Redução de 3-SAT para Subconjunto Independente em Grafos: Implicações e Perspectivas

Vitor José Ferreira dos S. de Santana[1], **Ericksulino M. de A. Moura**[1], **José Miqueias de A. Pereira** [1], **Juliana Oliveira de Carvalho** [1] , **Glauber Dias Gonçalves**[1]

[1]Universidade Federal do Piauí (UFPI)
Picos – PI – Brazi

vitorsantos,ericksulino,jmiqueias,julianaoc,ggoncalves}@ufpi.edu.br

Abstract. *This study investigates the reduction of the 3-SAT problem to Independent Subset in graphs, preserving the NP-complete complexity of the original problems. The methodology uses the 3-SAT mathematical structure to translate Boolean clauses into graphical relationships, enabling the verification of independent sets. The results show that this approach can offer efficient and scalable solutions, useful in consistency checks in specialized systems, optimization of digital circuits and other complex computational challenges.*

Resumo. *Este estudo investiga a redução do problema 3-SAT para Subconjunto Independente em grafos, preservando a complexidade NP-completa dos problemas originais. A metodologia utiliza a estrutura matemática do 3-SAT para traduzir cláusulas booleanas em relações gráficas, viabilizando a verificação de conjuntos independentes. Os resultados mostram que esta abordagem pode oferecer soluções eficientes e escaláveis, úteis em verificações de consistência em sistemas especialistas, otimização de circuitos digitais e outros desafios computacionais complexos.*

1. Introdução

A redução de problemas e a complexidade de *software* desempenham papéis fundamentais na resolução de desafios computacionais no mundo contemporâneo. Para estudar a eficiência computacional de problemas computacionais quaisquer existe a área da Complexidade Computacional [Papadimitriou 1994]. Em um contexto onde a tecnologia permeia todos os aspectos da vida, desde a comunicação até a gestão de recursos críticos, a eficiência e a confiabilidade dos sistemas são cruciais. A complexidade excessiva pode resultar em sistemas difíceis de entender, manter e escalar, aumentando significativamente os custos e o risco de falhas. Portanto, a simplificação não apenas melhora a qualidade do *software*, mas também reduz os potenciais pontos de falha, promovendo maior estabilidade. Complexidade significa o nível de dificuldade em compreender e verificar um *software* programa ou um determinado componente de *software* [Committee et al. 1990].

Na prática, a redução de problemas envolve identificar e eliminar redundâncias, simplificar algoritmos e arquiteturas, e adotar abordagens mais elegantes e eficientes. Os benefícios da teoria da redução de problemas na complexidade do algoritmo incluem menor custo computacional, maior eficiência de pesquisa e abordagem da sobrecarga cognitiva para os tomadores de decisão [Sinha et al. 2013]. A redução objetiva na teoria da

complexidade do algoritmo pode tornar problemas com muitos objetivos solucionáveis, removendo objetivos não essenciais, melhorando a eficiência da pesquisa, o custo computacional e a tomada de decisões [Saxena et al. 2013].

O problema de Satisfatibilidade de ordem 3 (3-SAT) é um problema bem conhecido como NP-completo e possui muitas aplicações práticas, como verificação de consistência em bases de conhecimento de sistemas especialistas e síntese de circuitos assíncronos [Fallah et al. 1998]. A verificação de consistência é crucial para garantir que não existam conflitos entre as regras e fatos armazenados em uma base de conhecimento. A complexidade deste problema pode ser abordada de forma mais eficiente ao transformá-lo em um problema de Subconjunto Independente, facilitando a aplicação de algoritmos gráficos para encontrar soluções consistentes.

Além disso, o 3-SAT é amplamente resolvido pelo mapeamento do algoritmo de Davis-Putnam-Logemann-Loveland (DPLL) [Cané et al. 2023]. Outros casos de utilização incluem a Inteligência Artifical por meio da rede de Hopfield [Hopfield and Tank 1989], automação de projeto eletrônico [Pan and Chu 2023] e computação molecular, como demonstrado por Lipton [Lipton 1995] para aproveitar o paralelismo proporcionado pela computação de DNA [Braich et al. 2002]. Este trabalho examina a viabilidade de solução do problema 3-SAT por meio da redução para o problema de Subconjunto Independente, demonstrando como essa transformação preserva a complexidade original e pode ser uma alternativa de solução de problemas comuns do 3-SAT.

O artigo é estruturado em cinco seções, além da introdução, divididos da seguinte forma. A Seção 2 discute conceitos chave para a compreenção deste trabalho.A Seção 3 descreve os procedimentos e técnicas utilizadas na pesquisa. Na Seção 4, são apresentados os dados obtidos e suas análises. E por fim, a Seção 5 resume as principais descobertas e implicações do estudo, sugerindo possíveis direções para futuras pesquisas.

2. Fundamentação Teórica

2.1. Redução entre problemas

A redução entre problemas é uma técnica fundamental no projeto de algoritmos, permitindo a transformação de um problema em outro, de modo que uma solução eficiente para o problema transformado possa ser utilizada para resolver o problema original. A ideia intuitiva é utilizar um algoritmo que já existe para um certo problema, ou qualquer algoritmo que venha a ser criado para ele, para resolver outro problema [Lintzmayer and Mota 2020]. A principal técnica usada para demonstrar que dois problemas são relacionado é o de "reduzir"um ao outro, dando uma transformação que mapeia qualquer instância do primeiro problema em um instância equivalente do segundo [Garey and Johnson 1979].

Do ponto de vista matemático, a redução entre problemas é formalizada pela existência de uma função computável f que mapeia instâncias sim de A para instâncias sim de B e, por consequência, instâncias não de A para instâncias não de B. Fazer reduções com essa garantia nos permite usar um algoritmo para o problema B sobre $f(I_A)$ de tal forma que se tal algoritmo responder sim, teremos certeza de que I_A é sim, e se ele responder não, teremos certeza de que I_A é não [Lintzmayer and Mota 2020]. A Figura 1 ilustra o fluxograma de solução de um problema A por meio de um problema B.

Figura 1. Fluxograma de ilustração de solução de um problema A por meio de redução para um problema B. Fonte: [Lintzmayer and Mota 2020].

2.2. Classes de complexidade

A Complexidade Computacional caracteriza a complexidade de um problema a partir da quantidade de recursos computacionais, como espaço e tempo, para resolvê-lo [Moreira 2016]. Na complexidade computacional classificamos os problemas computacionais em classes de complexidade [Papadimitriou 2003]. Essas classes nos ajudam a entender quais problemas podem ser resolvidos de forma eficiente e quais são intrinsecamente difíceis, requerendo mais recursos computacionais conforme aumenta a complexidade dos problemas. Uma classe de complexidade é especificada pelo modelo de computação e representa um conjunto de problemas relacionados aos recursos computacionais baseados em complexidade na Teoria da Complexidade Computacional [Rocha 2014].

Conforme definido por Cook [Cook 2021], P é a classe dos conjuntos reconhecíveis em tempo polinomial, significando que existe um algoritmo que resolve o problema de entrada em um tempo que é uma função polinomial do tamanho da entrada [Sipser 1996]. Informalmente, P pode ser definida como a classe dos problemas de reconhecimento resolvíveis por um algoritmo de tempo polinomial [Papadimitriou and Steiglitz 2013].

NP é a classe das linguagens decididas por máquinas de Turing não determinísticas em tempo polinomial [Papadimitriou 2003], contendo todos os problemas de decisão para os quais uma solução pode ser verificada por uma máquina de Turing determinística em tempo polinomial [Lucchesi 1979]. A classe NP-completo é um subconjunto de NP, sendo os problemas mais difíceis dentro dessa classe. Um problema é NP-completo se todos os outros problemas em NP se reduzem a ele em tempo polinomial [Papadimitriou 2003]. Se um problema NP-completo puder ser resolvido em tempo polinomial, então todos os problemas em NP também poderiam ser [Sipser 1996].

A classe NP-difícil inclui todos os problemas que são, pelo menos, tão difíceis quanto os problemas mais difíceis em NP, mas não precisam necessariamente estar em NP [Sipser 1996]. Milhares de problemas são conhecidos por serem NP-difícil, e para determinar se um problema é NP-difícil, deve-se verificar se alguém já provou sua dificuldade ou usou um problema NP-difícil conhecido para isso [Tovey 2002].

2.3. Satisfabilidade de ordem 3 (3-SAT)

O problema SAT é um protótipo de problema NP-completo de fundamental importância na teoria da complexidade computacional, com muitas aplicações em ciência e engenharia [Zhang et al. 2024]. Hipoteticamente, o problema de três satisfatibilidades (3-SAT) pode ser classificado como um problema NP-Hard clássico [Choi 2011]. O problema de satisfatibilidade considerará como uma tarefa de busca de uma atribuição de verdade que satisfaça a expressão lógica como verdadeira [Mansor and Sathasivam 2016]. Geralmente

SAT é uma lógica booleana composta por três literais que permitem escolhas de valores para cada literal[Mansor et al. 2018].

Este problema tem sido há muito tempo um cenário combinatorial padrão para estudos algorítmicos e teóricos. A densidade d é um indicador crucial que caracteriza o espaço de soluções e sua dificuldade computacional. Para uma instância 3-SAT com d pequeno, o problema é subdeterminado e provavelmente satisfazível, enquanto que com d grande, ele é superdeterminado e provavelmente insatisfazível [Zhang et al. 2024]. Dada uma instância ϕ de 3-SAT em forma normal conjuntiva (CNF), o problema é dado pela Fórmula 1. De modo que cada l_{ij} representa um literal booleano. A saída do problema é determinar se existe uma atribuição de valores booleanos para as variáveis de modo a fórmula seja satisfeita. Logo, há uma atribuição de valores booelanos que deve tornar a formula ϕ verdadeira.

$$\phi = (l_{11} \vee l_{12} \vee l_{13}) \wedge (l_{21} \vee l_{22} \vee l_{23}) \wedge \ldots \wedge (l_{m1} \vee l_{m2} \vee l_{m3}) \tag{1}$$

2.4. Subconjunto Independente

Um Subconjunto Independente (SI) de um grafo é um conjunto de vértices tal que nenhum par de vértices do conjunto é adjacente, ou seja, não há aresta conectando qualquer par de vértices no subconjunto [Garey and Johnson 1979]. O Subconjunto Independente é o problema no qual, dado um grafo G e um inteiro k, o objetivo é determinar se existe um conjunto independente de G com tamanho pelo menos k [Iwata and Yoshida 2015].

O problema do Subconjunto Independente em grafos é um problema fundamental em teoria dos grafos, onde o objetivo é encontrar o maior conjunto de vértices em um grafo tal que nenhum par de vértices esteja conectado por uma aresta. Um conjunto [1, k] independente em um grafo é um subconjunto onde cada vértice v V S é adjacente a pelo menos um, mas não mais que k vértices em S, e o problema de decisão relacionado a este conjunto é NP-completo [Chellali et al. 2014].

Formalmente, dado um grafo $G = (V, E)$, um subconjunto $S \subseteq V$ é considerado independente se para todo par de vértices $u, v \in S$, não existe uma aresta $(u, v) \in E$. O problema busca encontrar S maximizando $|S|$, o tamanho do conjunto independente. Um conjunto independente de um grafo G = (V, E) é um conjunto $S \subseteq V$ tal que G[S] não tem arestas [Iwata and Yoshida 2015].

3. Trabalhos Relacionados

Nesta seção, revisamos estudos relevantes que exploram a aplicação de algoritmos e teorias de redução em problemas NP-completos, com um foco especial em 3SAT e problemas de grafos. Embora tenhamos procurado por trabalhos relacionados ao problema do Subconjunto Independente, não encontramos material significativo. Por isso, concentramos nossa análise em pesquisas que abordam problemas de grafos, onde a literatura existente oferece diversas abordagens para a compreensão e redução desses problemas complexos.

O artigo de [Gibney et al. 2020] foca em reduções rígidas do problema de satisfatibilidade de fórmulas booleanas gerais (Fórmula-SAT) para dois novos problemas: correspondência de padrões em gráficos rotulados (PMLG) e isomorfismo de subárvore. Destaca-se que as reduções da Fórmula-SAT são mais plausíveis e trazem consequências

até para melhorias logarítmicas nos limites superiores de um problema, ao contrário das reduções usuais da CNF-SAT. Além disso, o artigo utiliza técnicas semelhantes às usadas em reduções anteriores, mas aplicadas em um contexto mais simples, facilitando a compreensão das técnicas envolvidas.

O artigo de [Kusper et al. 2020] foca na comparação de diferentes modelos de conversão de problemas 2-SAT e 3-SAT em grafos direcionados, com destaque para o Modelo Balatonboglár e sua versão simplificada. O estudo inclui a apresentação de exemplos e resultados de benchmarks utilizando o CSFLOC, analisando como o tamanho dos arquivos e o número de cláusulas não utilizadas variam com a densidade do grafo representado.

O artigo de [Karve and Hirani 2021] tem como objetivo explorar a interseção entre a satisfatibilidade e a teoria dos grafos por meio do problema de decisão GraphSAT. A metodologia empregada envolve uma análise aprofundada das relações entre esses dois campos, buscando estabelecer conexões significativas. Os autores apresentam informações sobre como a satisfatibilidade e a teoria dos grafos se relacionam, culminando em um resultado que contribui para a compreensão e avanço dessas áreas de estudo.

O artigo de [Marchetti and Bodily 2022] apresenta uma ferramenta visual para ajudar os alunos a entender o processo de redução entre 3SAT e CLIQUE, abordando os desafios no ensino de conceitos teóricos de ciência da computação.A ferramenta de visualização serve como um recurso prático para educadores para facilitar o aprendizado e a compreensão de tópicos teóricos complexos, como NP-completude e teoria da redução. As conclusões ressaltam a importância de ferramentas interativas em ambientes educacionais para tornar os conceitos abstratos mais acessíveis e melhorar a compreensão e retenção dos alunos dos princípios teóricos da Ciência da Computação.

Tabela 1. Comparação de Trabalhos Relacionados

Trabalho	**I**	**II**	**III**	**IV**
[Gibney et al. 2020]	Sim	Sim	Não	Sim
[Kusper et al. 2020]	Não	Sim	Não	Sim
[Karve and Hirani 2021]	Sim	Sim	Não	Sim
[Marchetti and Bodily 2022]	Sim	Sim	Não	Sim
Este artigo	**Sim**	**Sim**	**Sim**	**Sim**

I - Faz redução entre problemas; II - 3-SAT; III - Subconjunto Independente; e IV - Grafos;

A Tabela 1 compara este trabalho com outras pesquisas, destacando sua dimensão ao cobrir todas as dimensões consideradas: I-Redução entre problemas, II-3-SAT, III-Subconjunto Independente e IV-Grafos. Este trabalho se diferencia dos estudos [Gibney et al. 2020], [Karve and Hirani 2021] e [Marchetti and Bodily 2022], que não abordam o subconjunto independente, e de [Kusper et al. 2020], que não inclui a redução entre problemas. Ao oferecer uma cobertura completa, este artigo apresenta uma solução mais robusta e integrada, potencialmente promovendo avanços na teoria dos algoritmos e suas aplicações práticas.

4. Metodologia

Esta seção apresenta a metodologia deste trabalho que consiste na análise de algoritmos e em investigar a redução de problemas NP-completos. O principal objetivo deste trabalho é analisar a viabilidade de solução do problema 3-SAT por redução para o Subconjunto Independente de modo que possam ser aplicados no contexto de utilização do mundo real. A Figura 2 apresenta a visão geral da metodologia utilizada neste trabalho.

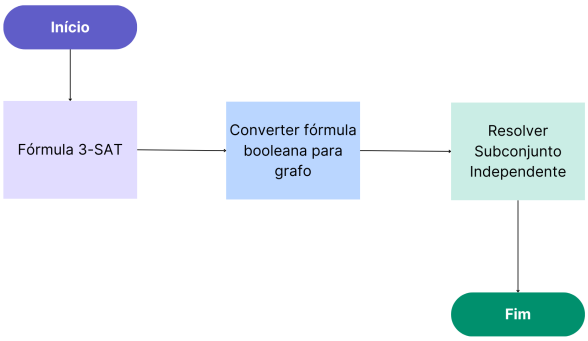

Figura 2. Fluxograma da metodologia aplicada a este trabalho.

O Algoritmo 1 converte uma fórmula 3-SAT em um Grafo $G(V, A)$, de modo a permitir a visualização e manipulação das relações entre variáveis e cláusulas de forma gráfica. Inicialmente, o conjunto de vértices V e o conjunto de arestas A são iniciados como vazios. Para cada cláusula na fórmula, são extraídos os literais e adicionados ao conjunto de vértices, caso ainda não estejam presentes (Linha 6 a 10). Em seguida, arestas são adicionadas entre os vértices correspondentes aos literais, estabelecendo conexões que refletem a estrutura da cláusula (Linha 11 a 15). Adicionalmente, para cada variável da fórmula, são criadas arestas de incompatibilidade entre os vértices que representam a variável e sua negação, garantindo que estas relações sejam mantidas no grafo resultante. Ao final, o algoritmo retorna o grafo $G(V, A)$, que encapsula a estrutura lógica da fórmula 3-SAT em um formato adequado para análise gráfica.

O Algoritmo 2 visa determinar a existência de um Subconjunto Independente de tamanho k em um grafo $G(V, A)$. Inicialmente, o algoritmo itera sobre todos os subconjuntos de vértices de tamanho k, verificando se cada subconjunto satisfaz a condição de independência (Linha 2). Para isso, verifica-se se há alguma aresta conectando quaisquer dois vértices dentro do subconjunto (Linha 5 a 8). Caso nenhum par de vértices esteja conectado por uma aresta, o subconjunto é considerado independente. Se um conjunto independente de tamanho k é encontrado durante a iteração, o algoritmo retorna **TRUE** (Linha 11); caso contrário, retorna **FALSE** (Linha 14), indicando a não existência de tal Subconjunto no Grafo.

A combinação dos algoritmos apresentados permite resolver eficientemente o problema 3-SAT quando há uma correspondência com o problema do subconjunto independente em grafos. No contexto do problema 3-SAT, uma fórmula é considerada satisfatível se existe uma atribuição de verdade às variáveis que a torna verdadeira. A estratégia de conversão da fórmula 3-SAT em um grafo, conforme descrito no primeiro algoritmo, transforma o problema em uma representação gráfica onde as cláusulas são mapeadas como conexões entre vértices.

Algorithm 1 Transformação de uma fórmula 3-SAT em um grafo

```
1: function SAT3PARAGRAFO(F)
2:     Entrada: Fórmula 3-SAT F
3:     Saída: Grafo G(V, A)
4:     G ← Graph()
5:     for each clause ∈ F do
6:         for each literal ∈ clause do
7:             if literal ∉ G then
8:                 G.add_node(literal)
9:             end if
10:        end for
11:        for i ← 0 to len(clause) − 1 do
12:            for j ← i + 1 to len(clause) do
13:                G.add_edge(clause[i], clause[j])
14:            end for
15:        end for
16:    end for
17:    return G
18: end function
```

Algorithm 2 Verificar Subconjunto Independente de Tamanho k

Entrada: Grafo $G(V, A)$, inteiro k
Saída: TRUE se existe um SI de tamanho k, FALSE caso contrário

```
1: function VERIFICARSUBCONJUNTOINDEPENDENTE(G, k)
2:     for cada subconjunto S ⊆ V com |S| = k do
3:         independente ← TRUE
4:         for cada par de vértices (u, v) ∈ S × S do
5:             if (u, v) ∈ A then                ▷ Se existe uma aresta entre u e v
6:                 independente ← FALSE
7:                 break
8:             end if
9:         end for
10:        if independente then
11:            return TRUE
12:        end if
13:    end for
14:    return FALSE
15: end function=0
```

Em seguida, o segundo algoritmo verifica se existe um conjunto independente de tamanho k nesse grafo. Quando aplicado em conjunto, se o grafo resultante possui um conjunto independente de tamanho igual ao número de cláusulas na fórmula 3-SAT, isso implica na existência de uma atribuição de verdade que satisfaz a fórmula original. Portanto, essa abordagem combinada oferece uma solução eficaz e escalável para o problema 3-SAT, explorando as propriedades estruturais dos grafos para determinar sua satisfatibilidade de maneira computacionalmente viável.

5. Resultados

Nesta seção, são apresentados os resultados obtidos a partir da redução do problema 3-SAT para o Subconjunto Independente. Com base na metodologia adotada neste estudo foi demonstrado que é possível realizar a redução entre os dois problemas de forma que a complexidade computacional do 3-SAT, sendo NP-completo, é preservada no problema de Subconjunto Independente.

Os algoritmos desenvolvidos para esta redução, como exemplificado pelo Algoritmo 1 que converte uma fórmula 3-SAT em um Grafo, e o Algoritmo 2 que verifica a existência de um conjunto independente de tamanho k, foram fundamentais para demonstrar a aplicabilidade dessa metodologia. A transformação da estrutura lógica das cláusulas 3-SAT para representações em Grafos, seguida pela verificação estrutural de conjuntos independentes no grafo resultante, demonstra uma abordagem que pode resolver problemas computacionais como a verificação de consistência em bases de conhecimento de sistemas especialistas e síntese de circuitos assíncronos resultando em respostas não contraditórias e confiáveis.

6. Conclusão

Este estudo demonstrou que a redução do problema 3-SAT para o problema de Subconjunto Independente em grafos é uma estratégia eficaz para preservar a complexidade computacional dos problemas originais. A metodologia utilizada explorou a estrutura matemática subjacente do 3-SAT, traduzindo-o em termos de grafos e permitindo a aplicação de técnicas bem estabelecidas para verificação de conjuntos independentes. Os resultados obtidos destacam a viabilidade dessa abordagem em contextos onde a resolução exata de problemas NP-completos como o 3-SAT é impraticável, mas soluções aproximadas são aceitáveis e eficientes. Além disso, este estudo sublinha a relevância dessa metodologia em aplicações práticas, como verificação de consistência em bases de conhecimento de sistemas especialistas e síntese de circuitos assíncronos.

Para futuras investigações, recomenda-se explorar a extensão e aplicação desta metodologia em diferentes classes de problemas NP-completos além do 3-SAT. Investigar a adaptação de outras reduções para problemas relacionados, como o Problema do Clique em grafos, pode proporcionar uma análise aprofundada da eficácia dessas abordagens em diversos contextos computacionais. Além disso, estudos empíricos para avaliar o desempenho dessas técnicas em larga escala e em ambientes de computação real poderiam fornecer validações práticas importantes para sua aplicação. Adicionalmente, explorar variantes heurísticas e algoritmos de aproximação específicos para problemas derivados dessa metodologia poderia ampliar ainda mais as possibilidades de aplicação prática dessas técnicas em problemas computacionais complexos.

Referências

Braich, R. S., Chelyapov, N., Johnson, C. R., Rothemund, P., and Adleman, L. (2002). Solution of a 20-variable 3-sat problem on a dna computer. *Science*, 296:499 – 502.

Cané, M., Coll, J., Rojo, M., and Villaret, M. (2023). Sat-it: the interactive sat tracer. In *Artificial Intelligence Research and Development*, pages 337–346. IOS Press.

Chellali, M., Favaron, O., Haynes, T., Hedetniemi, S., and McRae, A. A. (2014). Independent [1, k]-sets in graphs. *Australas. J Comb.*, 59:144–156.

Choi, V. (2011). Quantum information & computation. 11(7-8):638–648.

Committee, I. S. et al. (1990). Ieee standard glossary of software engineering terminology. *IEEE Std*, 610:12.

Cook, S. A. (2021). The complexity of theorem-proving procedures (1971).

Fallah, F., Devadas, S., and Keutzer, K. (1998). Functional vector generation for hdl models using linear programming and 3-satisfiability. In *Proceedings 1998 Design and Automation Conference. 35th DAC. (Cat. No.98CH36175)*, pages 528–533.

Garey, M. R. and Johnson, D. S. (1979). *Computers and intractability*, volume 174. freeman San Francisco.

Gibney, D., Hoppenworth, G., and Thankachan, S. V. (2020). Simple reductions from formula-sat to pattern matching on labeled graphs and subtree isomorphism. pages 232–242.

Hopfield, J. and Tank, D. (1989). Neural architecture and biophysics for sequence recognition. In *Neural models of plasticity*, pages 363–377. Elsevier.

Iwata, Y. and Yoshida, Y. (2015). On the equivalence among problems of bounded width. pages 754–765.

Karve, V. and Hirani, A. N. (2021). Graphsat - a decision problem connecting satisfiability and graph theory. *ArXiv*, abs/2105.11390.

Kusper, G., Biró, C., and Balla, T. (2020). Investigation of the efficiency of conversion of directed graphs to 3-sat problems. *2020 IEEE 14th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, pages 000227–000234.

Lintzmayer, C. N. and Mota, G. O. (2020). Análise de algoritmos e estruturas de dados. *Centro de Matemática, Computação e Cognição*.

Lipton, R. J. (1995). Dna solution of hard computational problems. *science*, 268(5210):542–545.

Lucchesi, C. L. (1979). *Aspectos teóricos da computação*, volume 8. Instituto de Matemática Pura e Aplicada, CNPq.

Mansor, M. A. and Sathasivam, S. (2016). Accelerating activation function for 3-satisfiability logic programming. *International Journal of Intelligent Systems and Applications*, 8(10):44.

Mansor, M. A., Sathasivam, S., and Kasihmuddin, M. S. M. (2018). 3-satisfiability logic programming approach for cardiovascular diseases diagnosis. In *AIP Conference Proceedings*, volume 1974. AIP Publishing.

Marchetti, K. and Bodily, P. (2022). Visualizing the 3sat to clique reduction process. *2022 Intermountain Engineering, Technology and Computing (IETC)*, pages 1–5.

Moreira, N. (2016). Classes de complexidade.

Pan, H.-Y. and Chu, Z.-F. (2023). A semi-tensor product based all solutions boolean satisfiability solver. *Journal of Computer Science and Technology*, 38(3):702–713.

Papadimitriou, C. H. (1994). *Computational Complexity*. Addison-Wesley, Reading, Massachusetts.

Papadimitriou, C. H. (2003). Computational complexity. In *Encyclopedia of computer science*, pages 260–265.

Papadimitriou, C. H. and Steiglitz, K. (2013). *Combinatorial optimization: algorithms and complexity*. Courier Corporation.

Rocha, T. A. (2014). Complexidade descritiva de classes de complexidade probabilísticas de tempo polinomial e das classes p e np conp através de lógicas com quantificadores de segunda ordem.

Saxena, D., Duro, J. A., Tiwari, A., Deb, K., and Zhang, Q. (2013). Objective reduction in many-objective optimization: Linear and nonlinear algorithms. *IEEE Transactions on Evolutionary Computation*, 17:77–99.

Sinha, A., Saxena, D., Deb, K., and Tiwari, A. (2013). Using objective reduction and interactive procedure to handle many-objective optimization problems. *Appl. Soft Comput.*, 13:415–427.

Sipser, M. (1996). Introduction to the theory of computation. *ACM Sigact News*, 27(1):27–29.

Tovey, C. A. (2002). Tutorial on computational complexity. *Interfaces*, 32(3):30–61.

Zhang, Z., Paredes, R., Sundar, B., Quiroga, D., Kyrillidis, A., Duenas-Osorio, L., Pagano, G., and Hazzard, K. R. (2024). Grover-qaoa for 3-sat: Quadratic speedup, fair-sampling, and parameter clustering. *arXiv preprint arXiv:2402.02585*.