com figuras, comentários e interpretações intuitivas, o estudo oferece um material mais acessível a estudantes e pesquisadores que desejam compreender NP-completude aplicada a problemas de escalonamento. Além disso, ao contextualizar os problemas P2 e P3 dentro da Teoria da Computação, o trabalho evidencia como reduções podem servir como ferramenta para analisar casos reais de sistemas operacionais, arquiteturas de processadores, computação paralela e engenharia de produção.

No âmbito acadêmico, a aplicabilidade desta investigação é ampla. A reconstrução didática das provas pode servir como apoio em disciplinas de Estruturas de Dados, Análise de Algoritmos, Teoria da Computação, Sistemas Operacionais e Escalonamento. O estudo também pode auxiliar estudantes na compreensão de reduções polinomiais, frequentemente uma das maiores dificuldades no apren-

 ISSN: 2675-3588

dizado de NP-completude, oferecendo exemplos concretos, visuais e contextualizados. Por fim, a exposição das limitações teóricas desses modelos reforça a importância de heurísticas, algoritmos aproximados e métodos experimentais em cenários reais onde soluções exatas são inviáveis.

Assim, os resultados alcançados não apenas reafirmam a NP-completude das variantes analisadas, mas também destacam o valor formativo do tema, demonstrando como problemas clássicos podem ser reinterpretados, visualizados e aplicados em contextos educacionais e práticos. O trabalho, portanto, contribui tanto para o rigor científico quanto para o fortalecimento do ensino da complexidade computacional.

VII. Considerações Finais

Este trabalho teve como objetivo analisar, formalizar e demonstrar a NP-completude das variantes de escalonamento P2 e P3, com base nas construções apresentadas por Ullman (1975). A partir da reconstrução detalhada das reduções a partir do 3-SAT — passando pelos problemas intermediários P4 e P5, foi possível compreender, de maneira estruturada e visual, como modelos de escalonamento aparentemente simples podem capturar a complexidade combinatória de problemas booleanos clássicos. Em síntese, demonstrou-se que tanto P2 quanto P3 pertencem à classe $\mathcal{NP}$ e são NP-difíceis, concluindo-se formalmente que ambos são NP-completos.

Durante o desenvolvimento da pesquisa, algumas dificuldades se mostraram centrais. A primeira refere-se à própria interpretação das construções utilizadas nas reduções, que exigem uma leitura atenta dos padrões de precedência, capacidade e sincronização temporal criados para simular variáveis e cláusulas. A segunda diz respeito ao esforço de transformar essas construções abstratas em explicações claras, diagramas compreensíveis e justificativas coerentes, mas essencial para consolidar o entendimento. Além disso, adaptar as provas originais para uma perspectiva didática, mantendo rigor matemático, demandou uma revisão cuidadosa da literatura e um tratamento sistemático das etapas envolvidas.

Apesar dessas dificuldades, os resultados obtidos ampliam a compreensão acadêmica sobre redução polinomial e sobre o caráter intratável de problemas de escalonamento. A abordagem adotada reforça o valor pedagógico das provas de NP-completude, especialmente quando apoiadas por esquemas visuais e interpretações intuitivas. O estudo também evidencia que a complexidade computacional permanece relevante não apenas em termos teóricos, mas também como fundamento para decisões práticas em sistemas operacionais, arquiteturas de processadores e modelos de produção.

Como perspectivas futuras, sugere-se aprofundar a análise de variações modernas dos problemas de escalonamento, incluindo modelos com preempção, janelas de tempo flexíveis, pesos múltiplos e ambientes heterogêneos. Outra linha de investigação envolve a exploração de algoritmos aproximados e heurísticas, fundamentais para aplicações reais nas quais soluções exatas são inviáveis devido à NP-completude. Por fim, estudos comparativos entre provas clássicas e abordagens contemporâneas de complexidade podem contribuir para o ensino, permitindo compreender como a teoria evoluiu e como esses problemas permanecem centrais na ciência da computação.

Assim, este trabalho não apenas consolida formalmente a NP-completude de P2 e P3, mas também contribui para o fortalecimento do entendimento teórico e pedagógico sobre reduções polinomiais, oferecendo bases sólidas para investigações futuras e para a aplicação prática desses conceitos em diferentes áreas da computação.

Referências

[1] S. A. Cook, "The complexity of theorem-proving procedures," in *Proceedings of the Third Annual ACM Symposium on Theory of Computing (STOC '71)*. New York, NY, USA: Association for Computing Machinery, 1971, pp. 151–158.

[2] J. D. Ullman, "Np-complete scheduling problems," *Journal of Computer and System Sciences*, vol. 10, no. 3, pp. 384–393, 1975.

[3] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan, "Optimization and approximation in deterministic sequencing and scheduling: A survey," *Annals of Discrete Mathematics*, vol. 5, pp. 287–326, 1979.

[4] P. Brucker and S. A. Kravchenko, "Scheduling jobs with equal processing times on parallel machines," in *Discrete Optimization and Operations Research*, P. M. Pardalos and V. Zhukovskii, Eds. Berlin: Springer, 2008, pp. 38–49.

[5] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 4th ed. New York: Springer, 2016.

[6] P. Baptiste, J. Y.-T. Leung, and C. Smith, *Scheduling: Algorithms and Complexity*. Berlin: Springer, 2001.

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

[8] Y. Lassance *et al.*, "Reflexões e práticas pedagógicas no escopo da disciplina de teoria da computação," *Academic Journal on Computing, Engineering and Applied Mathematics*, vol. 6, no. 2, pp. 10–17, Oct. 2025.

ISSN: 2675-3588

Reprodução de Resultados da Literatura e Contribuições Pedagógicas: Problema de Coloração de Vértices segundo o Teorema de Brooks

Reproduction of Results from the Literature and Pedagogical Contributions: The Vertex Coloring Problem according to Brooks' Theorem

Matheus Silva Pontes[1], Lucas Monteiro de Carvalho[1], Daniel Martins da Silva[1] e Tanilson Dias dos Santos[1]

[1] *Universidade Federal do Tocantins, Ciência da Computação, Tocantins, Brasil*

Data de recebimento do manuscrito: 03/12/2025
Data de aceitação do manuscrito: 19/01/2026
Data de publicação: 10/02/2026

Resumo—Este estudo reproduz uma prova final do Teorema de Brooks, um dos resultados fundamentais para a Coloração de Grafos, visando não apenas à consolidação do conhecimento teórico, mas também para a produção de um material didático de apoio para a comunidade acadêmica, traduzindo a complexidade da prova por meio de exemplos ilustrativos, figuras e explicações detalhadas. O teorema estabelece um limite superior para o número cromático $\chi(G)$ de qualquer grafo conexo com o seu grau máximo $\Delta(G)$, tal que o grafo analisado não seja um ciclo ímpar e nem um grafo completo. A metodologia utilizada foi a prova por contradição, assumindo um contraexemplo minimal, integrada com duas técnicas cruciais, juntamente com ilustrações para facilitar o ensino. A prova é iniciada com o Lema Estrutural de Lovász, o qual é aplicado para resolver o caso dos grafos Δ-regulares e não completos. E também, a utilização da justificativa de Cadeias de Kempe permite demonstrar que a falha estrutural da coloração só é possível em casos excepcionais onde o grafo é completo ou um ciclo ímpar. O resultado é a confirmação de $\chi(G) \leq \Delta(G)$ para todo grafo conexo, exceto os casos proibidos.

Palavras-chave—Teoria dos Grafos, Coloração de Grafos, Teorema de Brooks, Número Cromático

Abstract—*This study reproduces a complete proof of Brooks' Theorem, one of the fundamental results in Graph Coloring. The aim is not only to consolidate theoretical knowledge but also to produce didactic support material for the academic community, translating the complexity of the proof through illustrative examples, figures, and detailed explanations. The theorem establishes an upper bound for the chromatic number $\chi(G)$ of any connected graph with its maximum degree $\Delta(G)$, such that the analyzed graph is neither an odd cycle nor a complete graph. The methodology employed is proof by contradiction, assuming a minimal counterexample, integrated with two crucial techniques, along with illustrations to facilitate teaching. The proof begins with Lovász's Structural Lemma, which is applied to resolve the case of Δ-regular and non-complete graphs. Furthermore, the use of Kempe Chains justification allows us to demonstrate that the structural failure of the coloring is only possible in exceptional cases where the graph is complete or an odd cycle. The result is the confirmation that $\chi(G) \leq \Delta(G)$ for every connected graph, except for the forbidden cases.*

Keywords—*Graph Theory, Graph Coloring, Brooks' Theorem, Chromatic Number*

I. Introdução

A Teoria dos Grafos possui destaque e importância pela grande variedade de problemas. O interesse principal deste campo é resolver os problemas utilizando algoritmos eficientes, preocupando-se com a capacidade computacional. A busca por soluções eficientes move esta área para que ainda seja investigada através de conhecimentos teóricos aprofundados sobre grafos [1].

A coloração de grafos se trata de um caso especial o qual atribuímos rótulos, que são as cores. Elas estão sujeitas a restrições e podem ser aplicadas em vértices e arestas, de forma que os vértices e as arestas adjacentes não possuam a mesma cor [2]. Inicialmente, esta ideia despertou no homem o desejo de buscar novas maneiras de expressar diferentes tipos de regiões. O registro de desenhos e escritas gráficas com a inserção de cores nos mapas originou a cartografia [3].

Em 1852, por meio da coloração dos mapas, se deu início à

Dados de contato: Matheus Silva Pontes, matheus.pontes@uft.edu.br

história do problema das 4 cores. O matemático, advogado e botânico Francis Guthrie formulou uma conjectura afirmando que qualquer mapa pode ser colorido utilizando apenas 4 cores. Francis apresentou este problema para seu irmão mais novo, Frederick Guthrie, que mostrou para o seu professor De Morgan. O docente, entusiasmado, encaminhou este problema em suas cartas, despertando o interesse dos acadêmicos. Esta ideia propagou-se, impulsionando discussões e novos desenvolvimentos [4]. Em 1976, Kenneth Appel e Wolfgang Haken conseguiram apresentar a demonstração do Teorema das 4 Cores com o auxílio de computadores [3].

Os conceitos e problemas desenvolvidos na coloração de grafos, como o Teorema das 4 Cores e a coloração de vértices e arestas, foram fundamentais para a resolução de problemas reais e de jogos. Entre as aplicações de destaque estão a divisão de terras [3], a organização da grade de horários, a solução de um *sudoku* utilizando um algoritmo guloso de coloração de vértices e o transporte de produtos reagentes [2].

Acerca deste caso, este artigo visa reproduzir resultados da literatura e oferecer uma contribuição pedagógica da demonstração do Teorema de Brooks. O teorema estabelece um limite superior do número cromático $\chi(G)$ em função do grau máximo $\Delta(G)$ do grafo. Dessa forma, busca-se apresentar explicações mais claras e sustentadas com exemplos e figuras ilustrativas. Assim, este material servirá como um conteúdo pedagógico de apoio para a comunidade acadêmica.

Quanto à organização deste estudo, a seção II estabelece os conceitos básicos sobre grafos e coloração, essenciais para a compreensão do Teorema de Brooks acompanhados de exemplos detalhados. A seção III expõe as fontes pedagógicas e técnicas que apresentam propostas alinhadas à deste trabalho.

Em seguida, a seção IV descreve o Teorema de Brooks detalhadamente sobre os problemas lúdicos relacionados, suas aplicações e complexidades. A seção V inicia a exposição de sua prova com a apresentação de dois lemas, Lema Estrutural de Lovász e Cadeias de Kempe. Essas técnicas são fundamentais para o desenvolvimento do argumento.

Posteriormente, a seção VI destaca as considerações relevantes sobre o teorema, adversidades encontradas durante a escrita deste artigo, soluções para contornar os desafios e destaques deste estudo para o meio acadêmico. Por fim, a seção VII realiza uma síntese dos principais aspectos, acompanhada de sugestões de melhorias dos resultados obtidos, extensões de trabalhos futuros e temas relacionados não explorados em profundidade.

É de suma importância ressaltar que os grafos utilizados para as definições e provas neste material serão finitos e simples. Nessa perspectiva, a seção seguinte apresenta as noções básicas sobre grafos e coloração necessárias para a apresentação do problema.

II. PRELIMINARES

Nesta seção, os conceitos fundamentais sobre os grafos serão introduzidos e utilizados neste trabalho, os quais foram utilizados como base o livro pedagógico do Jayme [1]. A terminologia e as notações serão apresentadas posteriormente nos problemas.

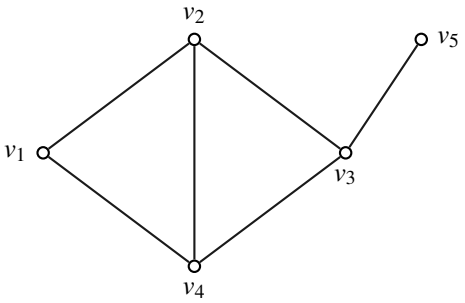


Figura 1: Representação do grafo G_1.

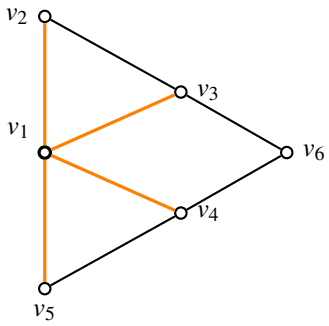


Figura 2: Ilustração da vizinhança $N(v_1)$ e do grau máximo $\Delta(G)$.

Um *grafo* $G = (V,E)$ consiste em um conjunto finito não vazio de V e um conjunto E de pares ordenados distintos de V. Os elementos de V são os *vértices* e os de E são as *arestas* de um grafo. Cada aresta $e \in E$ é denotada por um par de vértices (v,w), onde v e w são os extremos da aresta e e são ditas adjacentes. É dito que a aresta e é incidente aos vértices v e w.

Um conjunto de vértices de um grafo é denotado por $V(G)$, e um conjunto de arestas de um grafo é denotado por $E(G)$. A Figura 1 ilustra um grafo $G_1 = (V,E)$, tal que $V(G_1) = \{v_1,v_2,v_3,v_4,v_5\}$ e $E(G_1) = \{(v_1,v_2),(v_3,v_4),(v_4,v_2),(v_4,v_1),(v_3,v_5),(v_2,v_3)\}$.

A *vizinhança* de um vértice $v \in V$, denotada por $N(v)$, é definida como o conjunto de vértices adjacentes a v. O grau de v, representado por $d(v)$, corresponde à cardinalidade $|N(v)|$. O *grau máximo* de G, denotado por $\Delta(G)$, é o maior valor de grau encontrado entre todos os vértices de V. Esses conceitos podem ser visualizados na Figura 2. Como exemplo, considere a análise do vértice v_1. Sua vizinhança é dada por $N(v_1) = \{v_2,v_3,v_4,v_5\}$, de modo que seu grau é $d(v_1) = |N(v_1)| = 4$. Ao observar os demais vértices do grafo, obtemos, por exemplo, $d(v_2) = 2$, $d(v_3) = 3$, entre outros. Como $d(v_1) = 4$ é o maior grau entre todos os vértices, concluímos que o grau máximo é $\Delta(G) = 4$.

Dizemos que um grafo G é *conexo* se existir um caminho entre quaisquer dois vértices, como exemplificado no grafo à esquerda da Figura 3. Em contrapartida, um vértice v é denominado *vértice de corte* quando a sua remoção torna G desconexo. Este caso é ilustrado na Figura 3 onde o grafo da direita mostra um vértice de corte v_3.

Um *ciclo* C_n é um caminho $v_1,\ldots,v_k,v_{k+1}$ tal que $v_k = v_{k+1}$ e $k \geq 3$. Em um grafo não direcionado, todo ciclo deve possuir no mínimo 3 vértices. Se o caminho for denominado simples, o ciclo também é simples. Um ciclo simples é um ciclo onde o caminho inicia e termina no mesmo vértice [1]. Um ciclo par é aquele que possui número par de vértices e arestas, enquanto um ciclo ímpar possui número ímpar de arestas e vértices. A

ISSN: 2675-3588

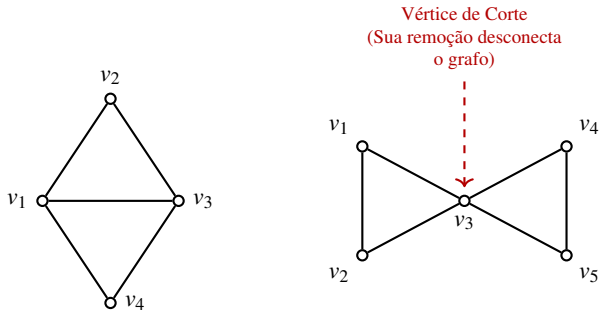
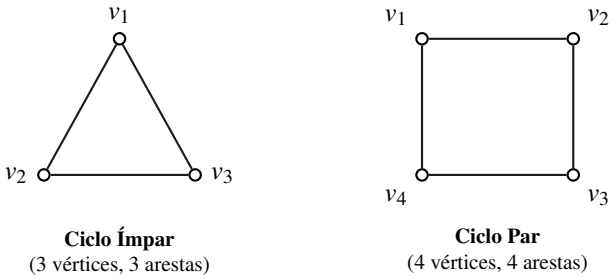


Figura 3: Esquerda: Exemplo de grafo conexo. Direita: O grafo G_3 com vértice de corte v_3.



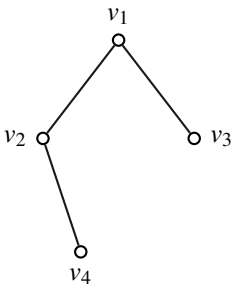


Figura 4: Exemplos de ciclos e um grafo acíclico.

Figura 4 ilustra um exemplo de ciclo par com 4 vértices e um ciclo ímpar com 3 vértices. Um grafo que não possui ciclos é chamado de grafo acíclico, o qual está exemplificado na Figura 4.

Um grafo é dito *completo* quando cada par de vértices é conectado por uma única aresta. É utilizada uma notação K_n para designar um grafo completo com n vértices. A Figura 5 mostra um grafo completo K_4, e cada vértice é ligado por uma única aresta com todos os outros vértices.

Uma *k-coloração (própria)* de G é uma função $c : V \to \{1,2,\ldots,k\}$ tal que $c(u) \neq c(v)$ para toda aresta $\{u,v\} \in E$. Observe que uma k-coloração de vértices de um grafo é a atribuição de k cores aos seus vértices de forma que quaisquer dois vértices adjacentes (conectados por uma aresta) recebam cores diferentes [1], podemos observar isso na Figura 6.

O *número cromático* de um grafo G, denotado por $\chi(G)$, é o menor inteiro k o qual G admite uma k-coloração [1]. Então, o número cromático de um grafo é o menor número de cores necessárias para colorir todos os seus vértices de forma que nenhum par de vértices adjacentes tenham a mesma cor. Este conceito é ilustrado na Figura 7, com um grafo que tem o número cromático igual a 4.

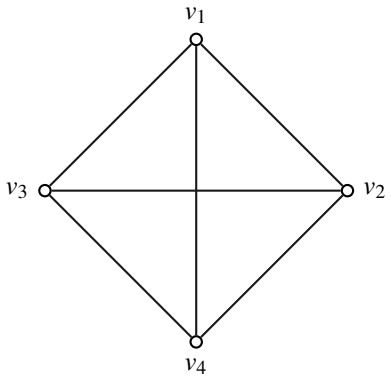


Figura 5: Exemplo de um Grafo Completo K_4.

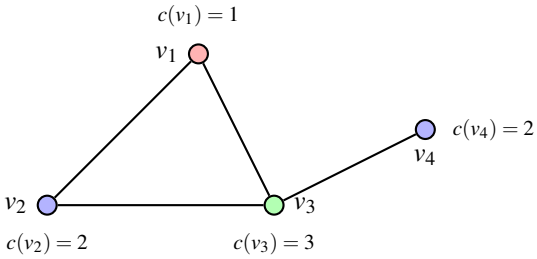


Figura 6: Uma 3-coloração própria válida para o grafo G_{col}.

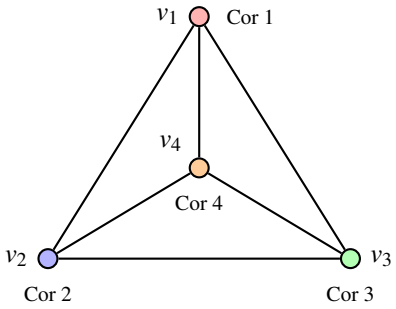


Figura 7: Grafo com número cromático igual a 4.

III. Trabalhos Relacionados

Vale destacar os trabalhos relacionados com o mesmo tema e problema deste estudo. Dentre eles, o artigo de Cranston e Rabern [5] apresenta diferentes demonstrações do Teorema de Brooks. O objetivo é ilustrar as técnicas principais da coloração de grafos, como coloração gulosa, cadeia de Kempe, lema de Kernel e *hitting sets*, com o intuito de torná-las acessíveis. Para isso, os autores desejaram mostrar as suas provas favoritas. Cada tópico das provas é apresentado em ordem de complexidade, cada um é autocontido e pode ser lido em qualquer ordem. Este trabalho será utilizado como base para a demonstração do Teorema de Brooks neste estudo.

Além disso, Sajith e Saxena [6] demonstram duas provas do Teorema de Brooks. A primeira prova é feita modificando a prova de Melkinov e Vizing [7] e de Wilson [8] que provam por contradição, porém é alterada para ser construtiva e resultar em um algoritmo de tempo linear. E a segunda prova combina com os elementos das demonstrações de Zajac [9] e do Bondy [10, 11], garantindo uma prova mais simples e resultando também em um algoritmo de tempo linear. Os

autores, Sajith e Saxena, consideram essas provas mais fáceis de serem ensinadas em aulas de Ciência da Computação.

Amiroch et al. [12] projetam um novo método para criação de cardápios alimentícios utilizando a coloração de vértices. Eles combinaram o algoritmo de Welsh-Powell [13] com uma técnica de combinação matemática a qual gera uma diversidade de cardápios que seguem diretrizes de baixa caloria. Para comprovar a eficácia da abordagem, simularam de forma dinâmica com a ferramenta MatLab para criarem três cardápios distintos com necessidades nutricionais específicas. Adicionalmente, para a organização e diversificação de cardápios nutricionalmente balanceados, empregam o conceito de caminhos disjuntos de grafos. Este estudo tem destaque pela solução de lidar com problemas em planejamento alimentar e pelo fornecimento de informações importantes para trabalhos futuros.

F. Radmehr et al. [14] focam em utilizar uma abordagem baseada em investigação para explorar o ensino e a aprendizagem sobre coloração de vértices para os alunos de graduação em matemática. Para isso, desenvolveram sete tarefas baseadas em investigação para ensinar o tema para os alunos e buscam descrever o engajamento deles. Como resultado, os discentes se entretiveram bastante com as tarefas e perceberam o quão importante essas práticas podem ser para o desenvolvimento de conhecimentos conceituais sobre matemática. Os autores promovem que essas tarefas sejam empregadas em cursos de matemática discreta de graduação para aprimorar o conhecimento matemático.

Rajagaspar e Senthil [15] buscam divulgar a ideia inicial sobre grafos e coloração de vértices. Eles investigam como a coloração de vértices pode ser usada para modelar problemas práticos, como escalonamento de horários, alocação de recursos, *networking* e mineração de dados.

Cabe mencionar o artigo de Yasser e Bianchini [16] como referência para contribuição pedagógica. Este fator é essencial para a escrita deste trabalho, dado que buscamos tornar este material acessível para a comunidade acadêmica que deseja entender sobre o Teorema de Brooks e se aprofundar na coloração de grafos.

Partindo disso, segue na próxima seção o detalhamento do problema, descrevendo o seu tipo, complexidade, problemas lúdicos relacionados, o contexto em que se enquadra, utilizando exemplos ilustrativos com explicações.

IV. Descrição do Problema

A coloração de grafos possui uma ampla variedade de aplicações práticas em diferentes áreas. Um exemplo clássico é a coloração de mapas, em que regiões adjacentes devem receber cores distintas [17]. Além desse caso bem conhecido, problemas de coloração em vértices surgem em diversas situações reais, como na alocação de frequências em redes de comunicação, onde transmissores próximos não podem operar na mesma frequência, e no escalonamento de tarefas que não podem ocorrer simultaneamente [18]. Outro uso importante aparece em compiladores [19], durante a etapa de alocação de registradores, e em sistemas de horários acadêmicos, garantindo que disciplinas que compartilham alunos não sejam ofertadas no mesmo período [13]. Além disso, é aplicado em problemas lúdicos como a resolução do *sudoku*, que pode ser modelado como um problema de coloração de grafos [20].

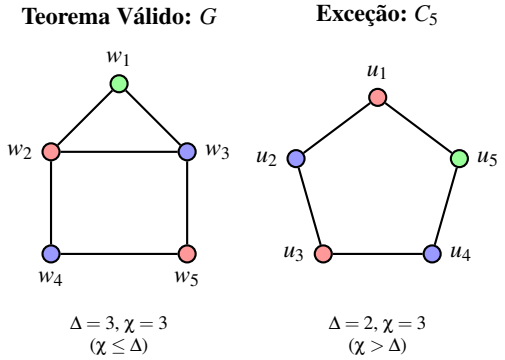


Figura 8: Visualização do Teorema de Brooks: Uma exceção (ciclo ímpar) e um caso válido.

Neste estudo, trabalharemos com coloração de vértices formulada como um problema de decisão. Em termos gerais, quando o grafo G é completo ou ciclo ímpar satisfaz $\chi(G) \leq \Delta(G)+1$. Porém, o Teorema de Brooks mostra que se um grafo G não é completo e nem ciclo ímpar, então $\chi(G) \leq \Delta(G)$ [17]. A Figura 8 ilustra o grafo G para o qual o Teorema de Brooks é válido para sua estrutura, diferente do grafo C_5, que é um ciclo ímpar.

O Teorema de Brooks insere-se no contexto mais amplo de resultados que buscam relacionar propriedades estruturais dos grafos (como conectividade, presença de ciclos, graus dos vértices) com sua coloração. Ele fornece um critério poderoso para limitar a complexidade cromática, com aplicações desde problemas de escalonamento (scheduling) até alocação de recursos e alocação de registradores.

Determinar o número cromático exato é considerado um problema *NP-completo* [21]. Existem muitas técnicas de coloração de vértices para provar o Teorema de Brooks que podem se estender em várias direções [5]. Neste estudo, buscamos utilizar duas técnicas para realizar a demonstração do teorema.

Assim, embora o Teorema de Brooks forneça um limite garantido de $\Delta(G)$ cores (exceto nos casos excepcionais), ele não fornece necessariamente um algoritmo polinomial para determinar se o grafo admite coloração com menos cores do que as previstas pelo limite. Em muitos cenários, algoritmos gulosos (greedy) podem se aproximar desse limite, mas não há garantias de optimalidade em geral.

V. Demonstração e Contribuições

Antes de iniciar a prova do teorema, apresentam-se dois lemas principais para a contextualização do problema: o **Lema de Lovász** e **Cadeias de Kempe**.

Lema 1 (Estrutura de Lovász)**.** *Seja G um grafo 2-conexo com $\delta(G) \geq 3$. Se G não for completo, então G contém um caminho induzido de três vértices, digamos u,v,w, tal que $G \setminus \{u,w\}$ é conexo.*

Proof. Para demonstrar este lema, utilizaremos a técnica de construção. Como G é conexo e não é completo, sabemos que existe algum caminho induzido de três vértices [5]. Se G for 3-conexo, a remoção de quaisquer dois vértices (u,w) não desconecta o grafo, então qualquer caminho induzido serve.

ISSN: 2675-3588

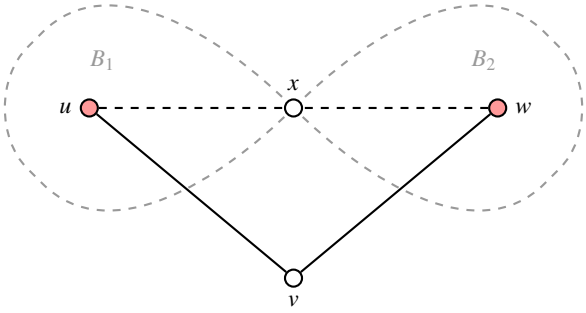


Figura 9: Visualização da estrutura: Os contornos tracejados indicam os blocos B_1 e B_2 que se encontram no vértice de articulação x.

O caso crítico ocorre quando G não é 3-conexo. Neste caso, existe um conjunto de corte de tamanho 2. Seja $\{v,x\}$ este conjunto de corte, onde v será o vértice central do nosso caminho desejado.

Considere o grafo $H = G - v$. Como $\{v,x\}$ é um corte em G, então x deve ser um vértice de corte em H (ou H é desconexo, mas como G é 2-conexo, H deve ser conexo e x é quem articula os componentes). O grafo H pode ser decomposto em seus blocos (subgrafos maximais 2-conexos). A estrutura desses blocos forma uma árvore (o "grafo de blocos"). Uma árvore com pelo menos uma aresta possui pelo menos duas folhas (blocos finais). Sejam B_1 e B_2 dois blocos finais (endblocks) de H. Pela propriedade de 2-conexidade de G: Cada bloco final de H deve conter pelo menos um vértice adjacente a v que não seja x. Se não houvesse tal vizinho, a remoção apenas de x em G desconectaria aquele bloco do resto do grafo, o que contradiz o fato de G ser 2-conexo (que exige remoção de 2 vértices para desconectar). Sejam $u \in V(B_1)$ e $w \in V(B_2)$ vizinhos de v (com $u,w \neq x$). O caminho $u - v - w$ é induzido (pois u e w estão em blocos diferentes separados por x, logo não há aresta direta entre eles, a menos que passem por x, mas estamos olhando vizinhança direta) podemos ver isso na figura 9. Agora verificamos a conectividade de $G' = G \setminus \{u,w\}$.

- O grafo $H = G - v$ é conexo.
- u e w não são vértices de corte em H (pois pertencem a blocos finais e não são a articulação x). Logo, $H \setminus \{u,w\}$ permanece conexo.
- Ao readicionarmos v (para formar G'), precisamos garantir que v se conecte a $H \setminus \{u,w\}$.
- Como o grau $\delta(G) \geq 3$, o vértice v tem grau pelo menos 3. Dois vizinhos são u e w. Logo, v tem pelo menos mais um vizinho (podendo ser x ou outro vértice em H). Isso garante que v não fica isolado, como visto na Figura 10.

Portanto, $G \setminus \{u,w\}$ é conexo. □

O procedimento construtivo descrito na demonstração acima é formalizado no Algoritmo [1].

Lema 2 (Corretude do Algoritmo 1)**.** *Seja G um grafo 2-conexo, não completo, com $\delta(G) \geq 3$. O Algoritmo 1 retorna, em tempo finito, uma tripla de vértices (u,v,w) tal que o caminho $u - v - w$ é induzido e o grafo $G' = G \setminus \{u,w\}$ é conexo.*

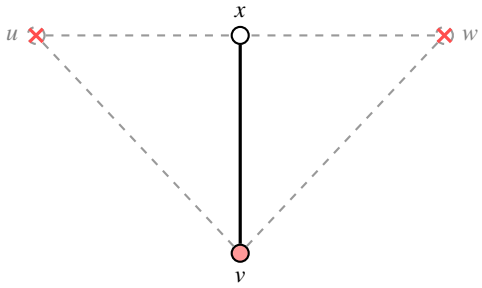


Figura 10: Conectividade de $G' = G \setminus \{u,w\}$.

Algorithm 1 Busca de Caminho Induzido com Extremidades Removíveis(Método de Lovász)

Require: Grafo 2-conexo G com $\delta(G) \geq 3$, não completo.
Ensure: Caminho induzido $u - v - w$ tal que $G \setminus \{u,w\}$ é conexo.

▷ *Caso base trivial*
1: **if** G é 3-conexo **then**
2: Encontrar qualquer caminho induzido $u - v - w$.
3: **return** (u,v,w)
4: **end if**
▷ *Passo 1: Identificar o corte e definir o centro v*
5: Encontrar um par de corte $\{v,x\}$ em G.
▷ *Nota: v será o centro do nosso caminho.*
▷ *Passo 2: Decomposição em blocos*
6: Construir $H = G - v$ e obter sua árvore de blocos-corte.
7: Sejam B_1 e B_2 dois blocos finais de H (separados por x).
▷ *Passo 3: Selecionar as pontas u e w*
8: Escolher $u \in V(B_1) \setminus \{x\}$ tal que u seja vizinho de v.
9: Escolher $w \in V(B_2) \setminus \{x\}$ tal que w seja vizinho de v.
▷ *Verificação Implícita: Como u,w estão em blocos separados por x,*
▷ *não há aresta direta u − w, logo o caminho é induzido.*
10: **return** (u,v,w)

Proof. A terminação do algoritmo é garantida, pois todas as operações (busca de componentes, identificação de blocos e cortes) são executadas em grafos finitos com complexidade polinomial. Resta demonstrar a corretude da saída em dois casos.

Caso 1: G **é** 3**-conexo (Linhas 1-4).** Pela definição de k-conectividade, a remoção de menos de k vértices não desconecta o grafo. Como $k = 3$, a remoção do conjunto $\{u,w\}$ (tamanho 2) resulta em um grafo G' conexo. Como G não é completo, existe pelo menos um caminho induzido de comprimento 2. Logo, a saída é válida.

Caso 2: G **não é** 3**-conexo (Linhas 5-13).** Neste caso, o algoritmo identifica um par de corte $\{v,x\}$. Definimos $H = G - v$. Como G é 2-conexo, H é conexo e x é um vértice de corte em H (separando os blocos finais B_1 e B_2).

- **Existência dos vértices** u **e** w**:** Pela 2-conectividade de G, cada bloco final B_i de H deve possuir pelo menos um vértice adjacente a v que não seja x. Caso contrário, $\{x\}$ seria um corte em G, contradizendo a hipótese inicial. Logo, a escolha de $u \in V(B_1) \setminus \{x\}$ e $w \in V(B_2) \setminus \{x\}$ nas linhas 9-10 é sempre possível.
- **Caminho Induzido:** Os vértices u e w pertencem a blocos distintos de H, articulados apenas por x. Como

$u \neq x$ e $w \neq x$, qualquer caminho entre u e w em H deve passar por x. Logo, não existe aresta direta (u,w) em G. Assim, o caminho $u - v - w$ é induzido.

- **Conectividade de $G' = G \setminus \{u,w\}$:** A conectividade é preservada em duas etapas:

  1. *Conectividade de H:* Em uma decomposição em blocos, os vértices que não são de articulação (como u e w dentro de blocos finais) não desconectam o grafo ao serem removidos. Portanto, $H \setminus \{u,w\}$ permanece conexo.
  2. *Reconexão de v:* O vértice v é readicionado para formar G'. Como $\delta(G) \geq 3$, $d(v) \geq 3$. O algoritmo remove dois vizinhos (u e w). Logo, resta pelo menos um vizinho de v em H (seja x ou outro vértice). Isso garante que v se conecta ao componente conexo restante $H \setminus \{u,w\}$.

Portanto, $G \setminus \{u,w\}$ é conexo e o algoritmo está correto. □

Lema 3 (Cadeia de Kempe)**.** *Em uma coloração própria de um grafo G, uma cadeia de Kempe (i,j) é uma componente conexa do subgrafo induzido pelos vértices coloridos com as cores i e j [5]. Se trocarmos as cores de i e j simultaneamente em todos os vértices dessa componente, obtemos novamente uma coloração própria de G.*

A demonstração do **Teorema de Brooks** segue a abordagem de contradição assumindo um **contraexemplo minimal** [5], combinada com duas técnicas fundamentais: a Estrutura de Lovász e as Cadeias de Kempe. Ademais, estão incluídas explicações intermediárias e observações pedagógicas para facilitar o entendimento da estrutura lógica da prova. Segue o teorema e a demonstração abaixo:

Teorema 1. *Seja G um grafo conexo. Se G não é um ciclo ímpar e nem um grafo completo, então $\chi(G) \leq \Delta(G)$.*

Proof. Suponha, por contradição, que G é um contraexemplo minimal ao teorema, ou seja, G é um grafo conexo Δ-regular com o menor número de vértices tal que $\chi(G) > \Delta$ e que não é um ciclo ímpar nem um grafo completo. Como G é minimal, todo subgrafo próprio $H \subseteq G$ satisfaz $\chi(H) \leq \Delta$.

Escolha um vértice arbitrário $v \in V(G)$. Então $G - v$ é Δ-colorível. Pelo **Lema 1**, $G - v$ possui pelo menos dois blocos terminais. Sejam u e w vértices não-cortantes pertencentes a blocos terminais distintos. Além disso, considerando novamente o **Lema 1** existe um caminho induzido $u - v - w$ tal que $G - \{u,w\}$ permanece conexo. A Figura 11 ilustra a estrutura inicial do grafo para a demonstração.

Colore u e w com a mesma cor, pois são vértices não adjacentes, e então colorimos $G - \{u,w\}$ gulosamente. A ordem utilizada segue a Estrutura de Lovász: começamos a coloração a partir dos blocos terminais de $G - v$, movendo-os em direção ao vértice v. Dessa forma, cada vértice (exceto v) é colorido quando todos os seus vizinhos posteriores na ordem já foram coloridos, o que garante a coloração. A Figura 12 mostra a coloração de u e w e colorindo gulosamente.

Após a coloração de todos os vértices de $G - \{u,w\}$, se alguma cor fica disponível para v, obtemos uma coloração usando Δ cores, encerrando a prova. Caso contrário, cada

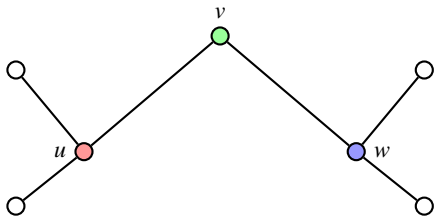


Figura 11: Estrutura inicial: vértices u e w em blocos terminais distintos de $G - v$.

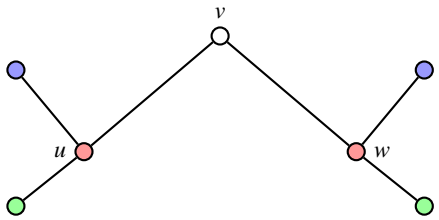


Figura 12: Coloração imediata após a coloração gulosa: todos os vértices, exceto v, já foram coloridos e u e w compartilham a mesma cor.

uma das Δ cores aparece em $N(v)$. Para liberar uma cor, empregamos cadeias de Kempe, explícito no **Lema 3**.

Para cada cor $i \in \{1,\ldots,\Delta\}$, considere v_i o vizinho de v usando a cor i. Por um argumento semelhante, para cada v_i, cada cor diferente de i aparece em um vizinho de v_i; se não, poderíamos recolorir v_i e colorir v com i. Para cada par de cores i e j, seja $C_{i,j}$ a cadeia de Kempe (i,j) contendo v_i.

A partir desta construção, formulamos as seguintes afirmações que descrevem configurações impossíveis para um contraexemplo minimal:

- **Afirmação 1:** Para qualquer par de cores i e j, a cadeia de Kempe que começa em um vértice da cor i e a cadeia que começa em um vértice da cor j têm que ser a mesma componente conexa. Porque, se fossem cadeias diferentes, poderíamos fazer uma troca na cadeia de Kempe onde está o vértice da cor i. Esta troca inverteria as cores nesse componente, e com isso faria a cor i desaparecer da vizinhança de v. A Figura 13 ilustra a situação antes da troca, com componentes de Kempe (i,j) disjuntos contendo v_i (esquerda) e v_j (direita), e a Figura 14 ilustra o momento em que troca é realizada no componente de v_i. Agora, v_i e v_j usam a cor j (azul), e a cor i (vermelho) está livre em $N(v)$. O vértice v pode ser colorido com i, contradizendo o contraexemplo minimal.
- **Afirmação 2:** Qualquer cadeia de Kempe precisa ser um caminho simples, isto é, ela não pode ter ramificações, não pode ter vértices com grau maior que 2 dentro da cadeia. Se

ISSN: 2675-3588

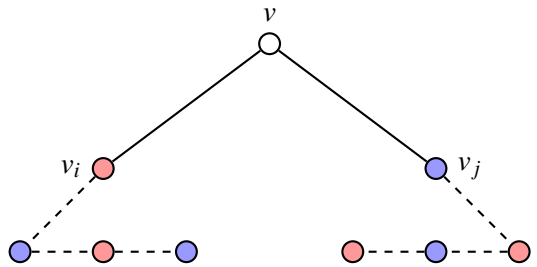


Figura 13: Afirmação 1: Antes da Troca (Violação $C_{i,j} \neq C_{j,i}$).

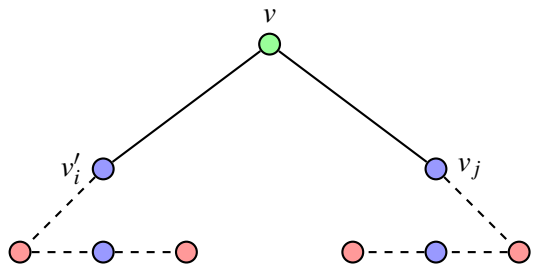


Figura 14: Afirmação 1: Depois da Troca.

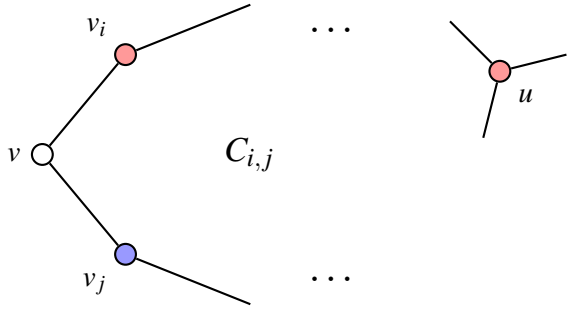


Figura 15: Ilustração da Afirmação 2. Fonte: Adaptado de Cranston e Rabern [5]

uma cadeia tivesse um vértice com grau 3, poderia recolorir apenas uma parte da cadeia de modo a liberar uma cor para um vértice vizinho de v, e de novo conseguiríamos colorir v, destruindo o contraexemplo. A Figura 14 consiste em uma adaptação do trabalho de Cranston e Rabern [5] e ilustra a interpretação da afirmação em questão.

- **Afirmação 3:** Duas cadeias que partem do mesmo vértice v_i e usam cores diferentes só podem se encontrar no próprio v_i e em nenhum outro lugar. Se houvesse qualquer outro vértice u que estivesse ao mesmo tempo em $C_{i,j}$ e em $C_{i,k}$, então u teria vizinhos em cores j e k dentro das cadeias, o que novamente permitiria recolorir parte das cadeias e liberar a cor i na vizinhança de v. Com isso, G deixaria de ser um contraexemplo. A Figura 16 também é uma adaptação do trabalho de Cranston e Rabern [5] e esclarece a afirmação discutida.
- **Afirmação 4:** Agora juntamos todas as três propriedades e mostramos que elas não podem valer ao mesmo tempo. Escolhemos três vizinhos de v, chamados v_1, v_2 e v_3, cada um com uma cor diferente. Pela estrutura do grafo, existe um vértice u na cadeia $C_{1,2}$. E como v_1 e v_3 usam as cores 1 e 3, também existe uma cadeia $C_{1,3}$ conectando esses dois vértices. Realizamos uma troca de Kempe na cadeia $C_{1,3}$: os vértices de cor 1 viram 3, e os de cor 3 viram 1. Depois da troca, o vértice u, que antes só estava na cadeia $C_{1,2}$, passa a estar ao mesmo tempo na cadeia $C'_{1,2}$ e na cadeia $C'_{2,3}$. Mas isso contradiz a Afirmação 3, que dizia que esse

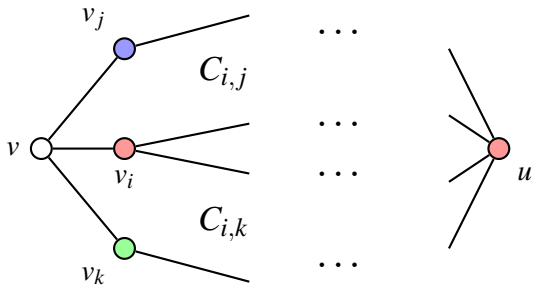


Figura 16: Ilustração da Afirmação 3. Fonte: Adaptado de Cranston e Rabern [5]

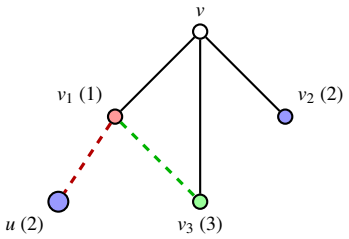


Figura 17: Afirmação 4: ANTES da troca em $C_{1,3}$. u pertence à cadeia $C_{1,2}$ e v_1 está conectado a v_3 na cadeia $C_{1,3}$.

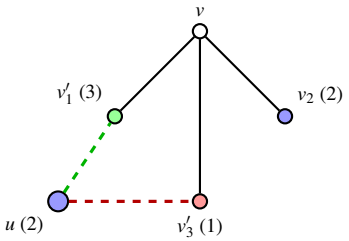


Figura 18: Afirmação 4: DEPOIS da Troca em $C_{1,3}$. $u \in C'_{1,2} \cap C'_{2,3}$.

tipo de interseção só pode ocorrer no próprio v_1. Como encontramos essa interseção, as três afirmações não podem ser todas verdade. Logo, o contraexemplo minimal não pode ser válido e o Teorema de Brooks é válido. Observe a Figura 17, ela mostra o momento antes da troca em $C_{1,3}$, v_1, v_2, v_3 são vizinhos de v, u (cor 2) é um vizinho de v_1 (cor 1) e faz parte da cadeia $C_{1,2}$. A Figura 18 ilustra o momento onde a troca é realizada, v'_1 agora tem cor 3, u ainda tem cor 2 e v_2 tem cor 2. O vértice **u** agora pertence a $C'_{1,2}$ e $C'_{2,3}$, violando a **Afirmação 3** ($C_{i,j} \cap C_{i,k} = v_i$).

As Afirmações 1, 2 e 3 descrevem propriedades que qualquer contraexemplo minimal precisaria ter. Mas a Afirmação 4 mostra que essas três condições juntas levam a uma contradição depois de uma troca de Kempe.

Com isso, existe uma troca válida que libera uma cor em $N(v)$. Após a troca, colorimos v com a cor liberada e, temos um grafo Δ-colorível. Isto contradiz o contraexemplo minimal, logo, não existe um contraexemplo para este teorema e, portanto:

$$\chi(G) \leq \Delta(G)$$

□

O contraexemplo precisa ser Δ-regular, pois a coloração gulosa produziria uma Δ-coloração de G, contradizendo $\chi(G) > \Delta$. As técnicas de Lovász e da cadeia de Kempe

garantem um vértice especial que pode ser controlado no entorno, uma vez que é possível colorir partindo de blocos terminais, e sempre existe uma troca de cores possível para resolver conflitos locais. Ademais, na Afirmação 4 se afirma que tem vizinhos v_1 e v_2 não adjacentes, porque os vizinhos de v não formam clique. Se fosse um clique, o grafo seria completo, porém é uma exceção para o teorema. Logo, os vizinhos de v não podem formar um clique.

Na seguinte seção, serão relatadas as contribuições adicionais dos autores, dificuldades encontradas, solução para as adversidades e como este problema possui potenciais pedagógicos para o meio acadêmico.

VI. Resultados e Reflexões

O trabalho de Cranston e Rabern [5] destaca-se pela abordagem clara e ilustrativa para apresentar o teorema. Este foi utilizado como base para a demonstração desenvolvida neste estudo. Os autores evidenciam as buscas de suas provas favoritas do Teorema de Brooks, expondo as técnicas principais da coloração de vértices usadas para auxiliar nas demonstrações. Cabe ressaltar que possuíam o objetivo de apresentar as provas, destacando suas vantagens e extensões de cada uma. A organização deste trabalho permite a leitura independente de suas seções, uma vez que cada seção apresenta, de forma autossuficiente, as técnicas e a demonstração. Ademais, o artigo explicita outros autores que adotaram estratégias diferentes para as provas baseado na demonstração apresentada em cada seção.

A ilustração dos conceitos abstratos e as pesquisas apresentadas no trabalho de Cranston e Rabern [5] contribuem para o meio acadêmico. Essas contribuições possibilitam a construção de novos temas para trabalhos futuros, aprimorando os argumentos e definindo novas abordagens. Embora os autores não descrevam as suas adversidades encontradas durante o processo da elaboração do artigo, eles salientam a relevância das contribuições recebidas por meio de comentários e sugestões de outros pesquisadores. Os feedbacks e as avaliações foram cruciais para o refinamento da escrita e para a atualização do problema apresentado.

As maiores dificuldades vistas na escrita deste trabalho estiveram relacionadas à escolha dos métodos para realizar a demonstração. Algumas abordagens requerem um conhecimento mais apurado em Teoria dos Grafos. A intenção é expor ao menos dois lemas de maneira coerente e didática. Este problema expandiu-se na prova do problema e dos lemas auxiliares, buscando ajudar no desenvolvimento da resolução do Teorema de Brooks. Além disso, houve dificuldade na construção de ilustrações que representassem de forma clara cada conceito apresentado sobre grafos e das técnicas aplicadas.

Para contornar essas adversidades, foram feitas pesquisas aprofundadas em artigos acadêmicos que mostravam a prova completa do Teorema de Brooks. Estes trabalhos utilizam diferentes lemas com suas técnicas de demonstração adequadas para uma conclusão correta. Para a elaboração das figuras ilustrativas, demandou mais análise sobre materiais didáticos que abordam noções básicas de grafos acompanhada de exemplos pedagógicos e explicativos. A partir disso, foram selecionados os lemas adequados para este trabalho. Dessa maneira, este material busca servir como um suporte didático para os acadêmicos que desejam se aprofundar no assunto de coloração dos grafos.

A coloração de grafos tem grande potencial para aplicações pedagógicas e acadêmica. Essa área é utilizada para modelagem de problemas reais, construção de ferramentas computacionais e desenvolvimento de algoritmos de coloração [22]. Ademais, a demonstração apresentada pode servir como base para a construção de teses e dissertações que explorem classificações de grafos e diferentes técnicas de coloração.

VII. Considerações Finais

Portanto, este estudo contextualiza o cenário sobre coloração de grafos, destaca os conceitos fundamentais e básicos sobre grafos para esclarecer o problema e a demonstração do Teorema de Brooks. O teorema estabelece que, se um grafo G tal que G não seja completo nem ciclo ímpar, então satisfaz $\chi(G) \leq \Delta(G)$. Para demonstrar o problema, foi utilizada a prova por contradição assumindo um contraexemplo minimal. A abordagem foi apoiada por duas técnicas essenciais: o Lema Estrutural de Lovász e cadeia de Kempe.

Adicionalmente, foram destacados outros trabalhos relacionados ao tema, os quais mostram diferentes formas de demonstração e técnicas elaboradas ou modificadas pelos autores. As contribuições evidenciam que o estudo de coloração de vértices permanece relevante para a resolução de problemas reais. Para a construção deste trabalho, uma pesquisa aprofundada de artigos, que exploram o mesmo problema, foi necessária com o intuito de reproduzir os resultados da literatura, buscando tornar o ensino mais acessível e descomplicado.

A construção da demonstração, a síntese das técnicas e a elaboração das ilustrações foram os desafios encontrados ao longo deste estudo. Isso se deve ao fato de que se priorizou a apresentação de uma explicação compreensível e coesa para a comunidade acadêmica.

A coloração de grafos apresenta potencial para o desenvolvimento de trabalhos futuros, principalmente no que se refere à aplicação em problemas reais e à estruturação de algoritmos. Como perspectiva de melhoria dos resultados apresentados, destaca-se a possível inclusão da prova baseada em Coloração de Listas (List Coloring) ou do lema de Kernel [5], que são técnicas mais modernas da literatura. No entanto, essas abordagens não foram implementadas no presente artigo devido à sua maior complexidade conceitual. Sua inclusão exigiria um detalhamento adicional de preliminares e poderia comprometer a acessibilidade do material para o escopo desta disciplina.

Referências

[1] J. L. Szwarcfiter, *Teoria Computacional de Grafos: os algoritmos*. Rio de Janeiro: Elsevier, 2018.

[2] R. P. Alves, "Coloração de grafos e aplicações," Ph.D. dissertation, Universidade Federal de Santa Catarina (UFSC), Florianópolis, SC, 2015, dissertação de Mestrado Profissional em Matemática.

[3] D. S. Vasconcelos, "Coloração em grafos e aplicações," Ph.D. dissertation, Universidade Federal de Sergipe (UFS), São Cristóvão, SE, 2018, Dissertação de Mestrado em Matemática PROFMAT.

[4] L. Sousa, "O Teorema das Quatro Cores," *Millenium*, no. 25, pp. 125–151, 2003.

 ISSN: 2675-3588

[5] D. W. Cranston and L. Rabern, "Brooks' Theorem and Beyond," *Journal of Graph Theory*, vol. 80, no. 3, pp. 199–225, 2014.

[6] G. Sajith and S. Saxena, "On Brooks' Theorem," *arXiv preprint*, 2025, Available at arXiv:2208.02186v2.

[7] L. S. Mel'nikov and V. G. Vizing, "New proof of Brooks' theorem," *Journal of Combinatorial Theory*, vol. 7, pp. 289–290, 1969.

[8] R. J. Wilson, *Introduction to Graph Theory*, 5th ed. London: Pearson Education, 2010.

[9] M. Zajac, "A short proof of Brooks' theorem," *arXiv preprint*, 2018, Available at arXiv:1805.11176.

[10] J. A. Bondy, "Short proofs of classical theorems," *Journal of Graph Theory*, vol. 44, no. 3, pp. 159–165, 2003.

[11] J. A. Bondy and U. S. R. Murty, *Graph Theory*, ser. Graduate Texts in Mathematics. Berlin: Springer, 2008, vol. 244.

[12] S. Amiroch, H. Chang, M. Jamhuri, and T. Yulianto, "Vertex coloring in graphs: A novel approach to nutritional menu planning," in *AIP Conference Proceedings*, vol. 3176, no. 1. AIP Publishing, 2024, p. 020006.

[13] D. J. A. Welsh and M. B. Powell, "An upper bound for the chromatic number of a graph and its application to timetabling problems," *The Computer Journal*, vol. 10, no. 1, pp. 85–86, 1967.

[14] F. Radmehr and co authors, "Teaching and learning of vertex coloring using an inquiry-based approach," *Journal of Graph Theory Education*, 2023.

[15] S. S. M. Rajagaspar, "Applications of Graph Coloring Using Vertex Coloring," *PublishOA Journal*, 2023, Acesso via PublishOA. DOI/ISSN não especificados.

[16] Y. O. Lassance, R. Bianchini, and R. R. Santos, "Uma experiência didática em Teoria da Computação: limites da computação e Ensino Médio," *Revista de Ensino de Ciências e Matemática (REnCiMa)*, vol. 16, no. 2, pp. 1–20, 2025.

[17] R. Diestel, *Graph Theory*, 4th ed., ser. Graduate Texts in Mathematics. Heidelberg: Springer, 2010, vol. 173.

[18] W. K. Hale, "Frequency assignment: Theory and applications," *Proceedings of the IEEE*, vol. 68, no. 12, pp. 1497–1514, 1980.

[19] G. J. Chaitin, "Register allocation & spilling via graph coloring," *SIGPLAN Notices*, vol. 17, no. 6, pp. 98–105, 1982.

[20] L. E. C. Laurindo and F. J. da Silva e Silva, "SaturBFS: Um Algoritmo de Coloração de Grafos para Resolução do Sudoku," in *Anais da Escola Regional de Computação do Ceará, Maranhão e Piauí (ERCEMAPI), 9º.* Sociedade Brasileira de Computação (SBC), 2021, pp. 74–81.

[21] R. M. Karp, *Reducibility among combinatorial problems*. New York: Plenum Press, 1972, pp. 85–103.

[22] C. B. Silva, M. E. Sena, C. M. S. Machado, and D. F. Adamatti, "Uma abordagem teórica e prática da coloração em problemas modelados por grafos," *Vetor*, vol. 26, no. 2, pp. 61–72, 2016.

ISSN: 2675-3588

Hitting Set: Contribuições Pedagógicas para o Aprendizado no Escopo da Teoria da Computação

Hitting Set: Pedagogical Contributions for Learning within the Scope of Theory of Computation

João Pedro Melo Póvoa[1], Benedito Jaime Melo Moraes Junior[1], Daniel Martins da Silva[2] e Tanilson Dias dos Santos[1]

[1] *Universidade Federal do Tocantins (UFT), Ciência da Computação, Palmas, Brasil*
[2] *Universidade Federal do Norte do Tocantins (UFNT), Araguaína, Brasil*

Data de recebimento do manuscrito: 03/12/2025
Data de aceitação do manuscrito: 22/01/2026
Data de publicação: 10/02/2026

Resumo—Este artigo apresenta uma abordagem pedagógica para o estudo do problema *Hitting Set*, com o intuito de reproduzir e tornar acessível a demonstração clássica de sua NP-completude, conforme estabelecida na literatura especializada. Diferentemente de trabalhos que visam propor novos resultados teóricos inéditos, o objetivo central desta pesquisa é preencher uma lacuna didática, detalhando minuciosamente a redução polinomial a partir do problema *Vertex Cover* (Problema Alvo) para o *Hitting Set* (Problema Atacado). A metodologia adotada inicia-se com uma revisão dos conceitos fundamentais, incluindo as definições formais das classes P e NP, bem como o conceito de certificado e verificação eficiente. Em seguida, uma prova inspirada na de Richard Karp é construída passo a passo, com ênfase na visualização da transformação das instâncias de grafos para coleções de conjuntos através de diagramas de "antes e depois". Adicionalmente, introduz-se o "Dilema dos Observadores", uma analogia original para ilustrar a complexidade combinatória. Por fim, discutem-se aplicações práticas em bioinformática e engenharia de software, consolidando o material como um recurso de apoio eficaz ao ensino de Teoria da Computação.

Palavras-chave—Hitting Set, NP-Completo, Vertex Cover, Redução Polinomial, Teoria da Computação.

Abstract—*This paper presents a pedagogical approach to the study of the Hitting Set problem, aiming to reproduce and make accessible the classic demonstration of its NP-completeness, as established in the specialized literature. Unlike works aiming to propose novel theoretical results, the central objective of this research is to bridge a didactic gap by meticulously detailing the polynomial reduction from the Vertex Cover problem to the Hitting Set problem. The adopted methodology begins with a review of fundamental concepts, including formal definitions of the P and NP classes, as well as the concepts of certificates and efficient verification. Subsequently, a proof inspired by Richard Karp is constructed step-by-step, emphasizing the visualization of transforming graph instances into set collections using "before and after" diagrams. Additionally, the "Observer's Dilemma" is introduced—an original analogy to illustrate combinatorial complexity. Finally, practical applications in bioinformatics and software engineering are discussed, consolidating the material as an effective support resource for teaching Theory of Computation.*

Keywords—*Hitting Set, NP-Complete, Vertex Cover, Polynomial Reduction, Theory of Computation.*

I. Introdução

O ensino de Teoria da Computação, especificamente no tópico de NP-Completude, impõe desafios significativos aos estudantes de graduação devido ao alto nível de abstração exigido. Compreender formalmente como a dificuldade de um problema pode ser "traduzida" para o conceito de redução polinomial é frequentemente uma barreira de aprendizado que exige mais do que apenas definições matemáticas, exige visualização e intuição. Entre os diversos problemas estudados no âmbito da classe $\mathcal{NP}$, o *Hitting Set* ocupa papel significativo, tanto por sua relevância teórica quanto por sua ampla gama de aplicações práticas conforme discutido por Karp [1] ao apresentar sua formulação clássica no estudo dos problemas NP-Completos. Tendo em vista a união de referências clássicas amplamente adotadas, é perceptível que esses materiais frequentemente

Dados de contato: Benedito Jaime Melo Moraes Junior, benedito.jaime@uft.edu.br

apresentam a prova de NP-completude do Hitting Set de forma condensada, com poucos recursos visuais e saltos lógicos que pressupõem um alto grau de maturidade matemática do leitor. Na prática de sala de aula, observa-se que estudantes de graduação têm dificuldade em acompanhar esses argumentos sem um material intermediário que detalhe a redução passo a passo, com exemplos graduais e analogias concretas. Assim, identifica-se uma lacuna didática entre a literatura de referência, voltada a um público mais avançado, e as necessidades de estudantes em disciplinas introdutórias de Teoria da Computação.

Neste contexto, este trabalho visa oferecer uma reprodução pedagógica da prova de NP-Completude do problema *Hitting Set*. Utilizando a redução clássica a partir do *Vertex Cover* (Cobertura de Vértices) [2], buscamos detalhar as etapas lógicas e fornecer recursos visuais que auxiliem o entendimento da literatura técnica padrão, facilitando a assimilação dos conceitos fundamentais por estudantes iniciantes. Dessa forma, como uma contribuição pedagógica, este trabalho apresenta recursos para facilitar o aprendizado dos conceitos de Teoria da Computação. O material inclui uma prova da NP-Completude do problema Hitting Set, desdobrando os aspectos técnicos para maior clareza. Para tornar os conceitos abstratos mais tangíveis, são fornecidas figuras e exemplos que ilustram tanto o processo de redução polinomial quanto a verificação das soluções. Como parte de uma estratégia lúdica, o estudo incorpora um problema ilustrativo (o "Dilema dos Observadores"), que aproxima o conceito de complexidade combinatória do cotidiano dentro da complexidade explorada na NP-Completude. Por fim, a compreensão da classe $\mathcal{NP}$ é reforçada com a inclusão de pseudocódigo e análise de verificação, demonstrando formalmente a eficiência do algoritmo que checa a validade de uma solução candidata.

A estrutura deste trabalho foi organizada para guiar o leitor desde os fundamentos até a prova formal. Iniciamos revisando os conceitos basilares de grafos e complexidade relacionados ao problema estudado. Em seguida, contextualizamos o problema na literatura, para então definirmos o *Hitting Set* e apresentarmos a demonstração técnica visual, encerrando com uma reflexão sobre as estratégias de aprendizado adotadas, seguindo a metodologia de seminários proposta por Lassance, Bianchini e Santos [3]. O objetivo central deste trabalho, portanto, não é apresentar novos resultados teóricos sobre Hitting Set, mas organizar uma rota de aprendizagem que torne a prova clássica de sua NP-completude acessível a estudantes iniciantes, complementando os livros-texto tradicionais, com ênfase em recursos de visualização que mostrem a transformação das instâncias, demonstrações que gradualmente aproximem o estudante da prova completa e conexões explícitas entre a prova abstrata e aplicações concretas.

II. PRELIMINARES

Para fundamentar a demonstração que será desenvolvida, revisamos nesta seção os conceitos essenciais e estabelecemos a notação utilizada. As definições aqui apresentadas seguem as convenções de Sipser [4] e Garey & Johnson [2].

O primeiro conceito fundamental é o de *grafo*, uma estrutura matemática amplamente utilizada para modelar

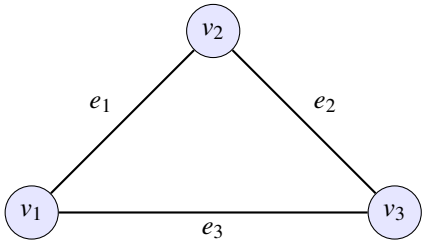


Figura 1: Representação de um grafo não direcionado $G = (V,E)$. As arestas são rotuladas como e_i, e os vértices como v_i.

relações entre objetos, como ligações entre computadores em uma rede, estradas ligando cidades ou conexões entre páginas da web. Formalmente, um grafo é denotado por $G = (V,E)$, onde V representa o conjunto de vértices (ou nós), que são os pontos do grafo, e E representa o conjunto de arestas, que são as conexões entre pares de vértices. Em um grafo simples e não direcionado, cada aresta é um par não ordenado $\{u,v\}$, indicando apenas que existe uma ligação entre u e v, sem sentido de direção. Esse tipo de estrutura é especialmente conveniente para problemas de cobertura, pois permite enxergar relações de conexão de maneira clara.

Para a redução proposta que será apresentada mais adiante, é crucial entender também o conceito de *incidência* e de *grau*. Dizemos que uma aresta $\{u,v\} \in E$ é incidente aos vértices u e v, isto é, ela "toca" exatamente esses dois vértices. O grau de um vértice, por sua vez, é o número de arestas incidentes a ele e indica quantas conexões diretas aquele ponto possui dentro do grafo. A Figura 1 apresenta uma ilustração visual desses componentes: os círculos representam os vértices (V) e as linhas representam as arestas (E). No exemplo, o vértice v_3 possui grau 2, pois está ligado a v_1 e v_2; esse tipo de contagem será reutilizado mais adiante quando mapearemos vértices e arestas para conjuntos e elementos na redução para o problema *Hitting Set*.

Além do conceito de grafos utilizado, é necessário abordar que o contexto deste trabalho exige a definição clara do ambiente de Complexidade Computacional de forma que facilite a compreensão dentro do âmbito conteúdo-aluno.

Finalmente, para realizar a prova de NP-Completude, utilizaremos o conceito de redução e um problema base. O problema escolhido como ponto de partida é o *Vertex Cover*. Sua NP-Completude foi demonstrada por Richard Karp [1], sendo uma das mais aceitas no contexto de cobertura de grafos. Ele é definido pela seguinte instância e questão: dado um grafo $G = (V,E)$ e um inteiro k, é possível escolher um subconjunto de vértices $C \subseteq V$ (com $|C| \leq k$) tal que todas as arestas de E tenham pelo menos uma extremidade em C?

Problema: VERTEX COVER (VC)

Entrada: Um grafo simples $G = (V,E)$ e um inteiro $k \in \mathbb{N}$.

Questão: Existe um subconjunto $C \subseteq V$ com $|C| \leq k$ tal que, para toda aresta $\{u,v\} \in E$, vale $u \in C$ ou $v \in C$? (Ou seja, cada aresta de G possui ao menos uma extremidade em C.)

ISSN: 2675-3588

Este problema servirá de alicerce para a construção do *Hitting Set* nas seções subsequentes, pois a prova de NP-Completude será obtida por meio de uma redução polinomial de *Vertex Cover* para *Hitting Set*. De maneira geral, o *Hitting Set* recebe como entrada um universo finito de elementos e uma coleção de subconjuntos desse universo, e pergunta se existe um subconjunto H com tamanho limitado por k que intercepte todos esses subconjuntos, isto é, que contenha pelo menos um elemento em comum com cada um deles. Na prática, o *Hitting Set* pode ser visto como uma generalização de problemas de cobertura em grafos, na qual arestas e vértices são substituídos por subconjuntos e elementos de um universo arbitrário.

Problema: HITTING SET (HS)

Entrada: Um conjunto finito U (universo), uma coleção $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ de subconjuntos de U e um inteiro positivo $k \in \mathbb{N}$.

Questão: Existe um subconjunto $H \subseteq U$ com cardinalidade $|H| \leq k$ tal que H intercepte todos os conjuntos de $\mathcal{S}$? (Ou seja, $H \cap S_i \neq \emptyset$ para todo $S_i \in \mathcal{S}$).

III. Trabalhos Relacionados

A fundamentação deste artigo baseia-se em três eixos principais: desenvolvimentos recentes em algoritmos e aplicações para o problema de *Hitting Set*, abordagens contemporâneas para ensino de complexidade computacional e o apoio da literatura basilar para estabelecer relação com as práticas pedagógicas na disciplina de Teoria da Computação. A seguir, destacam-se as obras diretamente relacionadas à proposta.

Do ponto de vista técnico, estudos recentes sobre geração de *Hitting Sets* mínimos e sobre aplicações em biologia de sistemas evidenciam que o *Hitting Set* permanece um problema central tanto na pesquisa teórica quanto em cenários aplicados por Gainer-Dewar, Vera-Licona e Haus[5, 6]. Esses trabalhos discutem algoritmos em contextos reais, reforçando a importância de compreender, mesmo em nível introdutório, por que o problema é intratável e quais estratégias práticas são adotadas na literatura recente.

É importante ressaltar que busca-se a inspiração basilar em trabalhos como o de Garey e Johnson [2], referência em intratabilidade para denotarmos o entendimento em materiais recentes, pois fornecem a definição formal do *Hitting Set* e sua classificação como problema NP-Completo, baseada na equivalência com o *Set Cover*. O trabalho seminal de Karp [1] é utilizado para contextualizar historicamente as reduções polinomiais, técnica central aplicada neste artigo, bem como para situar o *Hitting Set* no panorama dos problemas intratáveis.

No eixo de aplicações, resultados como os de Gainer-Dewar e Vera-Licona [5] e de Haus et al. [6] ilustram o uso de *hitting sets* na análise de redes biológicas e em ambientes de alto desempenho, o que contribui para motivar o estudo do problema junto a estudantes da área de computação e a visão da aplicabilidade no tom pedagógico. Ao mostrar que a mesma estrutura combinatória aparece em contextos atuais de pesquisa, esses trabalhos ajudam a conectar o conteúdo teórico da disciplina com problemas concretos de interesse científico e tecnológico.

Os trabalhos de Chvátal [7], Ammann e Offutt [8] trazem abordagens que alimentam a discussão de aplicabilidade prática. Chvátal discute heurísticas gulosas como forma de contornar a intratabilidade em problemas de cobertura, enquanto Ammann e Offutt conectam a teoria abstrata à prática de testes de software, justificando a relevância do *Hitting Set* para a formação de futuros profissionais.

Do ponto de vista pedagógico, o artigo de Lassance, Bianchini e Santos [3], que descreve o "Ciclo de Seminários em Teoria da Computação", serviu como referência metodológica direta. Dessa experiência, foi adotada a ideia de decompor a prova em "Problema Atacado" (Vertex Cover) e "Problema Alvo" (Hitting Set), bem como a ênfase na construção de recursos visuais e exemplos guiados como suporte à aprendizagem em disciplinas introdutórias. Em complemento, trabalhos que exploram o uso de visualizações, animações e ferramentas interativas para o ensino de NP-Completude, indicam uma tendência recente de tornar as reduções mais acessíveis por meio de abordagens ativas e multimodais, Crescenz e Marchetti [9, 10].

Em conjunto, essas referências não apenas sustentam a prova teórica apresentada nas seções seguintes mas abordam de maneira recente, e fundamentam a escolha de uma abordagem fortemente didática, alinhada com práticas contemporâneas de ensino de complexidade e com aplicações atuais do problema de *Hitting Set*.

IV. Descrição do Problema

O *Hitting Set* é um dos problemas mais dinâmicos na teoria da complexidade, justamente pela sua objetividade que serve como um bom drive de verificação entre problemas tratáveis e intratáveis. Sua classificação como NP-Completo foi estabelecida originalmente por Richard Karp em sua lista seminal de 21 problemas [1], devido à sua equivalência direta com o problema *Set Cover*. Posteriormente, Garey e Johnson [2] consolidaram sua importância como um problema ilustre para provas de redução, dada a sua estrutura combinatória limpa e versátil.

Para compreender a natureza deste problema, é essencial distinguir inicialmente entre as versões de otimização e decisão. Em sua forma natural, o *Hitting Set* é um problema de otimização que busca responder: "Qual é o *menor* número de elementos necessários para atingir todos os subconjuntos?". No entanto, para a classificação na classe $\mathcal{NP}$, utilizamos a versão de decisão, que impõe um limite superior k. A questão central torna-se: "É possível atingir todos os conjuntos utilizando no máximo k elementos?".

Formalmente, seguindo a notação proposta por Garey e Johnson [2], seja U um conjunto finito, chamado de *universo*, e seja $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ uma coleção finita de subconjuntos de U, isto é, $S_i \subseteq U$ para todo $1 \leq i \leq m$. Seja ainda $k \in \mathbb{N}$ um inteiro não negativo. O problema HITTING SET na forma de decisão é definido da seguinte maneira:

TABELA 1: INTRATABILIDADE: COMPARAÇÃO DO NÚMERO DE OPERAÇÕES NECESSÁRIAS CONFORME A ENTRADA n CRESCE.

Entrada (n)	Polinomial (n^2)	Exponencial (2^n)
10	100	1.024
30	900	$\approx$ 1 bilhão
50	2.500	$\approx 10^{15}$
100	10.000	$\approx 10^{30}$

TABELA 2: COMPARAÇÃO ENTRE ABORDAGENS PARA O HITTING SET. ($n = |U|$, ASSUMINDO $m \approx n$ PARA SIMPLIFICAÇÃO).

Abordagem	Complexidade de Tempo	Qualidade da Solução
Enumeração Completa (Força Bruta)	$O(2^n)$	Exata (ótima, mas inviável para n grande)
Construção Iterativa Orientada (passo a passo)	$O(n^2)$	Em geral não ótima, mas utilizável na prática

Um subconjunto $H \subseteq U$ que satisfaz $H \cap S_i \neq \emptyset$ para todo $S_i \in \mathcal{S}$ é chamado de *hitting set* (ou *conjunto atingidor*) para a coleção $\mathcal{S}$. Assim, o objetivo do problema é decidir se existe um hitting set de tamanho no máximo k. Nesta notação, U representa o conjunto de todos os elementos disponíveis, $\mathcal{S}$ é a família de subconjuntos que devem ser "atingidos", H é o conjunto solução candidato e k é o limite máximo permitido para o tamanho de H.

A complexidade computacional inerente a esta definição impõe desafios práticos severos. Para encontrar a solução com exatidão, a abordagem mais intuitiva é a chamada Força Bruta. O conceito é simples: o computador testa todas as combinações possíveis de elementos para ver qual é a menor que funciona. É como tentar abrir um cadeado de segredo testando todas as senhas, uma por uma: 000, 001, 002... Embora a Força Bruta seja correta, ela é extremamente lenta por sua natureza combinatória. Dado um universo U com n elementos, o número total de subconjuntos possíveis que podem ser formados é 2^n (incluindo o conjunto vazio). O algoritmo precisa, essencialmente, percorrer todas as 2^n possibilidades, ou pelo menos um grande subconjunto delas, para encontrar a solução ótima. O número de combinações cresce exponencialmente (2^n), tornando a resolução inviável para qualquer instância que não seja muito pequena [2]. Dessa forma, a **Tabela 1** ilustra como esse tempo de execução aumenta rapidamente, por meio da comparação do número de operações necessárias conforme a entrada cresce, baseando-se em análises assintóticas clássicas [4]. A potência 2^n aparece porque, para cada elemento do universo U com n elementos, há duas possibilidades independentes: ou ele entra no subconjunto H ou não entra.

Diante da impossibilidade de verificar todas as opções, pois o número de combinações cresce de forma exponencial, como ilustrado na **Tabela 1**, cientistas da computação recorrem a algoritmos de aproximação [7]. A ideia é aceitar abrir mão da garantia de solução ótima em troca de um algoritmo que rode em tempo polinomial e produza soluções "boas o suficiente" na prática.

De forma explicativa e alinhada à metodologia de ensino proposta por Lassance, Bianchini e Santos [3], adotamos aqui uma estratégia passo a passo voltada ao entendimento dos estudantes. A ideia é construir a solução de forma interativa, sempre observando a instância em um quadro ou diagrama: em cada passo, o aluno identifica quais conjuntos ainda não foram atingidos e escolhe um elemento que ajude a cobrir os casos restantes, atualizando o desenho a cada escolha.

Do ponto de vista algorítmico, essa construção iterativa pode ser vista como uma versão simplificada de uma abordagem gulosa clássica [7]: a cada passo, escolhe-se um elemento que contribui para cobrir muitos conjuntos ainda não atingidos. Esse tipo de estratégia não garante, em geral, a melhor solução possível (ao contrário da Força Bruta, que é exata porém inviável para entradas grandes [2]), mas apresenta duas vantagens fundamentais: (i) seu tempo de execução é polinomial, o que a torna utilizável em instâncias reais, e (ii) existem resultados teóricos que limitam quão pior a solução obtida pode ser em relação à solução ótima [7]. Para fins de análise didática e comparação de crescimento, assumimos aqui um cenário onde o número de subconjuntos m é proporcional ao tamanho do universo n, permitindo expressar a complexidade apenas em função de n. A **Tabela 2** resume esse contraste entre a exatidão da Força Bruta e a praticidade das abordagens iterativas.

Contudo, em cenários industriais onde a exatidão é inegociável (como no diagnóstico médico ou em configurações de segurança crítica), depender apenas de aproximações pode ser insuficiente. Para esses casos, a indústria recorre à *Complexidade Parametrizada* (FPT - *Fixed-Parameter Tractability*). Nesta abordagem, a complexidade é analisada em função de dois valores: o tamanho da entrada n e um parâmetro fixo k — que, neste problema, corresponde ao tamanho da solução buscada. A estratégia é confinar a "explosão combinatória" exclusivamente a esse parâmetro k, mantendo o tempo polinomial em relação a n. Algoritmos FPT com complexidade do tipo $O(2^k \cdot n)$ exemplificam bem essa vantagem: considere uma base de dados com $n = 1.000$ elementos onde buscamos um subconjunto de tamanho $k = 10$. Enquanto a abordagem FPT exigiria apenas $\approx 10^6$ operações, sendo resolvida em cerca de 1 milissegundo (supondo 10^9 operações/s), a força bruta (2^n) exigiria $2^{1.000}$ operações, o que levaria um tempo superior à idade do universo para ser concluído. Essa abordagem permite lidar com a intratabilidade de forma cirúrgica em instâncias reais, sem sacrificar a precisão dos resultados [11].

Apesar da dificuldade geral, existem exceções interessantes. Se restringirmos a instância de modo que cada subconjunto em $\mathcal{S}$ tenha tamanho máximo 2 (isto é, $|S_i| \leq 2$ para todo $S_i \in \mathcal{S}$), o problema torna-se idêntico ao *Vertex Cover*, que, apesar de ainda ser NP-Completo, permite análises mais detalhadas e soluções aproximadas bem estudadas em questão literária.

Para visualizar a definição formal na prática, considere a instância apresentada na Figura 2. Neste exemplo, temos o universo $U = \{1,2,3,4,5\}$ e a coleção $\mathcal{S} = \{\{1,2,3\},\{2,4\},\{3,4\},\{4,5\}\}$ com $k = 2$. A resposta é afirmativa, pois o conjunto $H = \{2,4\}$ possui tamanho 2 e intersecta todos os subconjuntos de $\mathcal{S}$. Os elementos da

ISSN: 2675-3588

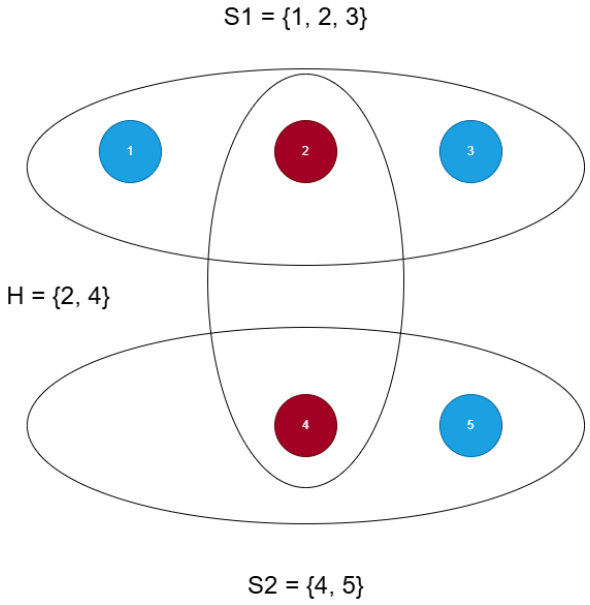


Figura 2: Representação visual de S, no qual $H = \{2,4\}$ é *Hitting Set*.

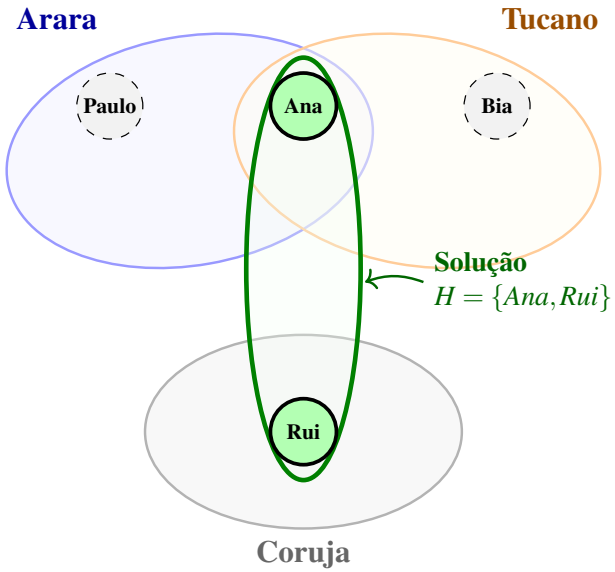


Figura 3: Ana cobre Arara e Tucano; Rui cobre a Coruja. O envelope verde destaca o conjunto solução $H = \{Ana, Rui\}$.

solução $H = \{2,4\}$ estão destacados em vermelho; note que cada elipse (conjunto) contém pelo menos um elemento vermelho.

Apesar da dificuldade geral, existem exceções interessantes. Se restringirmos a instância de modo que cada subconjunto em $\mathcal{S}$ tenha tamanho máximo 2 (isto é, $|S_i| \leq 2$ para todo $S_i \in \mathcal{S}$), o problema torna-se idêntico ao *Vertex Cover*, que, apesar de ainda ser NP-Completo, permite análises mais detalhadas e soluções aproximadas bem estudadas na literatura.

Para visualizar a definição formal na prática, considere a instância apresentada na **Figura 2**. Neste exemplo, temos o universo $U = \{1,2,3,4,5\}$ e a coleção $\mathcal{S} = \{\{1,2,3\},\{4,5\}\}$ com $k = 2$. A resposta é afirmativa, pois o conjunto $H = \{2,4\}$ possui tamanho 2 e intersecta todos os subconjuntos de $\mathcal{S}$. Os elementos da solução $H = \{2,4\}$ estão destacados em vermelho; note que cada elipse (conjunto) contém pelo menos um elemento vermelho.

Para facilitar a intuição sobre a complexidade combinatória, propomos uma analogia original denominada "O Dilema dos Observadores". A ideia é traduzir a definição formal do *Hitting Set* para uma narrativa concreta, em que os elementos do universo e os subconjuntos ganham interpretação no mundo real. Essa analogia é inspirada no clássico problema de *Crew Scheduling* (Escalonamento de Tripulações), citado por Garey e Johnson [2] como uma aplicação canônica de problemas de cobertura de conjuntos.

> "Uma equipe de biólogos precisa confirmar a presença de 5 espécies raras de pássaros (S_1 a S_5) em uma reserva. Eles têm 10 observadores disponíveis. Cada observador é especialista em identificar um subconjunto diferente de espécies. A equipe tem orçamento para contratar no máximo k observadores. A pergunta é: *é possível formar um time com $\leq k$ pessoas que identifique* ***todas*** *as espécies?*"

Nessa analogia, o conjunto de observadores é o universo U e cada espécie define um subconjunto S_i de quem pode avistá-la. Um conjunto de observadores contratados é um *Hitting Set*. A **Figura 3** ilustra uma instância desse dilema.

A personagem Ana (em vermelho) é uma generalista que cobre duas espécies (Arara e Tucano). Porém, ao escolhê-la, ainda precisamos cobrir a Coruja, que só é vista pelo Rui. Assim, uma solução possível seria o time $\{Ana, Rui\}$ (tamanho 2). Outra opção seria ignorar a Ana e contratar apenas especialistas dedicados: $\{Paulo, Bia, Rui\}$ (tamanho 3).

O "dilema" computacional é que não existe uma regra simples (como "sempre escolha quem cobre mais") que garanta a melhor solução em todos os casos. O computador precisa verificar as diversas combinações (Generalistas vs. Especialistas) para garantir que o orçamento k seja respeitado.

Além do interesse teórico, o *Hitting Set* modela desafios reais onde a eficiência é crítica. Na bioinformática, é aplicado na seleção de marcadores genéticos [1]. Na engenharia de software, é utilizado para minimizar suítes de teste [8]. Como o problema é NP-Completo, a inviabilidade da força bruta valida o uso das heurísticas de aproximação como a abordagem padrão na indústria.

Essa analogia faz sentido em relação ao problema de *Hitting Set* porque traduz, quase literalmente, cada componente da definição formal para elementos intuitivos da história, permitindo ao estudante "ver" o problema em vez de apenas manipulá-lo simbolicamente. Do ponto de vista matemático, o universo U do *Hitting Set* corresponde ao conjunto de observadores disponíveis, enquanto cada subconjunto $S_i \in \mathcal{S}$ é interpretado como o grupo de observadores capazes de identificar a espécie i. Um *hitting set* $H \subseteq U$ é um conjunto com elementos que intersecta todos os S_i; na analogia, isso significa escolher um time de observadores tal que, para cada espécie rara, pelo menos um membro do time consiga identificá-la. O parâmetro k que limita o tamanho de H é modelado diretamente pelo orçamento máximo de observadores que podem ser contratados.

Além de respeitar essa correspondência estrutural, o dilema também ajuda a construir intuição sobre a complexidade do problema. Por um lado, evidencia o caráter combinatório: há muitas formas possíveis de escolher subconjuntos de observadores, e nem todas cobrem todas as espécies, o que espelha o grande espaço de soluções candidatas no *Hitting Set*. Por outro lado, ilustra o perigo de decisões puramente locais: escolher a observadora "generalista" Ana parece uma boa escolha quando se olha apenas para o número de espécies cobertas, mas não resolve o caso da Coruja, exigindo a presença do especialista Rui. Dessa forma, a narrativa mostra que a melhor decisão local nem sempre leva à melhor solução global, um ponto central em problemas NP-Difíceis.

É imperativo, contudo, delimitar o escopo desta analogia lúdica para evitar simplificações excessivas. O "Dilema dos Observadores" atua estritamente como um facilitador para a compreensão do *enunciado* e das restrições do problema, não substituindo a formalização matemática necessária para a análise de complexidade. Em cenários cotidianos ou administrativos, como o descrito na narrativa, a intuição humana frequentemente encontra padrões que facilitam a resolução. No entanto, a classificação de NP-Completude lida com instâncias arbitrárias de "pior caso", onde tais padrões intuitivos inexistem ou são enganosos. Portanto, a analogia serve como porta de entrada cognitiva, mas o rigor algébrico — detalhado na demonstração da Seção V — permanece insubstituível para a validação científica da intratabilidade.

Em síntese, o "Dilema dos Observadores" funciona como um modelo mental que o aluno pode reutilizar nas seções seguintes: sempre que se deparar com a notação U, $\mathcal{S}$, H e k, pode lembrar de observadores, espécies e orçamento, o que reduz a carga cognitiva e facilita a compreensão das provas formais.

V. Demonstração e Contribuições Técnicas

Esta seção apresenta a sistematização da prova de NP-Completude do *Hitting Set*. Diferentemente dos manuais técnicos que priorizam a concisão, optamos aqui por uma abordagem expandida, detalhando os passos lógicos que frequentemente são omitidos na literatura especializada [2].

Para classificar formalmente um problema como NP-Completo, é necessário satisfazer simultaneamente duas condições: provar que $HS \in \mathcal{NP}$, e que $HS \in$ NP-Difícil. Essas provas serão feitas a seguir.

Lema 1. *O problema HS pertence a $\mathcal{NP}$.*

Proof. Seguindo esse fluxo lógico, o primeiro passo é demonstrar que o problema pertence à classe $\mathcal{NP}$. Isso exige a existência de um algoritmo que, dada uma solução candidata (certificado), consiga verificar sua validade em tempo eficiente, conforme a definição formal de verificadores polinomiais estabelecida por Sipser [4]. No contexto do *Hitting Set*, considere uma instância definida por um universo de elementos U, uma coleção S de subconjuntos de U e um inteiro k [2]. O certificado é um subconjunto candidato $H \subseteq U$. O algoritmo verificador recebe a instância (U,S,k) e o certificado H, respondendo "Sim" apenas se duas condições forem satisfeitas: (1) o tamanho de H respeita o limite k (i.e., $|H| \leq k$); e (2) H intersecta todos os subconjuntos de S. Abaixo apresentamos o algoritmo que realiza essa verificação:

```
Algoritmo Verificador(U, S, H, k)
Início
   Se (tamanho(H) > k) então
      Retorne Falso;
   Fim-Se
   Para cada S_i em S faça
      Verifica se a interseção é vazia
      Se (H ∩ S_i == ∅) então
         Retorne Falso;
      Fim-Se
   Fim-Para
   Retorne Verdadeiro;
Fim
```

Para realizar a análise de eficiência deste algoritmo (tecnicamente chamada de análise assintótica [4]), definimos n como o tamanho total da entrada recebida pelo algoritmo (a soma dos tamanhos de U, S, H e a representação de k). É fácil analisar que a verificação de tamanho é uma operação linear $O(n)$, pois, no pior caso, o algoritmo precisa percorrer a lista de elementos de H para contá-los, e o tamanho de H nunca excede o tamanho total da entrada n. Já a Verificação de Cobertura é a etapa dominante: sua estrutura de repetição obriga a comparação dos elementos de H com os de cada subconjunto em S, resultando em uma complexidade quadrática $O(n^2)$. Como n^2 é um polinômio, garantimos que a verificação é eficiente.

Desta forma, no pior cenário possível, essa estrutura faz o algoritmo comparar sistematicamente os elementos, resultando em um número total de operações proporcional ao produto $m \times n$ (onde n é o tamanho do universo). Em termos de complexidade, isso é representado pela notação $O(n^2)$, indicando que o tempo cresce quadraticamente em relação ao tamanho da entrada [4]. Como uma função quadrática é um polinômio (e não uma exponencial como 2^n), garantimos que a verificação é computacionalmente viável, confirmando assim que o *Hitting Set* pertence à classe $\mathcal{NP}$ [2].

□

Uma vez estabelecida a NP-Pertinência, o próximo passo do núcleo da prova reside na demonstração de NP-Dificuldade. A estratégia utilizada é a redução polinomial, onde transformamos instâncias de um problema conhecido como NP-Difícil (**Problema Alvo**) em instâncias do problema que queremos classificar (**Problema Atacado**). Para este artigo, reduziremos o *Vertex Cover* ao *Hitting Set* ($VC \leq_p HS$).

Lema 2. *O problema HS é NP-Difícil.*

Proof. Com o problema de partida formalmente definido, passamos à construção da redução. O objetivo desta etapa é demonstrar um algoritmo que transforme, em tempo polinomial, qualquer instância de *Vertex Cover* em uma instância equivalente de *Hitting Set* [1]. Essa transformação deve garantir que a estrutura topológica do grafo seja preservada na forma de conjuntos, de modo que a existência de uma solução em um problema implique diretamente a existência de solução no outro [4].

ISSN: 2675-3588

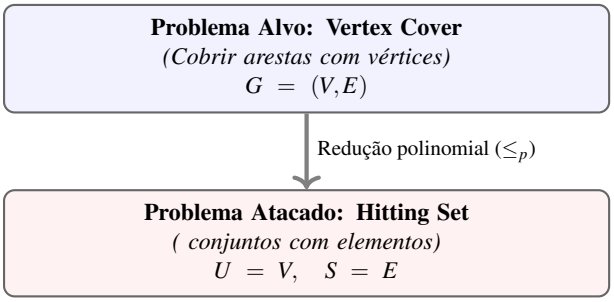


Figura 4: Esquema da redução: transformamos a estrutura do grafo (Atacado) em uma estrutura de conjuntos (Alvo).

Seguindo este raciocínio, no *Vertex Cover* devemos garantir que cada aresta seja coberta por um vértice. No *Hitting Set*, a obrigação é garantir que cada subconjunto seja interceptado por um elemento. Portanto, a estratégia consiste em converter cada aresta (que conecta dois vértices) em um subconjunto (contendo dois elementos) [2]. O esquema conceitual dessa estratégia é apresentado na **Figura 4**.

É fundamental observar que a redução proposta realiza uma tradução da estrutura topológica do grafo para uma estrutura combinatória de conjuntos. Neste contexto, a estrutura combinatória refere-se à organização de elementos baseada estritamente em relações de pertinência e agrupamento, abstraindo qualquer noção de conectividade espacial ou adjacência visual típica dos grafos. A relação de adjacência entre vértices, representada pelas arestas, é remapeada para uma relação de inclusão em subconjuntos.

Desta maneira, a restrição topológica de "cobrir uma aresta" (garantir que uma conexão seja vigiada) é reformulada como a necessidade algébrica de "interceptar um subconjunto" (garantir que um grupo contenha um elemento selecionado). O mapeamento é definido da seguinte maneira: o universo U é constituído pelos vértices de V; a coleção S é formada convertendo cada aresta $\{u,v\}$ em um subconjunto contendo exatamente esses vértices; e o parâmetro de otimização k' preserva seu valor original ($k' = k$). A **Figura 5** ilustra essa transformação por meio da conversão da aresta $\{1,2\}$ no subconjunto $S_i = \{1,2\}$.

A corretude desta redução depende da prova de que a instância construída preserva a resposta da original. Demonstramos isso através de duas proposições:

Proposição 1 (Ida $\Rightarrow$): Se G possui um Vertex Cover de tamanho k, então S possui um Hitting Set de tamanho k.

Seja $C \subseteq V$ o Vertex Cover. Escolhemos $H = C$. Para qualquer conjunto $S_i \in S$, sabemos pela construção que ele corresponde a uma aresta $\{u,v\} \in E$. Como C cobre todas as arestas, ele deve conter u ou v. Logo, H contém u ou v, interceptando S_i. Portanto, H é um Hitting Set válido.

Proposição 2 (Volta $\Leftarrow$): Se S possui um Hitting Set de tamanho k, então G possui um Vertex Cover de tamanho k.

Seja $H \subseteq U$ o Hitting Set. Escolhemos $C = H$. Para qualquer aresta $e = \{u,v\} \in E$, existe um conjunto correspondente $S_e = \{u,v\}$ em S. Como H atinge todos os conjuntos, ele deve conter u ou v. Logo, C contém uma extremidade da aresta e. Portanto, C cobre todas as arestas de G.

A Figura 6 ilustra a equivalência lógica. No caso mostrado, a aresta $\{1,2\}$ é coberta no *Vertex Cover* pelo vértice 1 (destacado em vermelho). Na construção do *Hitting Set*, o conjunto correspondente $S_e = \{1,2\}$ é atingido pelo mesmo elemento 1, preservando a equivalência entre as duas estruturas.

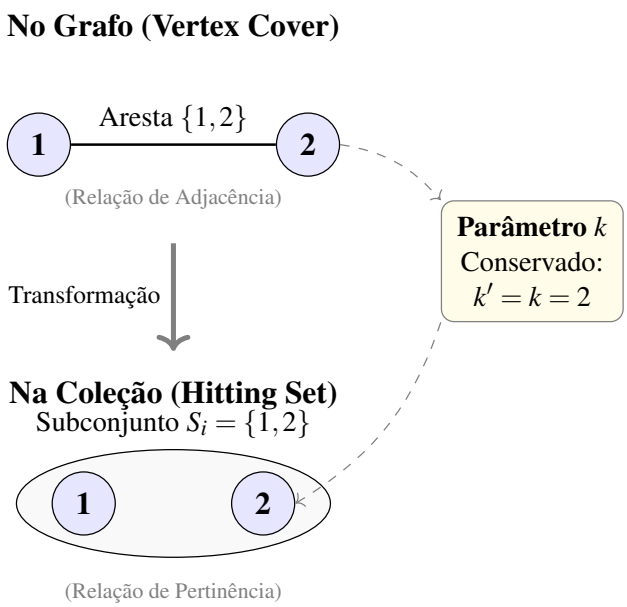


Figura 5: Visualização da Construção: A aresta conectando 1 e 2 no grafo é convertida em um conjunto $S_i = \{1,2\}$.

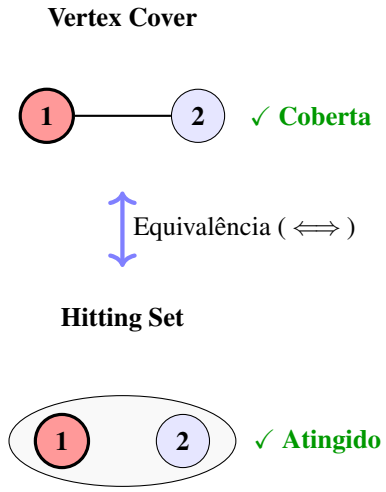


Figura 6: Cobrir a aresta $\{1,2\}$ com o vértice 1 (vermelho) corresponde a atingir o conjunto $\{1,2\}$ com o elemento 1.

Desta forma, a partir das provas demonstradas da **Proposição 1** e **Proposição 2**, podemos concluir que HS $\in$ NP-Difícil.

□

Lema 3. *O problema HS é NP-Completo.*

Proof. Para demonstrar que um problema é NP-Completo, é preciso demonstrar que ele pertença simultaneamente as classes $\mathcal{NP}$ e NP-Difícil. Essas demonstrações foram feitas e provadas respectivamente no **Lema 1** e **Lema 2**.

Desta forma, podemos concluir que o problema *Hitting Set* é NP-Completo.

□

Como contribuição pedagógica final, é importante alertar sobre uma armadilha comum no estudo de reduções: a direção da prova. Estudantes frequentemente tentam reduzir

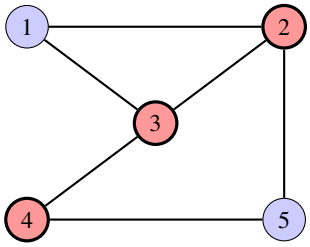


Figura 7: ANTES (Vertex Cover): O grafo de entrada com $k = 3$. Os vértices em vermelho $\{2,3,4\}$ cobrem todas as arestas.

Universo: $U = \{1,2,3,4,5\}$

Coleção S (baseada nas arestas):
$\{1,2\},\{1,3\},\{2,3\},$
$\{2,5\},\{3,4\},\{4,5\}$

Solução Mapeada:
$H = \{2,3,4\}$

Figura 8: DEPOIS (Hitting Set): A instância resultante. O conjunto $H = \{2,3,4\}$ intercepta todos os subconjuntos listados.

o problema novo para o problema conhecido ($HS \leq_p VC$). Isso provaria apenas que o HS é "fácil" o suficiente para ser resolvido pelo VC, mas não que ele é "difícil". A prova de NP-Dificuldade exige o oposto: mostrar que o problema novo é capaz de simular qualquer instância do problema difícil conhecido ($VC \leq_p HS$).

VI. Resultados e Reflexões

A elaboração deste artigo permitiu consolidar o entendimento sobre a hierarquia de complexidade e as técnicas de redução polinomial. Mais do que a demonstração formal, o principal produto deste trabalho é a sistematização didática apresentada, que busca preencher lacunas de compreensão comuns em estudantes iniciantes. A visualização do mapeamento entre instâncias mostrou-se uma ferramenta poderosa para tangibilizar a abstração algébrica.

Uma reflexão crítica sobre a metodologia adotada revela que a escolha do *Vertex Cover* como problema de partida (Problema Atacado) foi determinante para a clareza da exposição. Embora a literatura clássica frequentemente utilize reduções a partir de problemas lógicos como o 3-SAT, essa abordagem exige que o estudante transite entre o domínio da lógica booleana e a teoria dos conjuntos, o que adiciona uma carga cognitiva extra. Ao optarmos por uma redução grafo-para-conjunto ($VC \leq_p HS$), mantivemos a natureza visual do problema, permitindo que a transformação seja verificada "a olho nu", como ilustrado na sequência da **Figura 7** e **Figura 8**.

A construção dessas contribuições pedagógicas foi o foco central. Em vez de presumir conhecimento prévio, dedicamos as seções iniciais a explicar termos essenciais utilizando analogias. Um dos pontos altos foi o uso do "Dilema dos Observadores" para explicar os fundamentos teóricos: utilizamos essa analogia para concretizar que verificar uma solução (conferir a equipe contratada) é rápido, mas encontrar a solução ótima (testar todas as combinações) é exponencialmente difícil. Essa distinção é crucial para que o estudante compreenda a natureza da classe $\mathcal{NP}$ não como uma medida de "impossibilidade", mas como uma medida de "custo de busca".

Ainda sobre a estratégia lúdica, é pertinente observar que o "Dilema dos Observadores" também serve para ilustrar as limitações das abordagens intuitivas. Em sala de aula, é comum que alunos sugiram algoritmos gulosos (como contratar sempre a pessoa mais versátil) como solução geral. A modelagem do problema permitiu demonstrar que, em cenários de complexidade NP-Completa, a intuição local falha diante da necessidade de uma otimização global, validando a necessidade de rigor matemático na análise de algoritmos.

No entanto, o processo de elaboração deste material não foi isento de dificuldades. O maior desafio encontrado não foi a complexidade técnica da prova em si — pois a redução de *Vertex Cover* é direta — mas sim o desafio de transposição didática: explicar os fundamentos sem recorrer a jargões herméticos que afastam o leitor iniciante. A estratégia adotada de explicar cada conceito técnico (como a análise assintótica) imediatamente antes de sua aplicação mostrou-se essencial para manter a clareza e a acessibilidade do texto.

Quanto à aplicabilidade acadêmica, este trabalho foi feito para servir como um material complementar para futuros alunos da disciplina de Teoria da Computação. Acreditamos que a exposição visual da redução e a discussão sobre as nuances entre decisão e otimização oferecem um ponto de entrada mais suave para o tema. Tanto este artigo quanto as referências discutidas podem ser usados como um guia introdutório e acessível para quem precisa entender como uma prova de NP-Completude é estruturada na prática, cumprindo o objetivo de facilitar o aprendizado e desmistificar a teoria.

VII. Conclusão

A elaboração deste estudo permitiu atingir o objetivo principal de demonstrar a NP-Completude do problema *Hitting Set* de forma pedagógica seguindo o rigor exigido pela literatura clássica. A prova foi estruturada em duas etapas fundamentais: a verificação de pertinência à classe $\mathcal{NP}$, realizada através da análise de um algoritmo verificador polinomial, e a demonstração de NP-Dificuldade, executada por meio da redução canônica a partir do *Vertex Cover*. Este resultado teórico não é apenas um rótulo classificatório; ele carrega uma implicação prática profunda: a menos que $\mathcal{P} = \mathcal{NP}$, não existem algoritmos eficientes para resolver o *Hitting Set* de forma exata em todos os casos, validando a necessidade de abordagens aproximadas.

Do ponto de vista pedagógico, o material foi aplicado em uma turma de Teoria da Computação, envolvendo aproximadamente 20 estudantes de graduação.[1] Em uma atividade de seminário, os alunos foram convidados a reconstruir a redução $VC \leq_p HS$ utilizando os diagramas apresentados e a reprodução guiada das etapas da prova, antes do contato direto com os livros-texto formais. Nessa dinâmica, as interações e discussões em sala facilitaram o compartilhamento de diferentes formas de explicar a redução, em uma linguagem mais próxima dos próprios estudantes, mediadas pela equipe de pesquisadores. Observou-se que os alunos passaram a demonstrar maior segurança para

[1] Relato de aplicação didática conforme descrito na seção de Resultados e Reflexões.

ISSN: 2675-3588

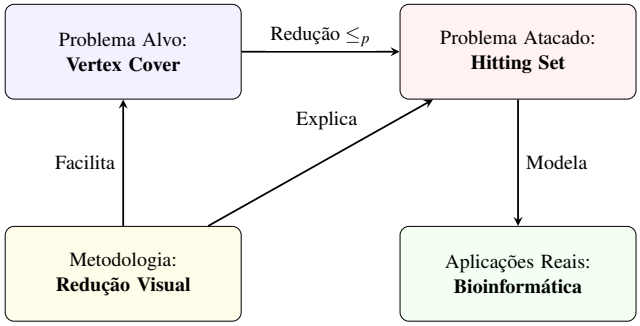


Figura 9: Mapa síntese da abordagem: A metodologia visual conecta o problema base ao alvo.

explicar, com suas próprias palavras, o papel do certificado em $\mathcal{NP}$ e o encadeamento lógico da redução, apoiados por estratégias visuais organizadas em slides interativos, com tempo de exposição limitado para evitar sobrecarga cognitiva. Ainda que esses registros não constituam um estudo quantitativo rigoroso, eles fornecem indícios qualitativos de que a abordagem visual e narrativa contribuiu para reduzir a sensação de "salto conceitual" frequentemente associada às provas de NP-completude [9, 10].

Para além da demonstração matemática, o contributo mais expressivo deste trabalho reside na sua proposta pedagógica. Conforme as diretrizes da disciplina, buscou-se transpor a barreira da abstração que frequentemente dificulta o aprendizado de Teoria da Computação. A introdução do "Dilema dos Observadores" serviu como uma ponte cognitiva, traduzindo a aridez da notação de conjuntos para um problema tangível de gestão de recursos. Essa analogia facilitou a intuição sobre a assimetria fundamental da complexidade: a facilidade de verificar uma solução dada (auditar uma equipe contratada) em contraste com a dificuldade de encontrar a solução ótima (testar todas as combinações de equipes).

Para consolidar a jornada de aprendizado proposta, a **Figura 9** apresenta um mapa conceitual que resume a estrutura lógica desenvolvida no artigo, conectando a teoria de base, a prova de redução e as aplicações práticas.

Embora a sistematização proposta tenha êxito em seus objetivos didáticos, o trabalho apresenta limitações no seu escopo, concentrando-se majoritariamente no aspecto teórico da classificação de complexidade. Não foram exploradas, nesta etapa, implementações computacionais de algoritmos de aproximação como a Heurística Gulosa [7] ou algoritmos parametrizados (FPT), que constituem a abordagem padrão para lidar com a intratabilidade do problema em cenários industriais reais [8, 1]. Adicionalmente, a redução restringiu-se ao caminho clássico via *Vertex Cover*, sem explorar reduções alternativas que poderiam oferecer outras perspectivas.

Como desdobramento natural deste estudo, trabalhos futuros podem focar na vertente experimental, implementando e comparando o desempenho de algoritmos exatos (para instâncias pequenas) versus algoritmos aproximativos (para instâncias grandes). Outra via promissora seria o aprofundamento em classes especiais de instâncias, como aquelas com restrições de cardinalidade nos subconjuntos, investigando cenários onde o problema se torna tratável e enriquecendo ainda mais o repertório de exemplos didáticos disponíveis para o ensino de Computação.

Referências

[1] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*. Springer, 1972, pp. 85–103.

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

[3] Y. M. Lassance, G. d. B. Bianchini, and T. D. Santos, "Reflexões e práticas pedagógicas no escopo da disciplina de teoria da computação," *Academic Journal on Computing, Engineering and Applied Mathematics (AJCEAM)*, vol. 6, no. 2, pp. 10–17, 2025.

[4] M. Sipser, *Introduction to the Theory of Computation*, 3rd ed. Cengage Learning, 2012.

[5] A. Gainer-Dewar and P. Vera-Licona, "The minimal hitting set generation problem: Algorithms and computation," *SIAM Review*, vol. 58, no. 3, pp. 448–491, 2016.

[6] U.-U. Haus *et al.*, "Finding exact hitting set solutions for systems biology applications using heterogeneous gpu clusters," *Future Generation Computer Systems*, vol. 67, pp. 418–429, 2016.

[7] V. Chvátal, "A greedy heuristic for the set-covering problem," *Mathematics of Operations Research*, vol. 4, no. 3, pp. 233–235, 1979.

[8] P. Ammann and J. Offutt, *Introduction to Software Testing*. Cambridge University Press, 2016.

[9] P. Crescenzi, "Using avs to explain np-completeness," in *Proceedings of the 5th International Conference on Theory and Applications of Diagrams*. Springer, 2010, pp. 95–111.

[10] K. Marchetti *et al.*, "Redux: An interactive, dynamic knowledge base for teaching np-completeness," in *Proceedings of the 2024 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 2024, pp. 255–261.

[11] R. G. Downey and M. R. Fellows, *Fundamentals of Parameterized Complexity*, ser. Texts in Computer Science. Springer, 2013.

ISSN: 2675-3588

De 3-Coloração a Coloração de Arestas: Contribuições Pedagógicas para o Aprendizado no Escopo da Teoria da Computação

From 3-Coloring to Edge Coloring: Pedagogical Contributions for Learning in the Scope of Computation Theory

Ana Júlia Campos Vieira [1], Dallyla de Moraes Sousa [1], Daniel Martins da Silva [2] e Tanilson Dias dos Santos [1]

[1] *Universidade Federal do Tocantins, Ciência da Computação, Tocantins, Brasil*
[2] *Universidade Federal do Norte do Tocantins, Logística, Tocantins, Brasil*

Data de recebimento do manuscrito: 03/12/2025
Data de aceitação do manuscrito: 05/01/2026
Data de publicação: 10/02/2026

Resumo—Este artigo analisa a NP-completude do problema Edge Coloring através de uma cadeia de redução polinomial iniciada no problema 3-Coloring. A metodologia utiliza a construção de um Grafo Linha para demonstrar a pertinência à classe NP e estabelece a NP-Dificuldade via redução do 3-SAT, baseada em Holyer. São apresentadas definições formais, contexto histórico e revisão de trabalhos fundamentais em teoria da complexidade computacional. O principal resultado demonstra que Edge Coloring é NP-completo por meio de um método de redução claro e acessível. O trabalho oferece exemplos educacionais com ilustrações visuais e explicações passo a passo sobre gadgets lógicos. Este material serve como recurso de aprendizagem para auxiliar estudantes na compreensão de reduções polinomiais e conceitos de NP-completude em Ciência da Computação.

Palavras-chave—NP-completude, Coloração de Arestas, Coloração de Vértices, Reduções Polinomiais, Complexidade Computacional

Abstract—*This article examines the NP-completeness of the Edge Coloring problem through a polynomial reduction chain starting from the 3-Coloring problem. The methodology employs a Line Graph to demonstrate membership in NP and establishes NP-Hardness via reductions from 3-SAT, following Holyer's construction. We present formal definitions, historical context, and a review of fundamental works in computational complexity theory. The main result demonstrates that Edge Coloring is NP-complete using a clear and accessible reduction method. The work provides educational examples with visual illustrations and step-by-step explanations of logical gadgets. This material serves as a learning resource to help students understand polynomial reductions and NP-completeness concepts in computer science courses.*

Keywords—*NP-completeness, Edge Coloring, Vertex Coloring, Polynomial Reductions, Computational Complexity*

I. Introdução

A Teoria da Computação investiga os limites fundamentais dos algoritmos, sendo a classe dos problemas NP-Completos o cerne dos desafios práticos e teóricos da área. O estudo desta classe é essencial para entender a intratabilidade computacional, orientando o desenvolvimento de heurísticas e algoritmos de aproximação para problemas cruciais em otimização e inteligência artificial [1].

Nesse contexto, os problemas de coloração de grafos, como o 3-Coloring (Coloração de Vértices com 3 cores) e o Edge Coloring (Coloração de Arestas), são centrais. O 3-Coloring consiste em determinar se os vértices de um grafo podem ser coloridos com três cores de forma que vértices adjacentes não compartilhem a mesma cor. Historicamente, ele é um dos primeiros problemas a ter sua NP-completude provada por redução do SAT (Satisfiability) [2]. Já o Edge Coloring questiona se as arestas de um grafo podem ser coloridas com k cores de modo que arestas adjacentes (que compartilham um vértice) tenham cores distintas.

Ambos os problemas, apesar de conceitualmente distintos, compartilham uma estrutura computacional equivalente. Enquanto o 3-Coloring possui aplicações clássicas em planejamento e alocação de frequência, o Edge Coloring é

Dados de contato: Ana Júlia Campos Vieira, campos.ana@uft.edu.br

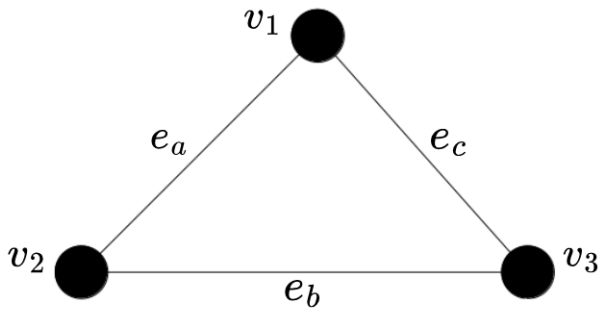


Figura 1: Grafo completo K_3 — exemplo de grafo não-direcionado com 3 vértices.

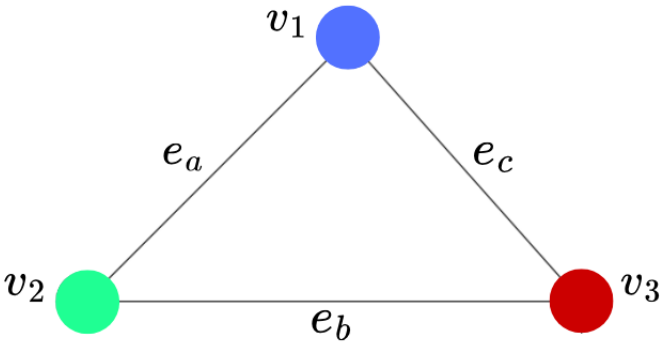


Figura 2: Grafo exemplo para 3-Coloring — estrutura com restrições de adjacência que permite coloração com 3 cores.

fundamental em problemas de escalonamento, alocação de recursos em redes e otimização de tempo [3].

Este artigo reúne e organiza demonstrações presentes na literatura sobre a NP-completude do *Edge Coloring*, com foco na clareza conceitual. A redução polinomial de 3-Coloring $\leq_p$ Edge Coloring é apresentada por meio do conceito de *Grafo Linha*, destacando os elementos centrais da transformação. O objetivo é oferecer um material que apoie o estudo das técnicas de redução e sua importância dentro da Teoria da Computação.

O trabalho expõe a prova e discute o raciocínio envolvido na construção, enfatizando aspectos que contribuem para o ensino de complexidade computacional. As seções seguintes apresentam os fundamentos necessários, a descrição da redução e as reflexões que surgem a partir dessa análise.

II. Preliminares

Começamos revisitando alguns conceitos fundamentais sobre grafos e problemas de coloração, essenciais para compreender o desenvolvimento deste trabalho.

Um *Grafo Não-Direcionado* é definido como uma estrutura $G = (V,E)$, onde V representa um conjunto finito e não vazio de vértices, e $E \subseteq \{(u,v) \mid u,v \in V, u \neq v\}$ é o conjunto de arestas que conectam esses vértices. Neste tipo de grafo, as arestas não possuem orientação, o que significa que a aresta (u,v) é idêntica à aresta (v,u), estabelecendo uma relação simétrica entre os vértices. Grafos não-direcionados são particularmente úteis para modelar relações mútuas, como amizades em redes sociais, conexões entre computadores em uma rede ou relações de adjacência em mapas.

Dentro dessa estrutura, dizemos que dois *vértices são adjacentes* se existe uma aresta conectando-os diretamente. Esta relação de adjacência é fundamental para definir a estrutura do grafo e suas propriedades. No grafo K_3 (Figura 1), todos os vértices são adjacentes entre si, formando um triângulo completo onde cada vértice possui grau 2 (duas conexões). O conjunto de vértices adjacentes a um vértice v é denominado sua *vizinhança*. A adjacência é uma relação binária que determina a conectividade direta no grafo, sendo essencial para definir caminhos, ciclos e outras propriedades estruturais.

Da mesma forma, duas *arestas são incidentes* quando compartilham um vértice em comum. Esta relação de incidência conecta o conceito de vértices com o conceito de arestas, criando a estrutura combinatória do grafo. No exemplo da Figura 1, qualquer par de arestas entre e_a, e_b e e_c compartilha um vértice, o que as torna incidentes entre si. Uma aresta é dita *incidente* a um vértice quando este vértice é uma de suas extremidades. O *grau* de um vértice é definido como o número de arestas incidentes a ele, sendo esta uma medida fundamental da centralidade do vértice no grafo.

Uma *coloração de vértices* é uma atribuição de cores aos vértices de um grafo por meio de uma função $c : V \to C$, onde C representa um conjunto finito de cores disponíveis. A notação $c(v)$ indica a cor atribuída ao vértice v, isto é, o resultado da função quando aplicada a esse vértice. Dessa forma, cada vértice recebe exatamente uma cor, permitindo analisar propriedades estruturais do grafo a partir dessa atribuição.

Uma coloração é dita *própria* quando nenhum par de vértices adjacentes compartilha a mesma cor. Em termos formais, isso significa que, para toda aresta $(u,v) \in E$, deve valer $c(u) \neq c(v)$. A condição $c(u)$ e $c(v)$ serem diferentes garante que vértices conectados não entrem em conflito de cor, constituindo o requisito fundamental em problemas clássicos como o 3-Coloring.

A Figura 2 apresenta um grafo simples usado para ilustrar relações de adjacência e incidência em um contexto de coloração. O exemplo evidencia como diferentes conexões afetam as possibilidades de coloração de vértices e de arestas.

O conceito de *Grafo Linha* estabelece uma dualidade entre vértices e arestas. Dado um grafo $G = (V,E)$, seu grafo linha $L(G) = (V_L, E_L)$ é construído mapeando cada aresta em E para um vértice em V_L, e dois vértices em $L(G)$ são adjacentes se as arestas correspondentes em G compartilham um vértice [4]. Esta transformação permite analisar propriedades das arestas do grafo original através do estudo dos vértices do grafo linha.

No exemplo da Figura 3, o grafo linha de K_3 é isomorfo ao próprio K_3, ilustrando como a transformação preserva a estrutura de adjacência. Isso acontece porque, em K_3, todo par de arestas compartilha um vértice. Por exemplo, e_a e e_b são adjacentes no grafo linha porque compartilhavam v_2 no grafo original; o mesmo ocorre para os outros pares. Essa correspondência direta entre incidência e adjacência é o que torna o grafo linha uma ferramenta tão útil em reduções entre problemas de coloração.

No contexto de problemas de coloração, dois se destacam:

ISSN: 2675-3588

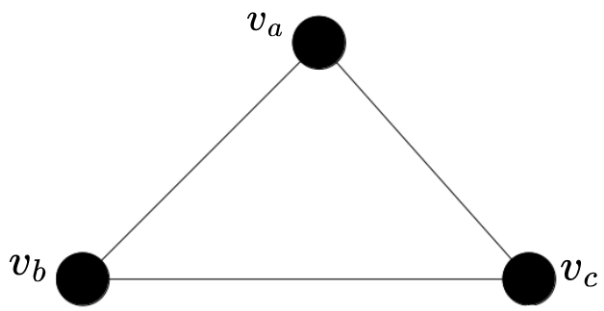


Figura 3: Grafo linha $L(K_3)$ — arestas do grafo original tornam-se vértices, e a incidência transforma-se em adjacência.

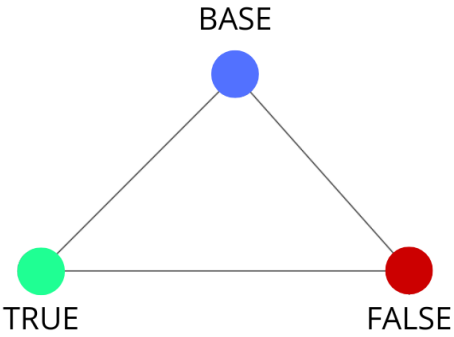


Figura 4: Gadget baseado em K_3, usado para impor restrições de coloração em reduções para 3-Coloring.

o *3-Coloring* (Coloração de Vértices com 3 cores) e o *Edge Coloring* (Coloração de Arestas). O primeiro é um problema de decisão que busca determinar se existe uma função $c : V \to \{1,2,3\}$ tal que, para toda aresta $(u,v) \in E$, tenhamos $c(u) \neq c(v)$ [5, 6]. Uma coloração que satisfaz esta propriedade é chamada de *coloração própria*. A restrição de usar apenas três cores torna este problema particularmente desafiador, já que muitos grafos exigem mais cores para uma coloração própria, enquanto outros podem ser coloridos com menos.

Os *gadgets* desempenham um papel central nas reduções polinomiais envolvendo problemas de coloração. Um gadget é um pequeno subgrafo construído para impor restrições locais sobre as escolhas de cor, funcionando como um "componente lógico" dentro de reduções maiores. Esse conceito aparece na técnica clássica de *Component Design*, frequentemente utilizada em reduções para o problema 3-Coloring.

O *triângulo* K_3 é um dos gadgets mais utilizados, pois sua estrutura força necessariamente três cores distintas, representando escolhas mutuamente exclusivas — como valores lógicos *TRUE*, *FALSE* e um estado base. A Figura 4 ilustra essa construção didática amplamente adotada em reduções clássicas.

Já o *Edge Coloring* (Figura 5) pergunta se é possível colorir as arestas de G usando até k cores, por meio de uma função $c : E \to \{1,2,\ldots,k\}$, de forma que arestas incidentes recebam cores diferentes. O número mínimo de cores necessárias para essa coloração é o *índice cromático*, denotado por $\chi'(G)$. Este problema tem aplicações práticas em escalonamento de tarefas, alocação de frequências em redes wireless e design de torneios esportivos.

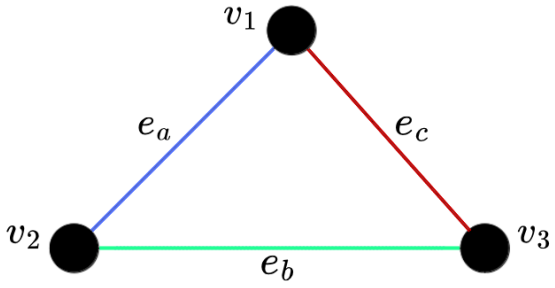


Figura 5: Edge Coloring válido no K_3 — cada aresta recebe uma cor distinta: e_a (azul), e_b (verde), e_c (vermelho).

No âmbito da complexidade computacional, destacamos duas definições centrais. Uma linguagem de decisão L pertence à *classe NP-Completo* se L está em $\mathcal{NP}$ e toda linguagem em $\mathcal{NP}$ pode ser reduzida a L em tempo polinomial [1, 2]. As reduções são formalizadas pelo conceito de *Redução Polinomial*, onde um problema A é redutível a um problema B (denotado por $A \leq_p B$) se existe uma transformação computável em tempo polinomial que preserva respostas entre instâncias dos dois problemas. Esta noção de redução é a base para estabelecer relações de dificuldade entre problemas e para construir hierarquias de complexidade.

A importância das reduções polinomiais vai além do aspecto teórico, pois elas fornecem informações sobre a estrutura dos problemas e permitem que algoritmos desenvolvidos para um problema sejam adaptados para outros. No contexto educacional, compreender essas reduções é essencial para desenvolver uma intuição sobre quais problemas são computacionalmente difíceis e por quê.

III. Trabalhos Relacionados

No trabalho de Cook [2], estabeleceu-se a base da NP-completude. Na pesquisa, mostrou-se como verificar soluções em tempo polinomial e construiu a primeira redução polinomial para o SAT. O objetivo foi entender quando um problema permite conferir respostas em pouco tempo. Os resultados são interessantes por terem aberto caminho para Karp ampliar essa ideia, relacionando vários problemas clássicos, incluindo o 3-Coloring, e mostrando como muitos deles caem na mesma classe de complexidade.

No trabalho clássico de Garey e Johnson [1], é detalhada a complexidade do *Graph 3-Colorability*. Na obra, os autores organizam a teoria da NP-completude e utilizam a técnica de *Component Design*, que consiste em montar grafos a partir de peças pequenas que impõem restrições locais. Gadgets como o triângulo K_3 mostram como estruturas pequenas conseguem impor escolhas de cor e controlar o comportamento local do grafo, simulando literais e cláusulas. Os resultados são interessantes por padronizar as reduções que conectam problemas centrais da computação.

No survey escrito por Cao e outros autores [7], podemos notar uma visão geral desse tema. Na pesquisa, os autores reúnem resultados sobre algoritmos, limites estruturais,

casos especiais e questões abertas da coloração de arestas. O objetivo do survey é organizar o que já se sabe sobre o problema, desde técnicas simples de recoloração até métodos mais atuais. Os resultados são interessantes por consolidar o conhecimento disperso sobre limites superiores e conjecturas da área.

No trabalho de Holyer [3], surge o resultado que estrutura a base moderna dessa área. Na pesquisa, o autor apresenta a primeira prova de NP-completude do *Edge Coloring*, mostrando que o problema permanece intratável mesmo quando restrito a grafos cúbicos. A técnica usada envolve montar blocos que forçam escolhas de cor que se propagam pelo grafo inteiro. Os resultados são interessantes por revelar que o índice cromático capta decisões combinatórias fortes e que o problema não se resume a uma variação simples da coloração de vértices.

No trabalho de Basavaraju e Chandran [8], há um resultado importante para a coloração de arestas acíclica em grafos planares. Os autores demonstram que todo grafo planar admite tal coloração com $\Delta(G)+12$ cores, superando o limite anterior de $2\Delta(G)+29$. A prova utiliza configurações inevitáveis em grafos planares e trocas de cores para evitar ciclos bicromáticos, mostrando como o índice cromático acíclico reflete a estrutura desses grafos.

No estudo de Galby, Lima, Paulusma e Ries [9], trabalhos mais novos reforçam essa visão. Na pesquisa, o objetivo foi classificar o k-Edge Coloring para grafos H-livres, combinando reduções com análise estrutural para mapear quando o problema é polinomial.

No trabalho de Sinnamon [10], são propostos algoritmos para coloração de arestas que buscam ser acessíveis e eficientes. O objetivo da autora é desenvolver métodos simples e rápidos para produzir colorações com $d+1$ cores, onde d é o grau máximo do grafo. A técnica utilizada se baseia em decomposição recursiva e ciclos de Euler para agilizar o processo. Os resultados demonstram que é possível obter boas soluções de forma prática para grafos gerais em aplicações reais.

Por fim, no trabalho de Raeisi e Gholami [11], a coloração de arestas é aplicada à construção de grafos Tanner livres de ciclos curtos para códigos LDPC de peso-coluna três, melhorando a decodificação em canais ruidosos. O método utiliza algoritmos de coloração eficientes para garantir propriedades acíclicas nos grafos bipartidos. Os resultados conectam a combinatória gráfica a aplicações práticas em comunicações digitais.

IV. DESCRIÇÃO DO PROBLEMA

Nesta seção, definimos formalmente os três problemas que aparecem na cadeia de redução. A Tabela 1 apresenta a definição do problema 3-SAT, a Tabela 2 descreve o problema 3-Coloring e a Tabela 3 formaliza o problema Edge Coloring, indicando, em cada caso, a entrada e a pergunta associadas.

TABELA 1: DEFINIÇÃO DO PROBLEMA 3-SAT

3-SAT (Satisfatibilidade Booleana)
Entrada: um conjunto X de variáveis; uma coleção C de cláusulas sobre X onde, para cada $c \in C$, a cláusula possui exatamente 3 literais ($\|c\| = 3$).
Pergunta: Determinar se existe uma atribuição de valores às variáveis em X de modo que cada cláusula em C tenha pelo menos um literal verdadeiro.

Para fins de formalização, define-se um *literal* como uma variável booleana (x) ou sua negação ($\neg x$). Uma *cláusula* é composta pela disjunção lógica (operador OU) desses literais. A especificidade do problema 3-SAT reside na estrutura rígida onde cada cláusula deve conter estritamente três literais, o que permite a padronização dos componentes gráficos (gadgets) utilizados na redução.

A Tabela 1 define o 3-SAT, que serve como o elo de conexão fundamental nesta prova. Diferente dos problemas de coloração, que lidam com estruturas gráficas, o 3-SAT lida com lógica pura. A restrição de ter "exatamente três literais" é o que permite criar padrões geométricos fixos (como triângulos) nas reduções para grafos.

TABELA 2: DEFINIÇÃO DO PROBLEMA 3-COLORING

3-COLORAÇÃO (3-COLORING)
Entrada: Um grafo $G = (V,E)$.
Pergunta: Existe uma função $c : V \to \{1,2,3\}$ tal que vértices adjacentes recebam cores distintas?

A Tabela 2 define formalmente o problema 3-Coloring. A entrada é um grafo qualquer, e a pergunta questiona se é possível colorir seus vértices usando apenas três cores, respeitando a regra básica de que vértices conectados por uma aresta devem ter cores diferentes.

TABELA 3: DEFINIÇÃO DO PROBLEMA EDGE COLORING

COLORAÇÃO DE ARESTAS (EDGE COLORING)
Entrada: Um grafo $G = (V,E)$ e um inteiro k.
Pergunta: Existe uma função $c : E \to \{1,2,\ldots,k\}$ tal que arestas incidentes recebam cores distintas?

A Tabela 3 define o problema Edge Coloring. Aqui, a entrada inclui um grafo e também um número k que representa a quantidade de cores disponíveis. A pergunta busca saber se podemos colorir as arestas do grafo de forma que arestas que compartilham um vértice comum recebam cores diferentes. Note que enquanto no 3-Coloring colorimos

ISSN: 2675-3588

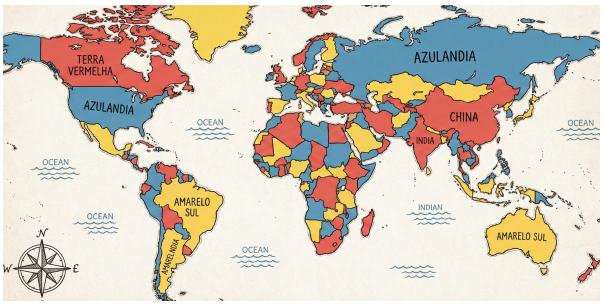


Figura 6: Mapa ilustrando a restrição de adjacência no 3-Coloring.

Figura 7: Ilustração de uma festa como analogia para o 3-SAT.

vértices, no Edge Coloring colorimos arestas, mas ambos impõem restrições de adjacência.

Os problemas de coloração de grafos representam uma classe fundamental na teoria da computação, com aplicações que vão desde o planejamento de horários até a alocação de recursos em sistemas distribuídos. Nesta seção, descrevemos formalmente os dois problemas centrais deste trabalho: o 3-Coloring, um problema clássico de coloração de vértices, e o Edge Coloring, seu análogo na coloração de arestas. Ambos são problemas de decisão NP-completos [1, 2, 3], mas cada um apresenta desafios próprios.

Para tornar esses conceitos mais próximos do cotidiano, podemos imaginar o *3-Coloring* como o ato de colorir um mapa usando apenas três cores, garantindo que países vizinhos nunca compartilhem a mesma cor, como ilustrado na Figura 6.

Em relação ao *3-SAT*, é como organizar uma festa onde o sucesso depende de satisfazer a todos. Para isso, o organizador deve fazer várias escolhas binárias (as variáveis), como definir se "Haverá Música Ao Vivo?" (Sim ou Não). Cada convidado importante impõe uma cláusula: ele só vem à festa se pelo menos uma de suas três condições for atendida. Por exemplo, um convidado pode exigir: "Eu vou se tiver Música Ao Vivo OU se o Amigo X não vier OU se o Buffet for vegetariano." O desafio *3-SAT* é encontrar uma única combinação de decisões (uma configuração Sim/Não para todos os fatores) que satisfaça a exigência de todos os convidados simultaneamente. Se essa combinação existir, a festa pode ser realizada.Essa analogia é ilustrada na Figura 7.

Já o *Edge Coloring* lembra a montagem da grade de horários de uma escola: arestas representam aulas e vértices representam professores ou salas. Aulas que usam o mesmo recurso não podem ocorrer no mesmo horário, e por isso precisam de cores diferentes, conforme ilustrado na Figura 8.

Essas analogias destacam como problemas abstratos da computação surgem em situações reais.

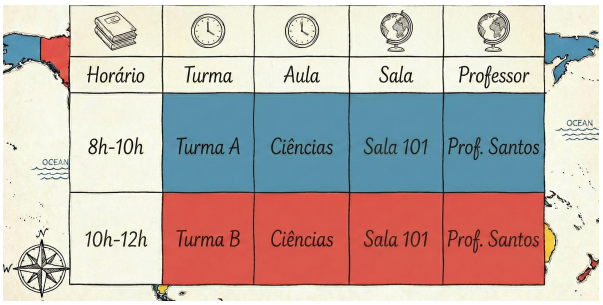


Figura 8: Grade escolar ilustrando o Edge Coloring.

V. Demonstração e Contribuições

Para provar que Edge Coloring é NP-Completo é necessário demonstrar duas condições:

- Edge Coloring $\in$ NP
- Edge Coloring $\in$ NP-Difícil

I) Edge Coloring $\in$ NP: Para demonstrar que o problema pertence a $\mathcal{NP}$, apresentamos um certificado de tamanho polinomial e um algoritmo verificador determinístico capaz de validar esse certificado em tempo polinomial [12].

Dado o seguinte certificado: uma coloração $c : E \to 1,2,\ldots,k$. Um verificador examina todos os pares distintos de arestas do grafo. Para cada par (e_1, e_2), o algoritmo testa se elas compartilham um vértice (são incidentes) e, caso positivo, confirma que $c(e_1) \neq c(e_2)$. Como existem no máximo $|E|^2$ pares de arestas para verificar, o custo computacional é limitado por $O(|E|^2)$, o que garante a execução em tempo polinomial.

II) Equivalência Estrutural (Grafo Linha): Uma forma pedagógica de visualizar a pertinência a $\mathcal{NP}$ é através da transformação para o Grafo Linha $L(G)$. Essa construção demonstra que o problema de Coloração de Arestas pode ser modelado como um problema de Coloração de Vértices (que sabemos estar em $\mathcal{NP}$). A transformação mapeia cada aresta de G em um vértice de $L(G)$, e as adjacências entre arestas incidentes em G tornam-se arestas entre vértices em $L(G)$. Assim, uma coloração própria das arestas de G corresponde diretamente a uma coloração própria dos vértices de $L(G)$. Uma vez que o Grafo Linha pode ser construído em tempo polinomial e a coloração de vértices é um problema bem conhecido em $\mathcal{NP}$, esta equivalência estrutural reforça a classificação do problema de Edge Coloring como pertencente à classe $\mathcal{NP}$.

Dado um grafo $G = (V, E)$, construímos $L(G) = (V_L, E_L)$ onde:

- $V_L = E$ (cada aresta de G torna-se um vértice em $L(G)$)
- $E_L = \{(e_1, e_2) \mid e_1, e_2 \in E \text{ são incidentes em } G\}$

Como exemplo ilustrativo, a Figura 9 mostra um grafo G_{ex} com vértices v_1, v_2, v_3, v_4 e arestas e_a (ligando v_1-v_2), e_b (v_2-v_3), e_c (v_1-v_3) e e_d (v_3-v_4). Observa-se que as arestas e_b e e_d compartilham o vértice v_3, o que configura uma relação de incidência direta. Por sua vez, e_b também é incidente com e_c, uma vez que ambas incidem em v_3. Essas relações de adjacência entre arestas no grafo original serão representadas como arestas no grafo linha $L(G_{ex})$.

A Figura 10 mostra o grafo linha correspondente $L(G_{ex})$, onde cada aresta do grafo original torna-se um vértice. A adjacência entre v_b e v_d reflete diretamente o compartilhamento do vértice v_3 pelas arestas e_b e e_d no grafo original.

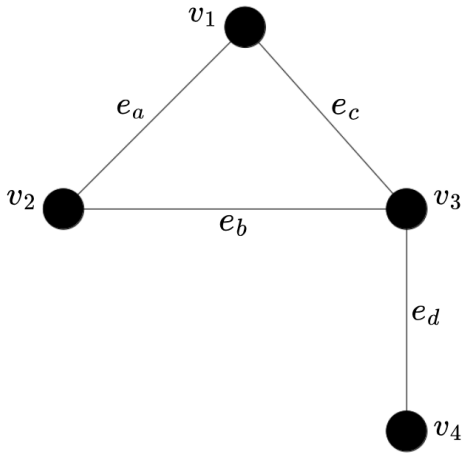


Figura 9: Grafo Base (G_{ex})

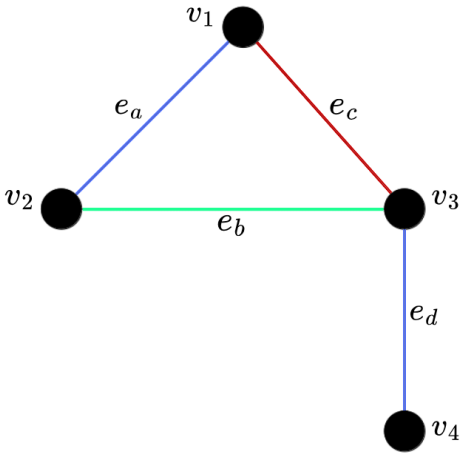


Figura 11: Coloração de arestas em G_{ex}

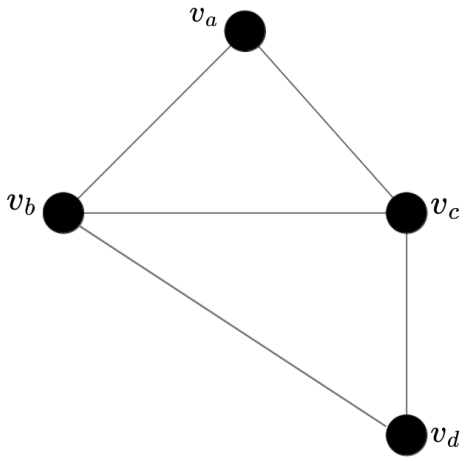


Figura 10: Grafo Linha ($L(G_{ex})$)

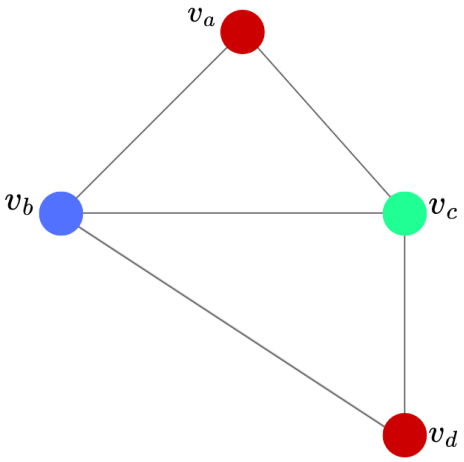


Figura 12: Coloração Válida em $L(G_{ex})$

A propriedade fundamental, conforme Jensen e Toft [13], é que G admite uma k-coloração de arestas se e somente se $L(G)$ admite uma k-coloração de vértices. Esta equivalência confirma que resolver Edge Coloring é redutível a resolver Vertex Coloring, reforçando sua pertinência à classe $\mathcal{NP}$.

a. Prova de Corretude da Equivalência

A seguir, apresenta-se a demonstração de que G é 3-aresta-colorível se e somente se $L(G)$ é 3-vértice-colorível.

Direção 1($\Rightarrow$): Se G é 3-aresta-colorível, então $L(G)$ é 3-vértice-colorível.

Prova: Seja $c : E(G) \rightarrow \{\text{vermelho}, \text{azul}, \text{verde}\}$ uma coloração própria das arestas de G. Para cada aresta $e \in E(G)$, atribuímos a cor $c(e)$ ao vértice correspondente $v_e \in V(L(G))$. Se duas arestas e_b e e_c são incidentes ao mesmo vértice v_3 em G, elas possuem cores diferentes, como ilustrado na Figura 11. Logo, os vértices v_b e v_c em $L(G)$, que são adjacentes, receberão cores diferentes.

Direção 2 ($\Leftarrow$): Se $L(G)$ é 3-vértice-colorível, então G é 3-aresta-colorível.

Prova: Seja $c' : V(L(G)) \rightarrow \{1,2,3\}$ uma coloração própria dos vértices de $L(G)$, como mostrado na Figura 12. Definimos a coloração das arestas de G tal que $c(e) = c'(v_e)$. A preservação da adjacência garante que arestas incidentes em G terão cores distintas, validando a solução.

III) Edge Coloring $\in$ *NP-Difícil:* A demonstração da dificuldade deste problema usa o princípio da transitividade das reduções polinomiais. Para mostrar que Edge Coloring é tão difícil quanto o 3-Coloring, construímos uma cadeia de redução em duas etapas. Primeiro, reduzimos o 3-Coloring para o 3-SAT, transformando restrições de cores em cláusulas booleanas. Depois, reduzimos o 3-SAT para o Edge Coloring, simulando variáveis e cláusulas com um grafo adequado. Pela transitividade, se 3-Coloring é NP-completo, então Edge Coloring é NP-difícil.

A primeira etapa consiste na redução polinomial 3-Coloring $\leq_p$ 3-SAT. Dado um grafo $G = (V,E)$ com n vértices, constrói-se uma fórmula lógica ϕ que codifica de forma precisa as restrições necessárias para uma coloração própria de G com três cores [1]. Para cada vértice $v_i \in V$, são criadas três variáveis booleanas: $x_{i,1}$ (representando a cor 1), $x_{i,2}$ (cor 2) e $x_{i,3}$ (cor 3).

A fórmula ϕ é formada pela conjunção de dois tipos de cláusulas, que juntas modelam as condições de uma coloração própria. A garantia de cor única exige que cada vértice v_i receba pelo menos uma cor, representada pela cláusula $(x_{i,1} \vee x_{i,2} \vee x_{i,3})$. Além disso, a restrição de adjacência assegura que, para cada aresta $(v_i, v_j) \in E$, vértices adjacentes não compartilhem a mesma cor, o que é modelado por três cláusulas de conflito para cada aresta: $(\neg x_{i,1} \vee \neg x_{j,1})$, $(\neg x_{i,2} \vee \neg x_{j,2})$ e $(\neg x_{i,3} \vee \neg x_{j,3})$. Essas cláusulas lógicas correspondem diretamente à restrição de exclusão mútua representada no gadget da Figura 4, demonstrando como condições combinatórias são traduzidas em restrições booleanas.

ISSN: 2675-3588

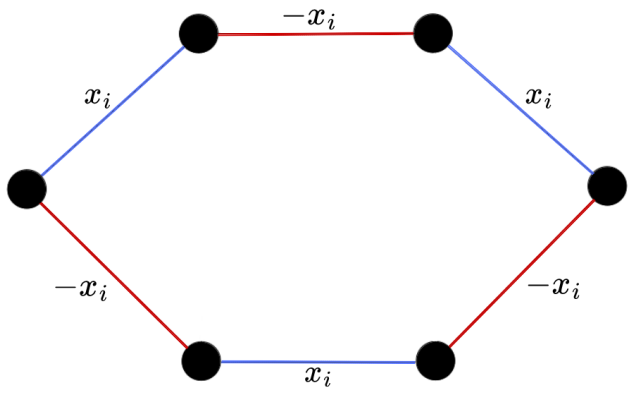


Figura 13: Esquema do Gadget de Variável: a alternância de cores no ciclo simula a negação lógica (x vs $\neg x$).

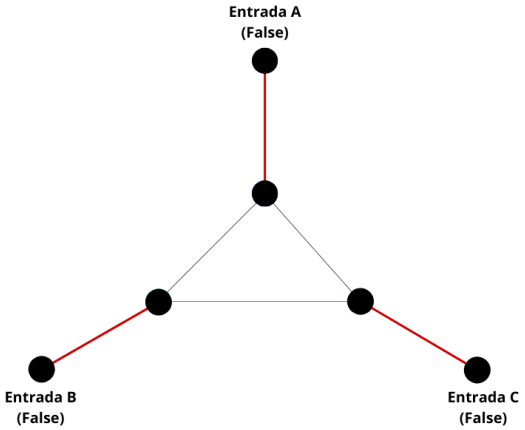


Figura 14: Lógica do Gadget de Cláusula: o componente falha (não é colorível) apenas se receber "Falso" em todas as entradas.

Uma vez obtida a fórmula satisfatível, avançamos para a redução final para Edge Coloring. Esta etapa baseia-se na construção clássica de Ian Holyer[3], que provou ser NP-completo determinar se o índice cromático de um grafo cúbico é 3 ou 4. A redução converte a fórmula lógica em um grafo cúbico utilizando componentes modulares específicos, denominados *gadgets*, que transportam valores de verdade através de pares de arestas coloridas.

A construção utiliza duas estruturas principais que podem ser compreendidas esquematicamente. A primeira é o *Componente de Variável*, que funciona como um gerador de verdade. Conforme ilustrado conceitualmente na Figura 13, ele é constituído por um ciclo de arestas. Devido à natureza da coloração de arestas, as cores devem se alternar obrigatoriamente ao longo do ciclo. Se associarmos uma cor ao valor "Verdadeiro" e outra ao "Falso", essa alternância garante a consistência lógica: sempre que uma aresta representa x, a adjacente representará $\neg x$.

O segundo elemento crítico é o *Componente de Cláusula*, que atua como um testador de satisfação. Este subgrafo conecta-se às arestas dos ciclos das variáveis correspondentes aos literais da cláusula. Sua propriedade topológica fundamental, representada na Figura 14, é o bloqueio condicional: o gadget é desenhado de tal forma que se torna impossível de colorir com apenas 3 cores se, e somente se, todas as suas arestas de entrada carregarem a cor correspondente ao valor "Falso".

Dessa forma, o grafo cúbico resultante completo só será 3-aresta-colorível se existir uma atribuição de verdade que satisfaça a fórmula 3-SAT original, evitando o conflito nos gadgets de cláusula. Conclui-se, assim, que resolver Edge Coloring é suficiente para resolver 3-SAT e, por transitividade, o 3-Coloring.

Ressalta-se que o resultado em um grafo cúbico fortalece a conclusão. Na teoria da complexidade, se um problema é NP-Difícil para uma classe restrita de entradas (grafos 3-regulares), ele mantém essa propriedade para o caso geral (grafos arbitrários), visto que a classe restrita compõe um subconjunto do problema global, conforme o princípio da restrição descrito por Garey e Johnson [1]. Dessa forma, a prova de Holyer fundamenta a classificação do problema de Edge Coloring como NP-Difícil.

Conclusão Geral: Por (I) e (II) demonstramos a pertinência a $\mathcal{NP}$. Por (III) justificamos a dificuldade via redução transitiva baseada em Holyer. Logo, Edge Coloring é NP-Completo.

b. Contribuições Pedagógicas

Este trabalho apresenta contribuições ao ensino de Teoria da Computação ao esclarecer a distinção entre equivalência estrutural e redução de dificuldade. A utilização do Grafo Linha permite demonstrar que o problema de Coloração de Arestas pode ser modelado como um problema de Coloração de Vértices, o que comprova sua pertinência à classe $\mathcal{NP}$ conforme as definições de Garey e Johnson [1]. Em contrapartida, a prova de dificuldade exige a construção de componentes lógicos, ou gadgets, como estabelecido por Holyer [3], evitando a confusão comum sobre a direção das reduções polinomiais.

A visualização da transformação estrutural $G \to L(G)$ [4], complementada por representações visuais, facilita a compreensão geométrica do processo. As ilustrações auxiliam os estudantes a visualizar conceitos abstratos, permitindo o entendimento das relações entre problemas de coloração e a lógica de satisfatibilidade booleana.

VI. Resultados e Reflexões

A análise estrutural via Grafo Linha evidencia como problemas de coloração de naturezas distintas (vértices e arestas) compartilham uma base computacional comum [4]. Esta relação reforça o conceito de que problemas diferentes podem pertencer à mesma classe de complexidade. A equivalência demonstrada confirma que uma instância de coloração de arestas possui solução se, e somente se, a instância correspondente de coloração de vértices no grafo linha também for solucionável.

Uma reflexão central deste estudo recai sobre a complexidade da prova de NP-Dificuldade. Inicialmente, a intuição geométrica sugere uma tentativa de redução direta entre os problemas de coloração. No entanto, a investigação teórica revelou que a redução padrão 3-Coloring $\leq_p$ Edge Coloring não é imediata em termos de construção de gadgets topológicos diretos. Foi necessário compreender que a

literatura estabelece essa conexão através de uma ponte lógica: a redução transitiva passando pelo problema 3-SAT.

Essa descoberta pedagógica é valiosa: ela demonstra que, embora problemas de grafos sejam visualmente similares, a prova de sua dificuldade muitas vezes exige o retorno aos fundamentos da lógica booleana. A construção de Holyer [3], utilizada neste trabalho, ilustra precisamente como restrições locais em um grafo cúbico simulam portas lógicas, confirmando a intratabilidade do problema mesmo em estruturas restritas.

Sob uma perspectiva prática, a confirmação da NP-completude do Edge Coloring para $k = 3$ indica a necessidade de abordagens alternativas para a solução exata em casos gerais [3]. Essa constatação direciona a investigação para o uso de heurísticas e a análise de casos especiais tratáveis, como os grafos bipartidos, onde o Teorema de Vizing assegura que o índice cromático iguala o grau máximo [14]. Para o ensino, este resultado demonstra que a classificação de complexidade orienta a escolha de estratégias algorítmicas.

A contribuição pedagógica deste trabalho reside na integração de técnicas de redução com suporte visual. A construção do Grafo Linha e a explicação dos componentes de Holyer tornam a demonstração acessível para estudantes de graduação. A apresentação sequencial dos conceitos permite a compreensão dos passos necessários para estabelecer a NP-Completude de um problema, desde a verificação via equivalência estrutural até a prova de dificuldade via satisfatibilidade lógica.

Os resultados destacam o Grafo Linha como ferramenta pedagógica na teoria da computação [15]. Esta estrutura facilita a compreensão das reduções polinomiais e serve como ponte conceitual entre diferentes áreas da teoria dos grafos. A metodologia adotada pode ser aplicada ao ensino de outros tópicos, combinando formalismo matemático com exemplos visuais para tornar conceitos abstratos tangíveis.

VII. Considerações Finais

Este artigo estabeleceu a classificação do problema Edge Coloring como NP-Completo mediante uma abordagem dupla. A pertinência a $\mathcal{NP}$ foi demonstrada através da equivalência estrutural com o problema de Coloração de Vértices via Grafo Linha, conforme teoria de Whitney [4]. A condição de NP-Dificuldade foi justificada pela redução polinomial a partir do problema 3-SAT, utilizando a construção de gadgets proposta por Holyer [3], o que valida a relação transitiva com o problema 3-Coloring.

As contribuições pedagógicas compreendem a formalização da prova de redutibilidade e a distinção metodológica entre verificação e prova de dificuldade. A incorporação de exemplos ilustrativos e a discussão sobre as implicações práticas da intratabilidade computacional visam apoiar o aprendizado. O material pode integrar cursos de teoria da computação como exemplo de técnicas de redução e análise de complexidade.

O trabalho demonstra a viabilidade de apresentar conceitos de teoria da computação de maneira compreensível para estudantes de graduação. A abordagem baseada em exemplos visuais constrói a intuição sobre reduções polinomiais e NP-completude [1].

Para o ensino de complexidade computacional, este recurso combina rigor teórico com acessibilidade. A estrutura apresentada permite acompanhar o processo de redução polinomial, desde a transformação inicial até a prova de corretude, o que desenvolve a compreensão dos fundamentos da teoria da NP-completude [2].

Trabalhos futuros podem investigar variantes do problema, como Edge Coloring em classes restritas de grafos ou desenvolver materiais interativos para visualização de reduções. A criação de recursos adicionais como vídeos explicativos ou simulações poderia complementar o material apresentado, ampliando o impacto educacional na área de complexidade computacional.

References

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

[2] S. A. Cook, "The complexity of theorem-proving procedures," in *Proceedings of the Third Annual ACM Symposium on Theory of Computing*. ACM, 1971, pp. 151–158.

[3] I. Holyer, "The np-completeness of edge-coloring," *SIAM Journal on Computing*, vol. 10, no. 4, pp. 718–720, 1981.

[4] H. Whitney, "Congruent graphs and the connectivity of graphs," *American Journal of Mathematics*, vol. 54, no. 1, pp. 150–168, 1932.

[5] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*. New York: North Holland, 1976.

[6] C. H. Papadimitriou, *Computational Complexity*. Reading, Massachusetts: Addison-Wesley, 1994.

[7] K. Cao, M. Wang, and W. Zhang, "Graph edge coloring: A survey," *arXiv*, 2018, arXiv:1810.08027.

[8] M. Basavaraju and L. S. Chandran, "Acyclic edge-coloring of planar graphs," *SIAM Journal on Discrete Mathematics*, vol. 23, no. 3, pp. 1618–1627, 2009.

[9] E. Galby, C. Lima, D. Paulusma, and B. Ries, "Classifying k-edge-colouring for h-free graphs," *arXiv*, 2019, arXiv:1907.03201.

[10] L. Sinnamon, "Fast and simple edge-coloring algorithms," *arXiv*, 2021, arXiv:2103.01311.

[11] G. R. Raeisi and M. R. Gholami, "Edge coloring of graphs with applications in coding theory," *China Communications*, vol. 18, no. 1, pp. 181–195, 2021.

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA: MIT Press, 2009.

[13] T. R. Jensen and B. Toft, *Graph Coloring Problems*. New York: Wiley-Interscience, 1995.

 ISSN: 2675-3588

[14] V. G. Vizing, “On an estimate of the chromatic class of a p-graph,” *Diskret. Analiz*, vol. 3, pp. 25–30, 1964, (em Russo).

[15] P. G. H. Lehot, “An optimal algorithm to detect a line graph and output its root graph,” *Journal of the ACM*, vol. 21, no. 4, pp. 569–575, 1974.

 ISSN: 2675-3588

AJC
Academic Journal on Computing, Engineering and Applied Mathematics
EAM

2026

Volume 7 Issue 2

Support

UNIVERSIDADE FEDERAL DO TOCANTINS

ISSN: 2675-3588