Estudo de Coloração de Grafos Aplicado ao Problema de Alocação de Horários: Uma Solução Focada no Aluno

François Barbosa[1], **José Jeovane**[1], **Guilherme Avelino**[1] **e Arlino Magalhães**[1]

[1] Universidade Federal do Piauí (UFPI)
Campus Universitário Ministro Petrônio Portella - Bairro Ininga.
CEP: 64.049-550 - Teresina - PI.

francois.barbosa@ifma.edu.br, jose.cordeiro@ifma.edu.br

gaa@ufpi.edu.br, arlino@ufpi.edu.br

Abstract. *The research explores the application of Graph Theory to the problem of timetable allocation, focusing on students in their final year. Using graph coloring, subjects were represented as vertices and students as edges. Two main approaches were compared: the Brute Force algorithm and the Degree of Saturation (DSATUR) heuristic. The results showed that although the Brute Force algorithm provides an optimal solution, it is computationally expensive. In contrast, the DSATUR heuristic proved to be extremely efficient in terms of execution time, although it does not always guarantee the optimal solution. The code for the proposed application is available at Google Colab.*

Resumo. *A pesquisa explora a aplicação da Teoria dos Grafos no problema de alocação de horários, focando nos alunos em ano de conclusão de curso. Utilizando a coloração de grafos, as disciplinas foram representadas como vértices e os alunos como arestas. Foram comparadas duas abordagens principais: o algoritmo de Força Bruta e a heurística Degree of Saturation (DSATUR). Os resultados mostraram que, embora o algoritmo de Força Bruta forneça uma solução ótima, ele é computacionalmente caro. Em contrapartida, a heurística DSATUR demonstrou ser extremamente eficiente em termos de tempo de execução, embora não garanta sempre a solução ótima. O código da aplicação proposta está disponível no Google Colab.*

1. Introdução

A Teoria dos Grafos é uma área da matemática que estuda a relação entre objetos, representados como vértices, e as conexões entre esses objetos, representadas como arestas. Esta teoria tem sido amplamente aplicada na resolução de problemas do cotidiano, desde o famoso Problema das Pontes de Königsberg até a modelagem de redes complexas em ciência da computação, biologia, transporte e redes sociais [Streicher and du Preez 2017]. Uma das principais vantagens de se trabalhar com grafos é a sua representação visual intuitiva, que facilita a análise e a solução de problemas complexos, mesmo para pessoas com conhecimento técnico limitado.

A organização dos horários nas Instituições de Ensino Superior (IES) é um desafio complexo, especialmente quando se trata de acomodar alunos com disciplinas pendentes no último semestre letivo. É crucial para as IES garantir que os alunos não ultrapassem o

período regular de formação, pois essa situação demanda mais recursos institucionais, incluindo tempo dos docentes, espaço físico em sala de aula e suporte administrativo. Além disso, considerando que as IES são frequentemente avaliadas com base em suas taxas de conclusão dentro do tempo padrão, a presença de alunos nessa situação pode diminuir essas taxas, impactando negativamente a reputação da instituição e sua capacidade de atrair novos estudantes.

Diante disso, este estudo explora a coloração de grafos como uma abordagem para mitigar o problema de alocação de horários, buscando uma distribuição eficiente das disciplinas e assegurando que não haja sobreposição de horários entre disciplinas com alunos em comum. Ao garantir que alunos com disciplinas pendentes possam completar seus cursos sem conflitos de horário, as instituições não apenas melhoram suas taxas de conclusão e eficiência operacional, mas também promovem uma experiência acadêmica mais positiva e produtiva para seus estudantes. Neste contexto, os vértices representam as disciplinas, e as arestas representam os alunos matriculados em disciplinas comuns.

Neste trabalho, a abordagem é organizada em seções para detalhar o problema e as soluções. Na Seção 2, discute-se a teoria da coloração de grafos e revisa-se a literatura. A Seção 3 examina métodos aplicados em contextos semelhantes, comparando-os com a proposta atual. A Seção 4 contextualiza o problema do IFMA, *campus* Caxias, na alocação de horários para alunos em conclusão. A metodologia e a modelagem são descritas na Seção 5. Na Seção 6, detalha-se a implementação dos algoritmos Força Bruta e DSATUR. A Seção 7 analisa os resultados dos experimentos e descreve o ambiente de testes. Por fim, na Seção 8, destacam-se as contribuições e sugestões para futuros trabalhos.

2. Referencial Teórico

Formalmente, um grafo G consiste em um conjunto de vértices $V(G)$ e um conjunto de arestas $E(G)$ que conectam pares de vértices. Segundo [Jensen and Toft 2011], essa estrutura permite modelar diversas situações que requerem a representação de relacionamentos binários entre elementos, fundamentando a teoria dos grafos e suas aplicações. A coloração de grafos, uma subárea da Teoria dos Grafos, destaca-se por oferecer uma abordagem matemática rigorosa e eficiente para resolver problemas, minimizando conflitos e otimizando a utilização de recursos [LABED et al. 2018].

Outras implementações, como a coloração de listas e a extensão de pré-coloração, também são frequentemente estudadas. Na coloração de listas, cada vértice possui uma lista de cores admissíveis, e o objetivo é encontrar uma coloração válida dentro dessas restrições. A extensão de pré-coloração envolve expandir uma coloração parcial de forma válida para todo o grafo [Tuza 1997]. Existem também modelos probabilísticos, como grafos de fatores e grafos de *clusters*, que têm se mostrado eficazes na formulação e solução de problemas de coloração de grafos. Em termos de precisão e eficiência computacional, [Streicher and du Preez 2017] indicam que grafos de *clusters* são superiores aos grafos fatoriais, especialmente em casos complexos.

De modo geral, para resolver problemas utilizando a coloração de grafos, podem ser consideradas duas abordagens principais: o uso de um algoritmo de Força Bruta e a aplicação de heurísticas. A técnica de Força Bruta é conhecida por sua alta precisão, pois testa todas as combinações possíveis para encontrar a solução ótima. Embora essa aborda-

gem garanta a solução ideal, ela apresenta a desvantagem de ter um tempo de execução exponencialmente elevado devido à complexidade combinatória [Bagheri et al. 2012]. Em contrapartida, embora heurísticas não garantam a solução ótima, elas são particularmente úteis, pois buscam encontrar uma solução satisfatória em um tempo razoável, utilizando significativamente menos recursos computacionais.

Técnicas como *Degree of Saturation* (DSATUR), *Recursive Largest First* (RLF) e *Tabu Search* (TS) são desenhadas para explorar o espaço de soluções de maneira inteligente, aumentando as chances de encontrar boas colorações [de Werra 1990]. A técnica DSATUR é uma heurística onde a ordem dos vértices é determinada dinamicamente durante o processo de coloração, priorizando vértices com maior grau de saturação (número de cores diferentes já atribuídas aos seus vizinhos). RLF é um algoritmo que colore recursivamente os vértices, começando pelo vértice de maior grau e escolhendo os subsequentes de forma a maximizar a utilização de cores. Por sua vez, *Tabu Search* é uma técnica de otimização que explora o espaço de soluções para evitar ciclos, utilizando uma lista tabu para impedir a repetição de movimentos recentes. [Duarte et al. 2021] destaca que a heurística DSATUR produz soluções ligeiramente com mais cores do que a RLF; no entanto, demonstra uma eficiência global significativamente melhor.

Sobre a complexidade dos problemas de coloração de grafos, esta varia significativamente com a estrutura do grafo e as restrições impostas. Alguns problemas são NP-completos, como é o caso geral da coloração de grafos [Blum 1994]. No entanto, para grafos específicos e condições adicionais, como em grafos bipartidos ou planos, existem algoritmos eficientes que podem ser aplicados [Tuza 1997].

3. Trabalhos Relacionados

No trabalho realizado por [Hussin et al. 2011], os autores utilizaram a abordagem de coloração de grafos para resolver o problema de *timetabling* de exames na Universidade Técnica da Malásia. O estudo emprega heurísticas de coloração de grafos para garantir que não haja conflitos de horários entre os exames. O artigo de [Egwuche 2020] também aborda a aplicação da coloração de grafos para resolver o problema de alocação de horários de exames, porém em instituições educacionais emergentes.

Em uma linha de pesquisa complementar, o estudo conduzido por [Pillay 2013] explora técnicas de otimização para problemas de alocação de horários, comparando diferentes abordagens de hiper-heurísticas na resolução do problema de *timetabling* escolar. O trabalho investiga a eficácia de quatro tipos de hiper-heurísticas: Construtiva de Seleção (SCHH), Construtiva de Geração (GCHH), Perturbativa de Seleção (SPHH) e uma abordagem híbrida (GPHH) que combina as duas anteriores. Embora o foco do estudo seja a alocação de horários em escolas, suas contribuições destacam a importância de combinar diferentes heurísticas para otimizar a alocação de recursos educacionais.

Embora os estudos de [Hussin et al. 2011], [Egwuche 2020] e [Pillay 2013] abordem a otimização da alocação de horários através de técnicas como a coloração de grafos e hiper-heurísticas, eles se concentram em contextos mais amplos, como a alocação de horários de exames ou a gestão de horários escolares em geral. Esses trabalhos, embora relevantes para a área de *timetabling*, não consideram as particularidades e desafios enfrentados por alunos em fase de conclusão de curso, em que a otimização de horários se torna ainda mais crítica para evitar sobrecarga e garantir a conclusão bem-sucedida

de suas disciplinas. O presente estudo se distingue ao focar diretamente nesse público específico, oferecendo uma solução mais adaptada às necessidades desses alunos, que frequentemente enfrentam restrições e demandas acadêmicas únicas em comparação aos demais estudantes.

4. Estudo de Caso

O Instituto Federal de Ciência e Tecnologia do Estado do Maranhão, *campus* Caxias, tem entre seus cursos mais procurados o bacharelado em Ciência da Computação, com ingresso anual de 40 alunos. Nesse contexto, tem-se observado uma alta taxa de retenção de alunos nesse curso, caracterizada pelo atraso na conclusão, que deveria ser integralizado em quatro anos.

Conforme dados extraídos do Sistema Unificado de Administração Pública (SUAP), cerca de 69% dos alunos formados até 2023 colaram grau com pelo menos um ano de atraso. Isso impacta, entre outros aspectos, a oferta de disciplinas e a construção dos horários, visto que o aluno em ano de formatura deve ter seu componente curricular pendente ofertado para que possa colar grau posteriormente.

Atualmente, entre os alunos vinculados ao curso, 54% apresentam componentes curriculares atrasados, sendo potenciais candidatos para uma oferta que atenda às suas necessidades de formatura no futuro. Em anos anteriores (2022 e 2023), o número de alunos nessa situação foi de oito em cada ano, além de outros doze, em situação semelhante, que optaram por não cursar naquele semestre.

Dada a relevância social e educacional deste problema, é evidente que, à medida que aumenta o número de alunos com essa demanda, cresce também a complexidade de organizar os horários sem conflito de disciplinas para quem precisa cursá-las naquele semestre. Portanto, a solução proposta para enfrentar essa questão pode ser amplamente aplicada em diversas instituições, beneficiando um maior número de estudantes e melhorando a eficiência dos cursos superiores.

5. Metodologia

Este trabalho é resultado de um projeto prático desenvolvido na disciplina de Programação e Análise de Algoritmos (PAA), evidenciando a aplicação prática dos conceitos teóricos estudados em sala de aula.

Inicialmente, foi selecionado o problema de Coloração de Grafos dentre vários outros problemas NP-completos sugeridos. A etapa seguinte envolveu a definição e descrição de um problema prático que poderia ser modelado pelo problema NP-completo escolhido. Nesta fase, contextualizamos o problema e justificamos sua relevância. Em seguida, implementamos uma instância do problema, desenvolvendo as estruturas de dados necessárias para representá-lo.

Para resolver o problema modelado, foram utilizadas duas abordagens principais: o uso de um algoritmo de Força Bruta e a aplicação de heurísticas. Para a realização de testes, ambos os algoritmos foram submetidos a duas cargas sintéticas de trabalho, uma estática e outra dinâmica. Dessa forma, este trabalho não apenas apresenta uma solução para o problema proposto, mas também fornece uma análise comparativa dos custos computacionais e da quantidade de cores utilizadas em cada uma das abordagens analisadas.

6. Algoritmos Propostos

Nesta seção, apresentamos os algoritmos de Força Bruta e DSATUR implementados para solucionar o problema de alocação de horários utilizando a coloração de grafos. É importante salientar que a solução proposta foca na alocação de horários, sem considerar fatores adicionais como choque de horários entre professores, pré-requisitos e disponibilidade de espaço físico. A seguir, detalhamos a implementação de cada algoritmo, incluindo a descrição das funções auxiliares utilizadas.

6.1. Algoritmo de Força Bruta

O algoritmo de Força Bruta foi implementado para explorar todas as possíveis combinações de alocação de horários para as disciplinas, garantindo que nenhuma compartilhe o mesmo horário se houver alunos em comum. O Algoritmo 1 apresenta o pseudocódigo para essa função.

Algorithm 1 Força Bruta

```
1:  procedure FORCABRUTA(disciplinas, alunos)
2:      alocacoes ← GERARALOCACOES(disciplinas)
3:      for i ← 0 to len(alocacoes) − 1 do
4:          if VERIFICARALOCACOES(alocacoes[i], disciplinas, alunos) then
5:              cores ← {}
6:              for j ← 0 to len(disciplinas) − 1 do
7:                  cores[disciplinas[j]] ← alocacoes[j]
8:              end for
9:              return cores
10:         end if
11:     end for
12:     return []
13: end procedure
```

A função acima utiliza as funções $gerarAlocacoes$ e $verificarAlocacoes$ para encontrar uma alocação válida de horários. Para demonstrar o funcionamento de ambas as funções, considera-se um cenário simplificado envolvendo três disciplinas, denotadas por **A**, **B** e **C**, e dois alunos: o Aluno 1 matriculado nas disciplinas **A** e **B** e o Aluno 2 nas disciplinas **B** e **C**.

Além disso, assume-se que existem dois horários disponíveis, representados por **1** e **2**. A função $gerarAlocacoes$ cria todas as possíveis alocações de horários para as disciplinas, gerando um produto cartesiano de todas as listas de alocações, representando todas as combinações possíveis. As combinações geradas para o exemplo são apresentadas a seguir.

Combinação	**A**	**B**	**C**
1	1	1	1
2	1	1	2
3	1	2	1
4	1	2	2
5	2	1	1
6	2	1	2
7	2	2	1
8	2	2	2

A função $verificarAlocacoes$ avalia cada combinação para garantir que não haja sobreposição de horários para disciplinas com alunos em comum. No exemplo apresentado, as combinações 3 e 6 são válidas, uma vez que asseguram que as disciplinas (A e B, bem como B e C) sejam alocadas em horários distintos, eliminando conflitos e tornando-as soluções viáveis para o problema de alocação de horários. Nesse contexto, ao encontrar uma alocação satisfatória, a função retorna um dicionário onde as chaves representam as disciplinas e os valores correspondem aos horários (cores) alocados. Essa combinação será então enviada para uma função subsequente, que gerará o grafo correspondente. Caso nenhuma alocação válida seja encontrada, a função retornará uma lista vazia.

6.2. Algoritmo DSATUR

A implementação deste algoritmo é uma heurística conhecida por proporcionar uma solução rápida, embora não necessariamente ótima. O código a seguir, representado pelo Algoritmo 2, apresenta o pseudocódigo para essa função.

Algorithm 2 DSATUR

```
1: procedure DSATUR(disciplinas, alunos)
2:     Grafo ← CRIARGRAFO(alunos)
3:     cores ← INICIALIZARCORES(disciplinas)
4:     disciplinasOrdenadas ← ORDENARDISCIPLINASGRAUSATURACAO(Grafo, cores)
5:     cores ← ATRIBUIRCORES(disciplinasOrdenadas, Grafo, cores)
6:     return cores
7: end procedure
```

No código acima, a função $criarGrafo$ é projetada para construir um grafo não direcionado que representa as relações de compartilhamento de alunos entre disciplinas. Neste grafo, as disciplinas são representadas pelos vértices, enquanto as arestas indicam o compartilhamento de alunos entre essas disciplinas. A função $inicializarCores$ é responsável por inicializar um dicionário que mapeia cada disciplina a uma cor inicial. Já a função $ordenarDisciplinasGrauSaturacao$ é projetada para ordenar as disciplinas em um grafo com base no grau de saturação. A função $atribuirCores$ é responsável por atribuir cores às disciplinas de forma a minimizar os conflitos de horários, seguindo a ordenação baseada no grau de saturação. Ao final, a função retorna o dicionário de cores atualizado, onde cada disciplina está associada a uma cor que minimiza conflitos de horários.

7. Experimentos

Nesta seção, analisamos os resultados obtidos a partir da implementação dos algoritmos de Força Bruta e DSATUR para a alocação de horários. Ambas as soluções foram testadas utilizando duas cargas sintéticas: uma definida pelos próprios pesquisadores e outra na qual as disciplinas foram geradas aleatoriamente e atribuídas dinamicamente aos alunos.

7.1. Carga Sintética Estática

Os resultados apresentados abaixo foram obtidos a partir da carga sintética descrita adiante.

1: **disciplinas** ← {WEB, PAA, PL, IA, BD, IHC, ES, PDI}
2: **alunos** ← {}
3: **alunos**[Lira] ← {(WEB, PL, IA, PDI)}
4: **alunos**[Carlos] ← {(PAA, PL, ES, PDI)}
5: **alunos**[Sofia] ← {(ES, PDI)}
6: **alunos**[Andre] ← {(PAA, BD, ES)}
7: **alunos**[Lauro] ← {(PAA, IA)}
8: **alunos**[Pedro] ← {(PAA, IHC)}

A Figura 1a apresenta o resultado obtido pelo algoritmo de Força Bruta, executado em 11,28 segundos. O grafo mostra que apenas quatro cores foram necessárias para colorir todos os vértices (disciplinas), representando uma solução sem sobreposições para as disciplinas compartilhadas por alunos em comum. Por sua vez, a Figura 1b apresenta o resultado da heurística DSATUR, executada em 0,0001 segundos. Embora esta heurística seja significativamente mais eficiente em termos de tempo de execução (como será discutido adiante), o algoritmo utilizou uma cor a mais, totalizando cinco cores para a coloração do grafo.

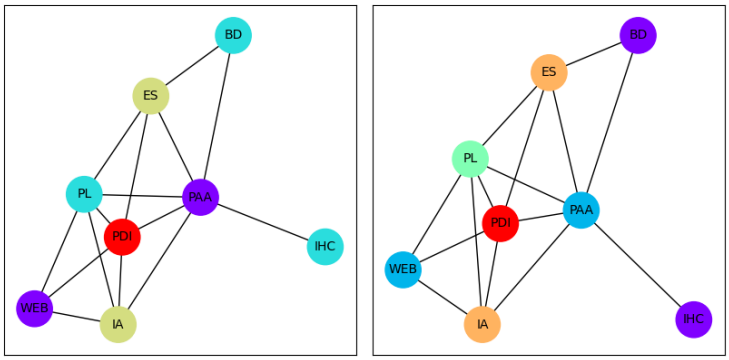

Figura 1. Força Bruta (a) x DSATUR (b)

Desse modo, disciplinas (vértices) com a mesma cor podem ser ofertadas simultaneamente, assegurando que não haverá prejuízo para os alunos, já que não haverá sobreposição de horários para disciplinas com alunos em comum.

7.2. Carga Sintética Dinâmica

Para a obtenção dos resultados, os dados foram coletados por meio de uma carga sintética, na qual o número de alunos (arestas) variava de 5 a 55, com incrementos de 5. O número máximo de disciplinas (vértices) foi definido como 8, pois valores superiores tornaram a execução do algoritmo de Força Bruta inviável devido à limitação computacional. Além disso, para fins de testes, determinou-se que o número máximo de disciplinas em que o aluno poderia estar reprovado fosse três. Todos os valores simulados podem ser ajustados conforme a necessidade de cada instituição.

Na Tabela 1, são apresentados os dados do tempo de execução de cada algoritmo. As medições de tempo foram convertidas para notação científica, facilitando a análise comparativa em escalas de tempo substancialmente diferentes.

No. de Alunos	Tempo de Execução (Força Bruta)	Tempo de Execução (DSATUR)
5	1.28e+01	9.41e-05
10	1.46e+01	1.28e-04
15	1.25e+01	1.32e-04
20	1.29e+01	1.49e-04
25	1.44e+01	1.90e-04
30	1.26e+01	1.82e-04
35	1.41e+01	2.06e-04
40	1.33e+01	2.21e-04
45	1.51e+01	2.27e-04
50	1.32e+01	2.52e-04
55	1.24e+01	2.65e-04

Tabela 1. Tempo de Execução: Força Bruta x DSATUR

A partir dos dados apresentados na Tabela 1, podemos observar que o algoritmo de Força Bruta apresenta um tempo de execução relativamente constante, variando de 1.24e+01 a 1.51e+01. Este comportamento indica que, apesar do aumento no número de arestas (alunos), a complexidade computacional do algoritmo não aumenta drasticamente dentro do intervalo testado. No entanto, é importante notar que este tempo de execução representa um valor elevado (aproximadamente 12 a 15 segundos), dado que o número de instâncias fornecidas é relativamente pequeno. Por outro lado, embora o algoritmo DSATUR demonstre um aumento no tempo de execução crescente, conforme o número de alunos cresce, ele ainda assim é mais eficiente em termos de tempo de execução (ordem de milissegundos) quando comparado ao algoritmo de Força Bruta, tornando-se adequado para aplicações práticas onde a velocidade é crucial.

A Figura 2 apresenta também a eficiência temporal dos algoritmos testados. A escala logarítmica no eixo do tempo permite uma comparação clara das ordens de magnitude dos tempos de execução de ambos os algoritmos. O DSATUR não apenas supera o Força Bruta em termos de tempo de execução, mas também mantém vantagem de forma consistente à medida que o tamanho do problema aumenta. Logo, para aplicações práticas onde a eficiência temporal é crítica, o DSATUR se destaca como a melhor escolha em comparação ao Força Bruta.

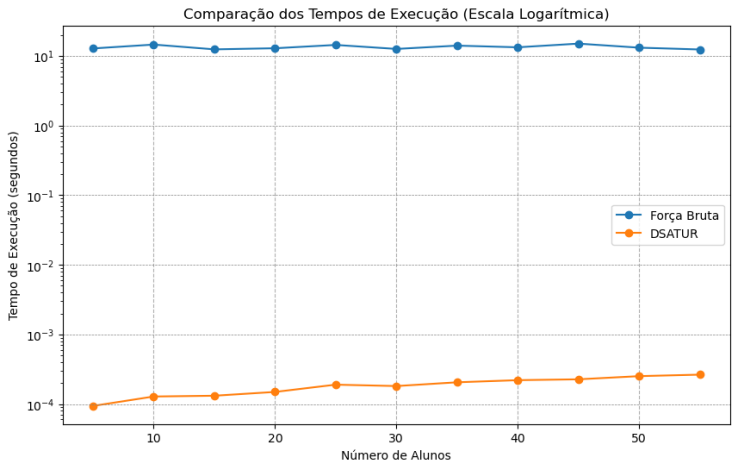

Figura 2. Comparação dos Tempos de Execução

Já a Tabela 2, apresenta a quantidade de cores necessárias para colorir o grafo

utilizando ambas as estratégias. Como pode ser observado, os dois algoritmos apresentam resultados bastante semelhantes em termos do número de cores utilizadas para a coloração do grafo. Em todos os casos testados, os dois algoritmos utilizam a mesma quantidade de cores para números maiores de alunos (arestas), especialmente a partir de 25 alunos. Essa estabilização no número de cores sugere que o problema de coloração atinge um ponto de saturação, onde adicionar mais alunos não aumenta a complexidade do grafo de maneira significativa para requerer mais cores.

Portanto, a análise dos dados apresentados na Tabela 2 indica que as implementações propostas têm desempenho semelhante na tarefa de coloração de grafos. Com exceção de alguns poucos casos, ambos os algoritmos usam a mesma quantidade de cores para todos os tamanhos de entrada testados. Esta análise reforça a escolha do DSATUR para aplicações práticas devido ao seu menor tempo de execução, sem comprometer a qualidade da solução em termos de número de cores utilizadas.

No. de Alunos	Cores (Força Bruta)	Cores (DSATUR)
5	3	4
10	5	6
15	5	6
20	5	6
25	8	8
30	7	7
35	7	8
40	8	8
45	8	8
50	8	8
55	8	8

Tabela 2. Quantidade de Cores: Força Bruta x DSATUR

7.3. Ambiente dos Experimentos

O computador utilizado para os testes dos algoritmos operava sob o sistema Ubuntu 22.04, com a versão do kernel 6.2.0-37-generic. A linguagem de programação usada para implementar os algoritmos foi Python, na versão 3.11.7. Quanto ao *hardware*, o processador utilizado foi um Intel(R) Core(TM) i7-10700K CPU @ 3.80GHz, com 16 núcleos. Além disso, o sistema contava com 128 GB de RAM, com uma velocidade de 2133 MT/s. O código da implementação pode ser acessado facilmente a partir do Google Colab[1].

8. Conclusão

O estudo apresentado explora a aplicação da teoria dos grafos, especificamente a coloração de grafos, no problema de alocação de horários de disciplinas em Instituições de Ensino Superior (IES). Focando em alunos prestes a se formar, foi demonstrado como técnicas matemáticas aplicadas à teoria dos grafos podem minimizar conflitos de horários e otimizar o uso dos recursos da instituição.

Dois algoritmos foram implementados e analisados: Força Bruta e DSATUR. A análise comparativa entre os algoritmos revelou que, embora o algoritmo de Força Bruta garanta a solução ótima, ele apresenta um tempo de execução exponencialmente elevado, tornando-se impraticável para grandes conjuntos de dados. Por outro lado, o algoritmo DSATUR mostrou-se significativamente mais eficiente quanto ao tempo de execução,

[1] Google Colab Link

mantendo um desempenho competitivo no número de cores utilizadas. Os resultados também indicam que disciplinas com a mesma cor podem ser ofertadas simultaneamente sem prejudicar os alunos, uma vez que não haverá sobreposição de horários para alunos em comum.

Por fim, este estudo, além de oferecer uma solução prática para a alocação de horários, também contribui para a compreensão das diferenças entre abordagens exatas e heurísticas na resolução de problemas de coloração de grafos. Futuras pesquisas podem explorar outras heurísticas e algoritmos híbridos, assim como a aplicação de técnicas de aprendizado de máquina para aprimorar ainda mais a eficiência e a adaptabilidade das soluções propostas.

Referências

[Bagheri et al. 2012] Bagheri, E., Noia, T. D., Gasevic, D., and Ragone, A. (2012). Formalizing interactive staged feature model configuration. *Journal of Software: Evolution and Process*, 24(4):375–400.

[Blum 1994] Blum, A. (1994). New approximation algorithms for graph coloring. *J. ACM*, 41(3):470–516.

[de Werra 1990] de Werra, D. (1990). *Heuristics for Graph Coloring*, pages 191–208. Springer Vienna, Vienna.

[Duarte et al. 2021] Duarte, I., Cancela, L., and Rebola, J. (2021). Graph coloring heuristics for optical networks planning. In *2021 Telecoms Conference (ConfTELE)*, pages 1–6.

[Egwuche 2020] Egwuche, O. S. (2020). Examination timetabling with graph coloring for emerging institutions. In *2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS)*, pages 1–6.

[Hussin et al. 2011] Hussin, B., Basari, A. S. H., Shibghatullah, A. S., Asmai, S. A., and Othman, N. S. (2011). Exam timetabling using graph colouring approach. In *2011 IEEE Conference on Open Systems*, pages 133–138.

[Jensen and Toft 2011] Jensen, T. and Toft, B. (2011). *Graph Coloring Problems*. Wiley Series in Discrete Mathematics and Optimization. Wiley.

[LABED et al. 2018] LABED, S., KOUT, A., and CHIKHI, S. (2018). A fast heuristic for graph b-coloring problem. In *2018 JCCO Joint International Conference on ICT in Education and Training, International Conference on Computing in Arabic, and International Conference on Geocomputing (JCCO: TICET-ICCA-GECO)*, pages 1–6.

[Pillay 2013] Pillay, N. (2013). A comparative study of hyper-heuristics for solving the school timetabling problem. In *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*, SAICSIT '13, page 278–285, New York, NY, USA. Association for Computing Machinery.

[Streicher and du Preez 2017] Streicher, S. and du Preez, J. (2017). Graph coloring: Comparing cluster graphs to factor graphs. In *Proceedings of the ACM Multimedia 2017 Workshop on South African Academic Participation*, SAWACMMM '17, page 35–42, New York, NY, USA. Association for Computing Machinery.

[Tuza 1997] Tuza, Z. (1997). Graph colorings with local constraints - a survey. *Discussiones Mathematicae Graph Theory*, 17(2):161–228.