Análise Comparativa dos Algoritmos DSATUR e RLF para Coloração de Grafos

Ana Julia Vieira Pereira Andrade da Costa
Gabriel Peixoto Menezes da Costa

***Resumo* – Este trabalho apresenta uma avaliação comparativa de duas heurísticas de coloração de grafos, DSATUR e Recursive Largest First (RLF), sobre instâncias padrão e grafos aleatórios de tamanhos variados (50–2000 vértices) e densidades diversas (0,1–0,5). Medimos tempo de execução, número de cores usadas e eficiência (cores por vértice), além de estimar a complexidade empírica em escala log–log.**

***Abstract* – This paper presents a comparative study between two graph coloring heuristics: DSATUR and Recursive Largest First (RLF). We conduct extensive benchmarks on both random graphs (50–2000 vertices, densities 0.1–0.5) and classic instances, measuring execution time, color count, and coloring efficiency (colors per vertex). We also perform an empirical complexity analysis in a log–log scale.**

***Index Terms*—graph coloring, NP-complete, heuristics, DSATUR, RLF, algorithm comparison**

I. Introdução

A. Ilustração do Problema

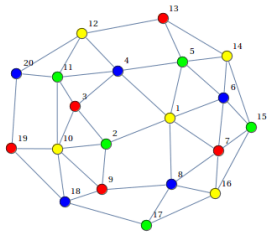

Fig. 1: Exemplo de coloração de grafo com múltiplas cores.

Esta ilustração (Figura 1) mostra um exemplo de como vértices de um grafo podem ser coloridos de forma a não haver conflitos entre adjacentes. Ela serve de motivação visual para as heurísticas que serão comparadas neste trabalho.

B. Contexto Geral

A coloração de grafos é um problema clássico e fundamental da Teoria dos Grafos e da Ciência da Computação. Seu objetivo é colorir os vértices de um grafo de modo que vértices adjacentes tenham cores diferentes, utilizando o menor número possível de cores.

C. Importância

Esse problema aparece em contextos práticos como escalonamento de tarefas, alocação de recursos, compiladores, e redes sem fio, onde conflitos entre elementos adjacentes devem ser evitados [1], [2].

D. Desafio Computacional

A coloração ótima de grafos é NP-difícil, o que significa que soluções exatas são inviáveis para grafos grandes. Por isso, heurísticas como RLF e DSATUR são amplamente estudadas por sua capacidade de gerar soluções próximas do ótimo com boa eficiência.

II. Motivação para o Uso dos Algoritmos RLF e DSATUR

A escolha pelos algoritmos RLF e DSATUR se justifica pelos motivos abaixo:

- O RLF tende a produzir soluções com menor número de cores em grafos esparsos, explorando conjuntos independentes grandes, como demonstrado por Leighton [3].
- O DSATUR, proposto por Brélaz [4], oferece excelente desempenho em grafos densos devido à sua natureza adaptativa, respondendo dinamicamente à saturação de vértices.
- Ambos são referências clássicas na literatura e frequentemente utilizados como base para variantes modernas [5]–[7].

A análise comparativa entre ambos fornece uma visão equilibrada entre qualidade de coloração e tempo de execução, o que é essencial para aplicações práticas de larga escala [8].

III. Definição do Problema e Contexto

A. Formalização

Seja um grafo simples não direcionado $G = (V, E)$. Uma *coloração própria* de G é uma função

$$c : V \to \mathbb{N}$$

tal que

$$c(u) \neq c(v), \quad \forall \{u, v\} \in E.$$

O objetivo é minimizar o número cromático

$$\chi(G) \;=\; |\, c(v) : v \in V \,|$$

isto é, o número de cores distintas utilizadas em G [9]. Cada cor corresponde a uma classe independente (conjunto de vértices mutuamente não adjacentes).

B. NP-Completude e Generalizações

Decidir se $\chi(G) \leq k$ para um dado inteiro $k \geq 3$ é um problema NP-completo, conforme demonstrado por Garey & Johnson [9]. Malaguti & Toth apresentam extensões importantes do problema clássico de coloração de vértices, tais como:

- **Bandwidth Coloring Problem (BCP):** impõe restrições de distância mínima entre cores de vértices adjacentes;
- **Weighted Vertex Coloring Problem (WVCP):** atribui pesos ou custos às cores, visando minimizar o custo total;
- **Multicoloring e Binding Constraints:** cada vértice pode requerer múltiplas cores ou estar sujeito a limites de capacidade [1].

C. Aplicações Práticas

A coloração de grafos é amplamente utilizada em diversos domínios:

- **Escalonamento de tarefas e exames:** cada tarefa (ou exame) é representada por um vértice, e arestas indicam conflitos de recursos ou horários; a coloração minimiza o número de slots temporais necessários [2].
- **Alocação de registradores em compiladores:** variáveis são vértices em um grafo de interferência; arestas conectam variáveis que precisam coexistir em registradores distintos, e cores correspondem a registradores físicos [2].
- **Atribuição de canais Wi-Fi:** modela-se a interferência entre transmissores como arestas ponderadas e aplica-se uma heurística de coloração espectral para minimizar interferência máxima na rede [10].

D. Desafios e Escolha de Heurísticas

Para grafos de grande escala (centenas a milhares de vértices), soluções exatas tornam-se impraticáveis, o que motiva o uso de heurísticas polinomiais [1]. Entre as heurísticas mais estudadas estão:

- **DSATUR** [4]: adapta dinamicamente a prioridade dos vértices com base no grau de saturação e no grau original, sendo eficiente em grafos densos.
- **RLF** [3]: constrói iterativamente grandes conjuntos independentes, mostrando bom desempenho em grafos esparsos e modulares.

Estudos recentes com variantes matheurísticas e híbridas demonstram ganhos adicionais em qualidade de coloração e tempo de execução [5]–[8].

IV. Algoritmos

Nesta seção, descrevemos duas heurísticas gulosas utilizadas para o problema de coloração de grafos: RLF (Recursive Largest First) e DSATUR (Degree of Saturation). Ambas seguem estratégias distintas para construção de soluções aproximadas e são amplamente utilizadas devido à sua simplicidade e eficiência prática.

A. Algoritmo RLF

Estratégia de Construção de Conjuntos Independentes: O RLF baseia-se na ideia de, a cada nova cor, formar o maior conjunto independente possível a partir de um vértice raiz de grau máximo. Para isso:

1) Seleciona-se o vértice v não colorido de maior grau no subgrafo residual.
2) Inicia-se um conjunto independente $I = \{v\}$.
3) Para cada vértice u não adjacente a todos em I, avalia-se se pode ser adicionado a I sem criar conflito — se sim, inclui-se u.
4) Repete-se o passo anterior até não restar vértice elegível.
5) Atribui-se a cor atual a todos os vértices de I e parte-se para a próxima cor.

Essa técnica maximiza localmente o tamanho de cada conjunto independente, reduzindo o número total de cores.

O algoritmo completo fica então:

Algorithm 1 Coloração RLF

Require: Grafo $G = (V, E)$
Ensure: Vetor de cores $colors[\,]$
$n \leftarrow |V|$
Inicialize $colors[v] \leftarrow -1,\ \forall v \in V$
Inicialize $colored[v] \leftarrow$ false, $\forall v \in V$
$color \leftarrow 0$
while existe $v \in V$ com $colored[v] =$ false **do**
    Escolha v não colorido com grau máximo no subgrafo de não coloridos
    $colors[v] \leftarrow color$
    $colored[v] \leftarrow$ true
    for all $u \in V$ com $colored[u] =$ false e $\{u, v\} \notin E$ **do**
        $canColor \leftarrow$ true
        for all $w \in V$ com $colored[w] =$ true e $colors[w] = color$ **do**
            if $\{u, w\} \in E$ **then**
                $canColor \leftarrow$ false
            end if
        end for
        if $canColor$ **then**
            $colors[u] \leftarrow color$
            $colored[u] \leftarrow$ true
        end if
    end for
    $color \leftarrow color + 1$
end while
return $colors$

B. Algoritmo DSATUR

Métrica de Saturação e Atualização Dinâmica: O cerne do DSATUR é a *saturação* de um vértice, definida como o número de cores diferentes já atribuídas aos seus vizinhos ainda não coloridos. A cada passo:

1) Seleciona-se o vértice não colorido de maior saturação — isto é, o que tem menos opções de cores livres.
2) Em caso de empate, escolhe-se, entre eles, o vértice de maior grau no grafo original.
3) Após pintar esse vértice com a menor cor disponível, incrementa-se em 1 o valor da saturação de todos os seus vizinhos não coloridos, pois eles agora perderam uma opção de cor.

Esse mecanismo garante que o DSATUR ajuste dinamicamente suas prioridades, focando sempre nos vértices mais "constrangidos" no momento.

Algorithm 2 Coloração DSATUR

Require: Grafo $G = (V, E)$
Ensure: Vetor de cores $colors[\,]$
$n \leftarrow |V|$
Inicialize $colors[v] \leftarrow -1,\ \forall v \in V$
Para cada $v \in V$:
    $degree[v] \leftarrow |\{u : \{v, u\} \in E\}|$
    $saturation[v] \leftarrow 0$
$colored[v] \leftarrow$ false, $\forall v \in V$
for $i = 1$ **até** n **do**
    Ordene os vértices não coloridos por:
    1) maior $saturation$, 2) desempate por maior $degree$
    Selecione o primeiro vértice v não colorido
    Construa $available[0 \ldots n-1] \leftarrow$ false
    for all u vizinho de v com $colors[u] \neq -1$ **do**
        $available[\,colors[u]\,] \leftarrow$ true
    end for
    Escolha a menor cor c tal que $available[c] =$ false
    $colors[v] \leftarrow c$
    $colored[v] \leftarrow$ true
    for all u vizinho de v com $colored[u] =$ false **do**
        $saturation[u] \leftarrow saturation[u] + 1$
    end for
end for
return $colors$

V. Resultados Experimentais

A. Ambiente de Testes

Todos os testes foram conduzidos em um processador Intel i7-1165G7 com 16 GB de RAM.

B. Instâncias de Benchmark

A Tabela I mostra os tempos de execução (ms) em três instâncias clássicas:

TABLE I: Tempos de Execução em *Benchmarks* Clássicas

Instância	RLF (ms)	DSATUR (ms)
miles1500.txt	30.89	1.00
myciel7.txt	1.00	2.00
queen16_16.txt	1.00	3.00

C. Grafos Aleatórios

Para grafos gerados aleatoriamente (tamanhos de 50 a 2000 vértices, densidades de 0.1 a 0.5), resumimos na Tabela II as médias gerais de tempo e detalhamos na Figura 2:

TABLE II: Tempos Médios de Execução em Grafos Aleatórios

Algoritmo	Tempo Médio (ms)
DSATUR	68.96
RLF	30.89

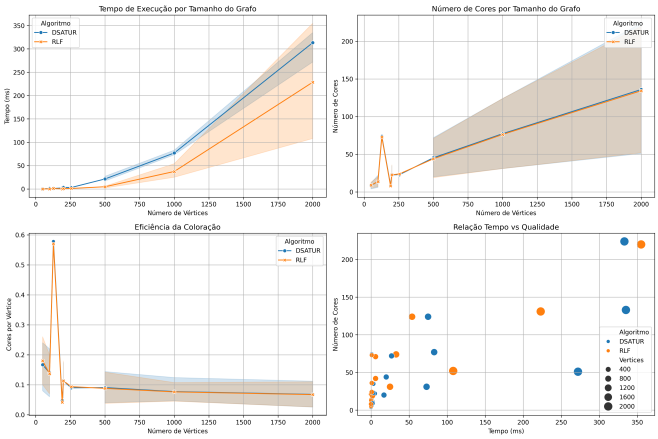

Fig. 2: Resultados em grafos aleatórios: (a) Tempo de execução por tamanho de grafo; (b) Número de cores por tamanho; (c) Eficiência de coloração (cores por vértice); (d) Relação entre tempo e qualidade (tamanho indicado pelo diâmetro do ponto).

D. Análise de Complexidade

A análise em escala log–log (Fig. 3) mostra que o DSATUR segue aproximadamente $O(n^{1.99})$, enquanto o RLF cresce como $O(n^{2.22})$, o que explica sua pior escalabilidade em grafos muito grandes.

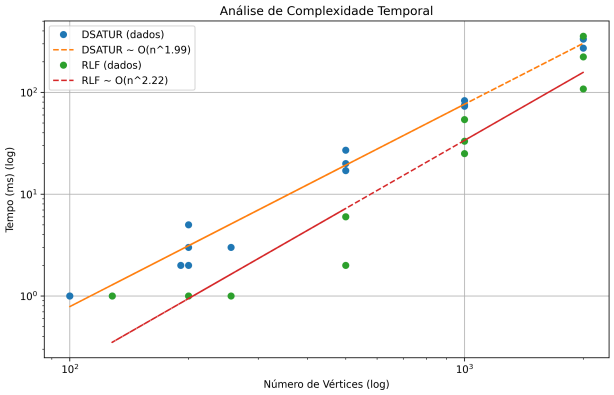

Fig. 3: Comparação de complexidade temporal em escala log–log.

VI. Conclusão

Os resultados experimentais apresentados neste trabalho ressaltam características complementares entre as heurísticas DSATUR e RLF, orientando sua aplicação conforme natureza e tamanho dos grafos:

- Em termos de **tempo de execução**, o RLF demonstrou, em média, desempenho superior sobre grafos aleatórios (30,9 ms contra 68,9 ms do DSATUR), especialmente em instâncias esparsas e de tamanho moderado (até 1 000 vértices), devido à sua construção de grandes conjuntos independentes (Fig. 2a e Tabela II).
- Quanto à **qualidade da coloração**, ambos os algoritmos se equiparam em grafos densos ou de maior escala, com variações mínimas no número de cores utilizadas (diferença média inferior a 1%), embora o DSATUR apresente ligeira vantagem em cenários de alta densidade (Fig. 2b).
- A **eficiência de coloração** (cores por vértice) converge para valores próximos em instâncias grandes ($\approx 0,08$), indicando escalabilidade similar na proporção cores/vértice (Fig. 2c).
- Na **análise de complexidade empírica**, observou-se que o DSATUR cresce como $O(n^{1.99})$, enquanto o RLF segue uma curva $O(n^{2.22})$ (Fig. 3), justificando o comportamento de maior desaceleração do RLF em grafos muito grandes.

Dessa forma, recomenda-se:

1) **RLF** para aplicações que priorizem rapidez em grafos esparsos ou de tamanho até cerca de 1 000 vértices.
2) **DSATUR** em cenários de alta densidade e em grafos muito grandes, onde sua melhor escalabilidade e ligeira redução no número de cores compensam o tempo adicional.

Para trabalhos futuros, propõe-se a investigação de *algoritmos híbridos* que combinem a velocidade do RLF na etapa inicial com o refinamento adaptativo do DSATUR, bem como a exploração de aceleração em GPU para mitigar o impacto da complexidade quasilinear em instâncias de grande escala.

References

[1] E. Malaguti and P. Toth, "A survey on vertex coloring problems," *International Transactions in Operational Research*, vol. 17, no. 1, pp. 1–34, 2010. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1475-3995.2009.00696.x

[2] P. Formanowicz and K. Tanaś, "A survey of graph coloring - its types, methods and applications," *Foundations of Computing and Decision Sciences*, vol. 37, no. 3, pp. 223–238, 2012. [Online]. Available: https://doi.org/10.2478/v10209-011-0012-y

[3] F. T. Leighton, "A graph coloring algorithm for large scheduling problems," *Journal of research of the National Bureau of Standards (1977)*, vol. 84, no. 6, pp. 489–506, Nov-Dec 1979. [Online]. Available: https://doi.org/10.6028/jres.084.024

[4] D. Brélaz, "New methods to color the vertices of a graph," *Commun. ACM*, vol. 22, no. 4, p. 251–256, Apr. 1979. [Online]. Available: https://doi.org/10.1145/359094.359101

[5] N. Dupin, "Matheuristic variants of DSATUR for the vertex coloring problem," Feb. 2024, working paper or preprint. [Online]. Available: https://hal.science/hal-04465758

[6] P. Verma, "Degree-based logical adjacency checking (dblac): A novel heuristic for vertex coloring," 2025. [Online]. Available: https://arxiv.org/abs/2501.12479

[7] E. Zhu, Y. Zhang, H. Sun, Z. Wei, W. Pedrycz, C. Liu, and J. Xu, "Hycolor: An efficient heuristic algorithm for graph coloring," 2025. [Online]. Available: https://arxiv.org/abs/2506.07373

[8] B. Qehaja, E. Hajrizi, G. I. Marinova, K. Dhoska, and L. Berisha, "A hybrid graph-coloring and metaheuristic framework for resource allocation in dynamic e-health wireless sensor networks," *Preprints*, July 2025. [Online]. Available: https://doi.org/10.20944/preprints202507.0720.v1

[9] J. Hartmanis, "Computers and intractability: A guide to the theory of np-completeness (michael r. garey and david s. johnson)," *SIAM Review*, vol. 24, no. 1, pp. 90–91, 1982. [Online]. Available: https://doi.org/10.1137/1024022

[10] D. Orden, J. M. Gimenez-Guzman, I. Marsa-Maestre, and E. De la Hoz, "Spectrum graph coloring and applications to wi-fi channel assignment," *Symmetry*, vol. 10, no. 3, 2018. [Online]. Available: https://www.mdpi.com/2073-8994/10/3/65