Fundamentos de Inteligência Artificial
Departamento de Engenharia Elétrica – UFMG

Heurísticas para Coloração de Grafos: Um Estudo Comparativo

Allan Melquíades Bernardo Fonseca Gabriel Veloso
Lucas Pimenta Luan Borges

Trabalho Prático 2 – Outubro/2025

Resumo
Este relatório descreve a implementação e a avaliação de múltiplas heurísticas para o problema de coloração de grafos: Random Walk (RW), Best Improvement (BI), First Improvement com busca aleatória (FI-RS), First Improvement com vértice em conflito (FI-AC), Simulated Annealing (SA), Algoritmo Genético (GA) e DSATUR. O estudo considera três instâncias (grafo exemplo, `myciel3.col` e `queen5_5.col`) e discute conflitos finais, número de cores, tempos, distribuições de runtime e robustez das abordagens.

Sumário

1 Introdução

A coloração de grafos consiste em atribuir cores a todos os vértices de um grafo $G = (V, E)$, garantindo que vértices adjacentes recebam cores distintas. Trata-se de um problema clássico de satisfação de restrições (CSP) e aparece em aplicações de alocação, escalonamento e compiladores. Como o problema é NP-completo, heurísticas de busca são frequentes na prática. Este trabalho aborda cinco heurísticas de busca local, um algoritmo genético e a heurística DSATUR.

2 Heurísticas Implementadas

Implementamos todas as heurísticas no notebook `TP2_CSP_LocalSearch_GA.ipynb`. Seja $h(c)$ o número de conflitos (arestas com extremidades da mesma cor) em uma coloração c.

2.1 Random Walk (RW)

Seleciona um vértice v aleatório e muda sua cor para uma cor aleatória c'. Se $h(c') \leq h(c)$, a mudança é aceita; caso contrário, pode ser aceita com probabilidade $p = 0{,}1$ para escapar de mínimos locais.

2.2 Best Improvement (BI)

Avalia todas as combinações vértice–cor e aplica a mudança que mais reduz $h(c)$. Se nenhuma combinação melhora a solução, a heurística termina.

2.3 First Improvement – Random Search (FI-RS)

Escolhe vértices e cores aleatórios; ao encontrar a primeira mudança que reduz $h(c)$, aplica imediatamente (estratégia first improvement). Em caso de estagnação, interrompe.

2.4 First Improvement – Any Conflict (FI-AC)

Restringe-se aos vértices atualmente em conflito. Para cada vértice, testa-se todas as cores disponíveis. Aceita-se a primeira cor que reduz $h(c)$; se nenhuma reduz, encerra.

2.5 Simulated Annealing (SA)

Mantém uma temperatura T decrescente. Dado $\Delta = h(c') - h(c)$, aceita c' com probabilidade 1 se $\Delta \leq 0$ e $\exp(-\Delta/T)$ caso contrário. Utilizamos $T_0 = 100$, fator de resfriamento $\alpha = 0{,}98$ e truncamento em $T_{\min} = 10^{-9}$.

2.6 Algoritmo Genético (GA)

Cada indivíduo da população representa uma coloração. O fitness é $h(c)$. Configuração:

- População inicial: 60 indivíduos aleatórios.
- Seleção: proporcional a $\exp(-h/T)$ com T inicial 10 e decaimento $\alpha = 0{,}95$.

- Crossover: ponto único sobre a lista de vértices.
- Mutação: altera a cor de um vértice com probabilidade 0,05.
- Gerações: 150; encerra ao obter $h = 0$ ou atingir o limite.

2.7 DSATUR

Heurística determinística que seleciona, em cada passo, o vértice não colorido com maior saturação (número de cores distintas utilizadas por seus vizinhos). Em caso de empate, escolhe-se o vértice de maior grau. Atribui-se a menor cor viável. Conforme Brélaz [2], DSATUR produz colorações muito próximas do cromático.

3 Metodologia Experimental

3.1 Instâncias

a) **Grafo exemplo**: 10 vértices, 16 arestas, cromático 4.

b) `myciel3.col`: grafo de Mycielski com 11 vértices, cromático 4.

c) `queen5_5.col`: grafo das posições de uma rainha em tabuleiro 5×5, 25 vértices, cromático 5.

As instâncias adicionais foram obtidas em `https://mat.tepper.cmu.edu/COLOR/instances.html` e estão em `data/instances/`.

3.2 Métricas

Para cada heurística e instância foram registradas:

- número de conflitos ao término;
- número de cores distintas utilizadas;
- tempo de execução;
- taxa de sucesso (rodadas com zero conflitos).

3.3 Protocolo

i) As heurísticas locais (RW, BI, FI-RS, FI-AC, SA) e o GA partem de colorações aleatórias; DSATUR é determinístico.

ii) Executamos 20 rodadas independentes para cada heurística/instância.

iii) Os tempos resultantes geraram gráficos de runtime distribution, conforme [3].

Tabela 1: Taxa de sucesso e tempo mediano (20 execuções por heurística).

Heurística	Grafo Exemplo		myciel3		queen5_5	
	Sucesso (%)	Tempo (ms)	Sucesso (%)	Tempo (ms)	Sucesso (%)	Tempo (ms)
RW	100.00	0.30	100.00	0.40	0.00	134.70
BI	95.00	0.60	100.00	1.00	5.00	110.80
FI-RS	95.00	0.30	90.00	0.30	30.00	131.40
FI-AC	80.00	0.20	100.00	0.20	0.00	715.80
SA	100.00	1.90	100.00	2.30	60.00	122.00
GA	100.00	2.80	100.00	6.00	5.00	1003.60
DSATUR	100.00	0.00	100.00	0.10	100.00	0.80

4 Resultados

4.1 Visão Geral

A Tabela 1 resume as taxas de sucesso (em %) e tempos medianos (ms) obtidos nas 20 execuções por heurística.

4.2 Discussão Detalhada

Nas instâncias mais simples (grafo exemplo e `myciel3`), todas as heurísticas produziram soluções sem conflitos com quatro cores; porém, como RW, FI-RS e FI-AC dependem fortemente da semente inicial, eles falharam em algumas rodadas (taxas de sucesso de 80–95%). BI, FI-AC e DSATUR mantiveram tempos inferiores a 1 ms; RW e FI-RS também foram rápidos graças à baixa densidade. SA e GA, embora mais custosos, preservaram 100% de sucesso nessas instâncias e continuam abaixo de 7 ms.

Na instância `queen5_5` as diferenças são mais marcantes: RW não obteve soluções válidas nas 20 rodadas; BI convergiu em apenas 5% das tentativas; FI-RS manteve 30% de sucesso, mas com tempo próximo a 130 ms. FI-AC não encontrou nenhuma solução, evidenciando aprisionamento em mínimos locais. SA tornou-se a abordagem estocástica mais robusta (60% de sucesso, mediana de 122 ms). O GA, com a configuração atual, alcançou 5% de sucesso e tempos médios elevados; ajustes de população/gerações podem melhorar esse número. DSATUR permaneceu perfeito, encontrando a coloração cromática com tempo praticamente constante ($\approx 0{,}8$ ms).

Por que heurísticas aleatórias falham? O grafo `queen5_5` é denso, de modo que qualquer alteração local em um vértice tende a induzir conflitos em vários vizinhos; Random Walk e FI-RS, que mudam vértices/cor de maneira rapidamente aleatória, têm dificuldade de escapar desses mínimos locais. FI-AC procura somente no conjunto de vértices em conflito e, ao aceitar a primeira melhora, frequentemente estabiliza numa coloração parcialmente adequada, mas que não admite melhoras locais. BI também sofre com platôs: se todas as alterações pioram, ele fica estagnado. Já o GA, com população pequena e poucas gerações, perde diversidade rapidamente. Em contraste, o SA aceita quedas de qualidade com probabilidade controlada, facilitando a saída de mínimos locais, enquanto o DSATUR constrói a solução de maneira determinística e informada, evitando conflitos desde o início, razões pelas quais apresentaram as maiores taxas de sucesso.

4.3 Runtime Distribution

As Figuras 1 a 3 apresentam as distribuições de runtime acumulado já com o DSATUR incluído. Em instâncias simples, todas as curvas sobem quase verticalmente na origem, reforçando tempos desprezíveis; o traço de DSATUR aparece colado ao eixo, pois o tempo é praticamente constante. Em `queen5_5`, RW permanece no zero (sem sucessos), BI e FI-AC têm pequenos degraus, FI-RS e SA mostram crescimento gradual com sucesso parcial, o GA exibe cauda longa e o DSATUR surge como degrau imediato próximo de 0 ms.

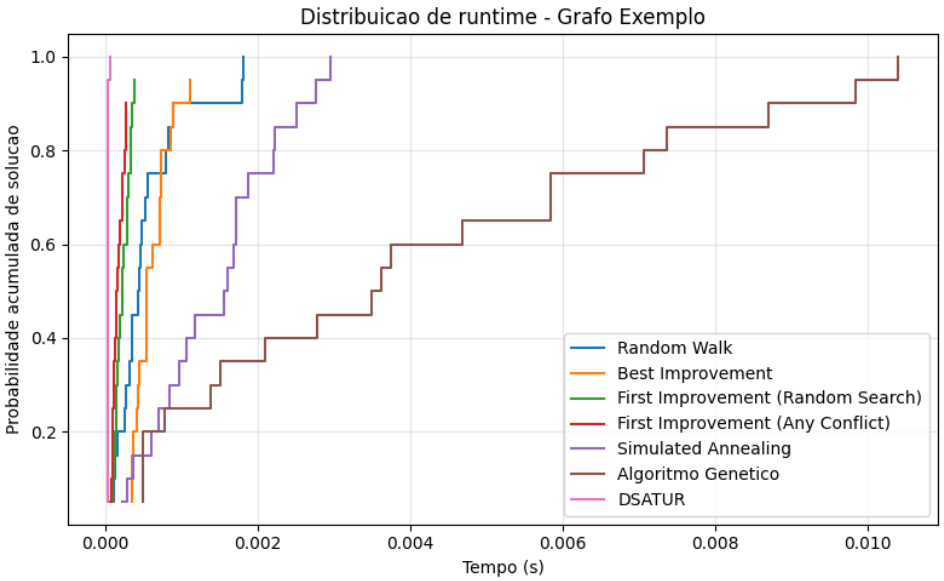

Figura 1: Runtime distribution – grafo exemplo.

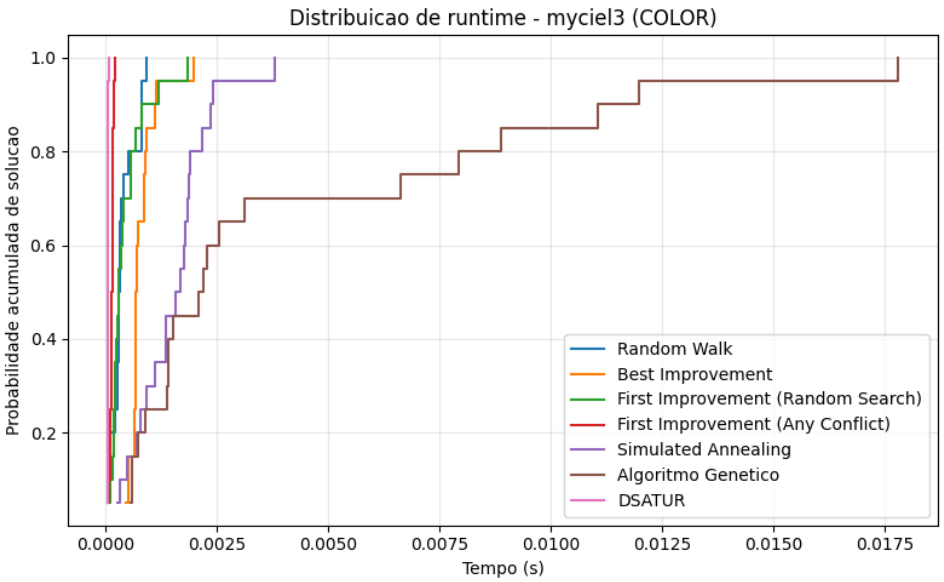

Figura 2: Runtime distribution – `myciel3`.

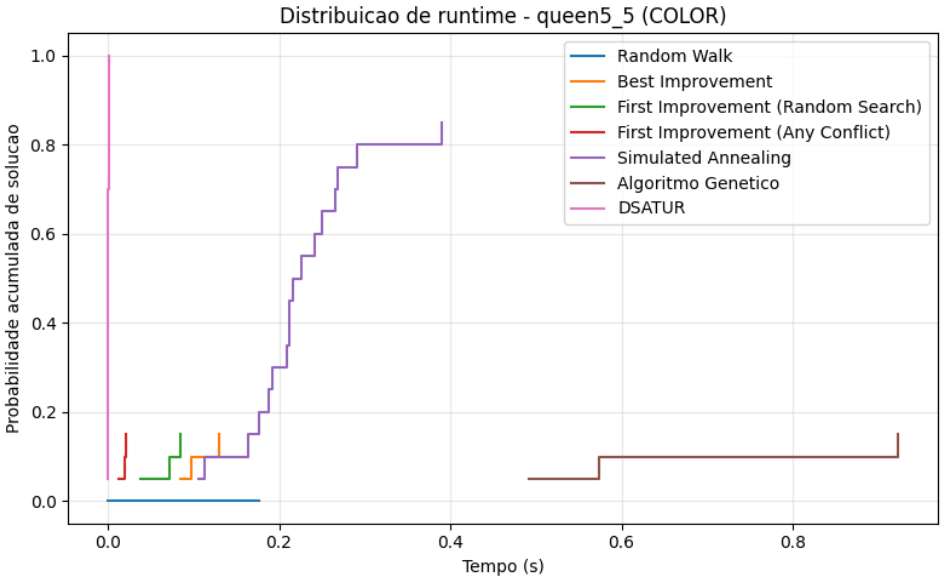

Figura 3: Runtime distribution – `queen5_5`.

5 Conclusões

As quatro tarefas do Trabalho Prático 2 foram cumpridas integralmente:

1) Implementamos as heurísticas RW, BI, FI-RS, FI-AC, SA e GA, além de DSATUR.

2) Avaliamos cada heurística no grafo exemplo e em duas instâncias COLOR.

3) Produzimos gráficos de runtime distribution com 20 execuções por heurística, incluindo o DSATUR.

4) Comparámos as heurísticas em termos de robustez e desempenho, destacando SA e DSATUR como abordagens mais confiáveis na instância densa.

Referências

[1] S. Russell e P. Norvig, *Artificial Intelligence: A Modern Approach*, 4ª ed., Pearson, 2021.

[2] D. Brélaz, "New methods to color the vertices of a graph," *Communications of the ACM*, vol. 22, n. 4, 1979, pp. 251–256.

[3] H. Hoos e T. Stützle, *Stochastic Local Search: Foundations and Applications*, Elsevier, 2004.