**Universidade Federal de Minas Gerais**

Engenharia de Sistemas

# Relatório do Trabalho Prático II - Coloração em Grafos

**Fundamentos de Inteligência Artificial**

**Professores:** Cristiano Castro e João Pedro Campos

**Alunos:**
Áquila Oliveira Souza — 2021019327
Arthur Jorge — 2022055718
Felippe Veloso Marinho — 2021072260
Jefferson Pereira de Souza — 2022099049
Josoé Santos Queiroz — 2019026982

Belo Horizonte, MG
28 de outubro de 2025

# Sumário

# 1 Introdução

A coloração de grafos é um problema clássico da teoria dos grafos com diversas aplicações práticas, como na alocação de frequências em redes sem fio, escalonamento de tarefas e planejamento de horários. O objetivo é atribuir cores aos vértices de um grafo de forma que vértices adjacentes não compartilhem a mesma cor, minimizando o número total de cores utilizadas.

Neste relatório, apresentamos uma solução para o problema de coloração de grafos, modelando-o formalmente como um grafo em que as arestas representam restrições binárias entre os vértices.

Além disso, detalhamos as heurísticas utilizadas para resolver o problema, discutindo suas abordagens teóricas e implementações práticas. Também analisamos as decisões tomadas durante o desenvolvimento e os resultados obtidos.

O documento está organizado da seguinte forma: na Seção 2, apresentamos o problema de coloração de grafos; na Seção 3, discutimos as heurísticas teóricas; na Seção 4, detalhamos suas implementações; e, por fim, na Seção 5, apresentamos as conclusões e possíveis melhorias futuras.

# 2 Problema da Coloração de Grafos

O problema da coloração de grafos consiste em associar uma cor a cada vértice de um grafo $G = (V, E)$, de modo que dois vértices adjacentes $u, v \in V$ não possuam a mesma cor. O desafio é minimizar o número total de cores utilizadas, conhecido como número cromático do grafo. Trata-se de um problema NP-difícil, o que motiva o uso de heurísticas e metaheurísticas para obtenção de soluções aproximadas em tempo viável.

# 3 Heurísticas — Fundamentação Teórica

## 3.1 Random Walk (RW)

O *Random Walk* é uma heurística de busca estocástica que explora o espaço de soluções movendo-se aleatoriamente entre colorações possíveis. O processo inicia-se com uma coloração aleatória e, em seguida, ajusta-se gradualmente os conflitos — isto é, os casos em que vértices adjacentes possuem a mesma cor. Esse procedimento é repetido até que se encontre uma solução viável ou que seja atingido um critério de parada.

## 3.2 Best Improvement (BI)

O *Best Improvement*, por ser heurística de busca local que busca a "melhor melhoria"possível em cada iteração, inicia-se de uma solução inicial, depois o algoritmo avalia todas as alterações viáveis e seleciona aquela que proporciona o maior ganho. No contexto da coloração de grafos, o objetivo é reduzir o número de conflitos (vértices adjacentes com a mesma cor) ou diminuir o total de cores utilizadas.

## 3.3 First Improvement with Random Local Search (FI-RS)

O *First Improvement with Random Local Search* (FI-RS) é uma metaheurística de busca local que combina o *Random Local Search* e o *First Improvement*. O primeiro realiza pequenas alterações aleatórias em uma solução, enquanto o segundo aceita a primeira melhoria encontrada, sem necessidade de examinar todas as opções. Aplicado à coloração de grafos, o FI-RS busca uma coloração com o menor número de conflitos de forma eficiente, evitando uma exploração exaustiva do espaço de busca.

## 3.4 First Improvement with Any Conflict (FI-AC)

A partir de uma solução inicial, possivelmente com conflitos, o algoritmo tenta melhorá-la iterativamente. A estratégia *First Improvement* implica aceitar imediatamente o primeiro movimento que melhora a solução, sem buscar o melhor global. Já o termo *Any Conflict* indica que qualquer vértice envolvido em conflito pode ser escolhido aleatoriamente para modificação. No problema de coloração de grafos, o FI-AC aprimora progressivamente a coloração até eliminar os conflitos entre vértices adjacentes.

## 3.5 Simulated Annealing (SA)

O *Simulated Annealing* (SA) é uma metaheurística de busca estocástica inspirada no processo físico de recozimento térmico de metais. Nesse processo, o material é aquecido e resfriado lentamente para alcançar uma configuração estável de mínima energia. Analogamente, o SA aceita piores soluções no início (alta temperatura) e torna-se mais seletivo à medida que a "temperatura" diminui, buscando escapar de ótimos locais e aproximar-se de uma solução globalmente ótima. Na coloração de grafos, o SA visa reduzir gradualmente os conflitos de coloração, aceitando ocasionalmente soluções piores para explorar melhor o espaço de busca.

### 3.6 Algoritmo Genético (GA)

O *Algoritmo Genético* (GA) é uma metaheurística inspirada nos princípios da evolução natural, proposta originalmente por Holland (1975). Ele baseia-se em mecanismos biológicos como seleção, cruzamento e mutação para evoluir uma população de soluções ao longo das gerações.

No contexto do problema de coloração de grafos, cada indivíduo da população representa uma coloração possível, onde os genes correspondem às cores atribuídas aos vértices. O processo evolutivo busca minimizar o número de conflitos e reduzir o total de cores utilizadas.

Os Algoritmos Genéticos (AG) são técnicas de otimização inspiradas no processo de seleção natural. Eles trabalham com uma população de soluções candidatas, que evoluem ao longo do tempo através de operadores genéticos como seleção, cruzamento (crossover) e mutação.

## 4 Heurísticas - Implementação

### 4.1 Random Walk (RW)

Por ser uma heurística de busca estocástica que explora o espaço de soluções movendo-se aleatoriamente entre as colorações possíveis, o processo se inicia com uma coloração aleatória e, em seguida, ajusta-se gradualmente os conflitos. Esse procedimento se repete até que não existam mais conflitos (uma coloração válida) ou que o limite de iterações seja atingido.

**Algorithm 1** Random Walk (RW)

**Require:** Grafo $G$, Coloração inicial $C_{\text{inicial}}$, Conjunto de cores $\mathcal{K}$, Máx. iterações $I_{\max}$
**Ensure:** Coloração $C$ com o menor número de conflitos

$C \leftarrow C_{\text{inicial}}$
$conflitos \leftarrow \text{ContarConflitos}(G, C)$
**for** $i = 1$ to $I_{\max}$ **do**
    $v \leftarrow \text{EscolherAleatoriamente}(\text{Vértices}(G))$
    $k_{\text{novo}} \leftarrow \text{EscolherAleatoriamente}(\mathcal{K})$
    $C_{\text{novo}} \leftarrow C$
    $C_{\text{novo}}[v] \leftarrow k_{\text{novo}}$
    $conflitos_{\text{novo}} \leftarrow \text{ContarConflitos}(G, C_{\text{novo}})$
    **if** $conflitos_{\text{novo}} < conflitos$ **then**
        $C \leftarrow C_{\text{novo}}$
        $conflitos \leftarrow conflitos_{\text{novo}}$
        **if** $conflitos = 0$ **then**
            **break**
        **end if**
    **end if**
**end for**
**return** $C, conflitos$

## 4.2 Best Improvement (BI)

O *Best Improvement* é uma heurística de busca local que busca a melhor melhoria possível em cada iteração. A partir de uma solução inicial, o algoritmo avalia todas as alterações viáveis dentro de uma vizinhança definida e seleciona aquela que proporciona o maior ganho. Como o objetivo é reduzir o número de conflitos (vértices adjacentes com a mesma cor), iteramos através de uma busca exaustiva nos dois loops verificando se há uma nova coloração com menos conflitos até que não haja mais.

**Algorithm 2** Best Improvement (BI)

**Require:** Grafo $G$, Coloração inicial $C_{\text{inicial}}$, Conjunto de cores $\mathcal{K}$, Máx. iterações $I_{\max}$
**Ensure:** Coloração $C$ com o menor número de conflitos

$C \leftarrow C_{\text{inicial}}$
$conflitos \leftarrow \text{ContarConflitos}(G, C)$
**for** $i = 1$ to $I_{\max}$ **do**
    $C_{\text{melhor}} \leftarrow C$
    $melhor\_melhora \leftarrow 0$
    **for all** vértices $v \in \text{Vértices}(G)$ **do**
        **for all** cores $k \in \mathcal{K}$ **do**
            $C_{\text{temp}} \leftarrow C$
            $C_{\text{temp}}[v] \leftarrow k$
            $conflitos_{\text{temp}} \leftarrow \text{ContarConflitos}(G, C_{\text{temp}})$
            $melhora \leftarrow conflitos - conflitos_{\text{temp}}$
            **if** $melhora > melhor\_melhora$ **then**
                $melhor\_melhora \leftarrow melhora$
                $C_{\text{melhor}} \leftarrow C_{\text{temp}}$
            **end if**
        **end for**
    **end for**
    **if** $melhor\_melhora = 0$ **then**
        **break** ▷ Ótimo local
    **end if**
    $C \leftarrow C_{\text{melhor}}$
    $conflitos \leftarrow conflitos - melhor\_melhora$
    **if** $conflitos = 0$ **then**
        **break**
    **end if**
**end for**
**return** $C, conflitos$

## 4.3 First Improvement with Random Local Search (FI-RS)

O *First Improvement with Random Local Search* (FI-RS) é uma metaheurística de busca local que combina o *Random Local Search* e o *First Improvement*. O algoritmo seleciona aleatoriamente um vértice e atribui-lhe uma cor aleatória. Este novo estado é então avaliado. O movimento é aceito se a nova coloração for estritamente melhor (tiver menos conflitos) que a co-

loração atual, aplicando o critério *First Improvement*. O código do FI-RS implementa essa lógica iniciando com uma coloração e contando os conflitos. A cada iteração, um vértice e uma nova cor são escolhidos aleatoriamente (*Random Local Search*). Se esta nova coloração resultar em menos conflitos (*First Improvement*), ela é imediatamente aceita e o loop continua. O processo é limitado por um número máximo de iterações.

**Algorithm 3** First Improvement with Random Local Search (FI-RS)

**Require:** Grafo $G$, Coloração inicial $C_{\text{inicial}}$, Conjunto de cores $\mathcal{K}$, Máx. iterações $I_{\max}$
**Ensure:** Coloração $C$ com o menor número de conflitos

$C \leftarrow C_{\text{inicial}}$
$conflitos \leftarrow \text{ContarConflitos}(G, C)$
**for** $i = 1$ to $I_{\max}$ **do**
    $v \leftarrow \text{EscolherAleatoriamente}(\text{Vértices}(G))$
    $k_{\text{novo}} \leftarrow \text{EscolherAleatoriamente}(\mathcal{K})$
    $C_{\text{novo}} \leftarrow C$
    $C_{\text{novo}}[v] \leftarrow k_{\text{novo}}$
    $conflitos_{\text{novo}} \leftarrow \text{ContarConflitos}(G, C_{\text{novo}})$
    **if** $conflitos_{\text{novo}} < conflitos$ **then**
        $C \leftarrow C_{\text{novo}}$
        $conflitos \leftarrow conflitos_{\text{novo}}$
        **if** $conflitos = 0$ **then**
            **break**
        **end if**
    **end if**
**end for**
**return** $C, conflitos$

*Nota:* No contexto de coloração de grafos, o FI-RS costuma ficar idêntico ao RW, onde a busca local aleatória é a escolha de $v$ e $k_{\text{novo}}$, e o *First Improvement* é o critério $conflitos_{\text{novo}} < conflitos$.

## 4.4 First Improvement with Any Conflict (FI-AC)

O *First Improvement with Any Conflict* (FI-AC) é uma variação da heurística de busca local. A estratégia *First Improvement* implica aceitar imediatamente o primeiro movimento que melhora a solução, sem buscar o melhor global. Já o termo *Any Conflict* indica que qualquer vértice envolvido em conflito pode ser escolhido aleatoriamente para modificação. O código do FI-

AC reflete essa prioridade de busca, onde a cada iteração, apenas um vértice escolhido aleatoriamente entre os *conflitantes* (*Any Conflict*) é avaliado. O algoritmo testa todas as cores disponíveis para este vértice, aceitando o primeiro movimento que reduz o número total de conflitos (*First Improvement*) e interrompendo a busca interna de cores.

**Algorithm 4** First Improvement with Any Conflict (FI-AC)

**Require:** Grafo $G$, Coloração inicial $C_{\text{inicial}}$, Conjunto de cores $\mathcal{K}$, Máx. iterações $I_{\max}$
**Ensure:** Coloração $C$ com o menor número de conflitos

$C \leftarrow C_{\text{inicial}}$
$(conflitos, V_{\text{conflito}}) \leftarrow \text{RetornarConflitos}(G, C)$
**if** $conflitos = 0$ **then**
    **return** $C, conflitos$
**end if**
**for** $i = 1$ to $I_{\max}$ **do**
    $v \leftarrow \text{EscolherAleatoriamente}(V_{\text{conflito}})$
    **for all** cores $k \in \mathcal{K}$ **do**
        $C_{\text{novo}} \leftarrow C$
        $C_{\text{novo}}[v] \leftarrow k$
        $(conflitos_{\text{novo}}, V_{\text{novo\_conflito}}) \leftarrow \text{RetornarConflitos}(G, C_{\text{novo}})$
        **if** $conflitos_{\text{novo}} < conflitos$ **then**
            $C \leftarrow C_{\text{novo}}$
            $conflitos \leftarrow conflitos_{\text{novo}}$
            $V_{\text{conflito}} \leftarrow V_{\text{novo\_conflito}}$
            **if** $conflitos = 0$ **then**
                **break** ▷ Solução encontrada
            **end if**
            **break** ▷ Aceita a primeira melhoria
        **end if**
    **end for**
    **if** $conflitos = 0$ **then**
        **break**
    **end if**
**end for**
**return** $C, conflitos$

## 4.5 Simulated Annealing (SA)

O *Simulated Annealing* (SA) é uma meta-heurística baseada no processo de recozimento de metais, onde a temperatura é lentamente reduzida para permitir que o material atinja um estado de energia mínima. Em problemas de otimização, ele é parecido com o First Improvement, mas aceita soluções piores (com mais conflitos) com uma probabilidade que é função do parâmetro de temperatura (que decai com o tempo) e da diferença entre o número de conflitos entre a coloração atual e a nova coloração. Essa aceitação de passos "ruins"permite que o algoritmo escape de ótimos locais. O código do SA implementa o mecanismo de recozimento utilizando um parâmetro de temperatura ($T$) que decai a cada iteração. A cada passo, uma mudança de cor é proposta aleatoriamente em um vértice, e a diferença de conflitos ($\Delta E$) é calculada. Se $\Delta E < 0$ (melhoria), a mudança é aceita. Caso contrário (piora), a aceitação é estocástica, determinada por uma função de probabilidade $P$ que depende de $\Delta E$ e da temperatura $T$ atual, garantindo a exploração no início e a explotação no final.

**Algorithm 5** Simulated Annealing (SA)

**Require:** Grafo $G$, Coloração inicial $C_{\text{inicial}}$, Conjunto de cores $\mathcal{K}$, Máx. iterações $I_{\max}$, Temperatura inicial $T_0$
**Ensure:** Coloração $C$ com o menor número de conflitos

$C \leftarrow C_{\text{inicial}}$
$conflitos \leftarrow \text{ContarConflitos}(G, C)$
$T \leftarrow T_0$
**for** $i = 1$ to $I_{\max}$ **do**
    $v \leftarrow \text{EscolherAleatoriamente}(\text{Vértices}(G))$
    $k_{\text{novo}} \leftarrow \text{EscolherAleatoriamente}(\mathcal{K})$
    $C_{\text{novo}} \leftarrow C$
    $C_{\text{novo}}[v] \leftarrow k_{\text{novo}}$
    $conflitos_{\text{novo}} \leftarrow \text{ContarConflitos}(G, C_{\text{novo}})$
    $\Delta E \leftarrow conflitos_{\text{novo}} - conflitos$ ▷ Diferença de "energia"
    **if** $\Delta E < 0$ **then** ▷ Melhoria
        $C \leftarrow C_{\text{novo}}$
        $conflitos \leftarrow conflitos_{\text{novo}}$
    **else**
        $P \leftarrow \text{ProbabilidadeAceitacao}(\Delta E, T)$
        **if** $\text{Aleatorio}(0, 1) < P$ **then**
            $C \leftarrow C_{\text{novo}}$ ▷ Aceita piora estocasticamente
            $conflitos \leftarrow conflitos_{\text{novo}}$
        **end if**
    **end if**
    **if** $conflitos = 0$ **then**
        **break**
    **end if**
    $T \leftarrow \text{DecairTemperatura}(T, i)$
**end for**
**return** $C, conflitos$

## 4.6 Algoritmo Genético (GA)

O *Algoritmo Genético* (GA) é uma meta-heurística de busca global inspirada na evolução biológica. Ele evolui um conjunto de soluções candidatas, denominado população. A cada passo (geração), as soluções atuais interagem entre si através de operadores genéticos (seleção, recombinação (crossover) e mutação) para produzir uma nova população, buscando a convergência para soluções de alta qualidade. No problema de coloração, cada indivíduo (cromossomo) na população representa uma coloração completa do grafo. O

código do GA inicia com a geração de uma população de colorações candidatas. O processo é iterado por um número máximo de gerações, onde a cada ciclo, pares de pais são selecionados. Estes pais geram dois filhos através da recombinação (*crossover*), e uma mutação aleatória é aplicada aos filhos. A nova população substitui a antiga, e a melhor solução encontrada até o momento ($C_{\text{melhor}}$) é atualizada, guiando a evolução para colorações com menos conflitos.

**Algorithm 6** Algoritmo Genético (GA) para Coloração

**Require:** Grafo $G$, Conjunto de cores $\mathcal{K}$, Tamanho da população $N$, Máx. gerações $G_{\max}$
**Ensure:** Melhor coloração $C_{\text{melhor}}$

$\mathcal{P} \leftarrow \text{GerarPopulacaoInicial}(N, G, \mathcal{K})$
$C_{\text{melhor}} \leftarrow \text{MelhorSolucao}(\mathcal{P})$
**for** $g = 1$ to $G_{\max}$ **do**
    $\mathcal{P}_{\text{nova}} \leftarrow \emptyset$
    **for** $j = 1$ to $N/2$ **do**
        $P_1, P_2 \leftarrow \text{Selecao}(\mathcal{P})$ ▷ Torneio, Roleta, etc.
        $F_1, F_2 \leftarrow \text{Crossover}(P_1, P_2)$
        $F_1 \leftarrow \text{Mutacao}(F_1, \mathcal{K})$
        $F_2 \leftarrow \text{Mutacao}(F_2, \mathcal{K})$
        $\mathcal{P}_{\text{nova}} \leftarrow \mathcal{P}_{\text{nova}} \cup \{F_1, F_2\}$
    **end for**
    $\mathcal{P} \leftarrow \mathcal{P}_{\text{nova}}$
    $C_{\text{atual\_melhor}} \leftarrow \text{MelhorSolucao}(\mathcal{P})$
    **if** $\text{Avaliar}(C_{\text{atual\_melhor}}) < \text{Avaliar}(C_{\text{melhor}})$ **then**
        $C_{\text{melhor}} \leftarrow C_{\text{atual\_melhor}}$
    **end if**
    **if** $\text{Avaliar}(C_{\text{melhor}}) = 0$ **then**
        **break**
    **end if**
**end for**
**return** $C_{\text{melhor}}, \text{Avaliar}(C_{\text{melhor}})$

## 4.7 DSATUR

O DSATUR (Degree of Saturation) é uma heurística gulosa para o problema de coloração de grafos que prioriza a coloração dos vértices com maior grau de saturação, ou seja, aqueles que possuem o maior número de cores diferentes já atribuídas aos seus vizinhos. Essa heurística é particularmente eficaz para

o problema de coloração., pois ao focar nos vértices mais "constrangidos", ela tende a minimizar o número total de cores necessárias para uma coloração válida.

A implementação do DSATUR envolve os seguintes passos principais:

- Os vértices do grafo são ordenados com base em seu grau de saturação.
- O vértice mais saturado é selecionado para a coloração.
- A cor mais baixa possível é atribuída ao vértice selecionado, garantindo que não haja conflitos com os vizinhos já coloridos.
- O processo é repetido até que todos os vértices estejam coloridos.

# 5 Experimentos

O objetivo dos experimentos é avaliar o desempenho das heurísticas implementadas na resolução do problema de coloração de grafos. Para isso, realizamos uma série de 30 execuções para cada heurística, utilizando diferentes instâncias do problema. As instâncias seleciondas foram a fornecida no enunciado da tarefa e outras duas instâncias.

Cada execução foi monitorada quanto ao tempo de processamento e a porcentagem de instâncias resolvidas. Neste experimento não limitamos o número de cores mas limitamos o tempo de execução para 2 segundos por execução.

As instâncias utilizadas nos experimentos foram:

- **default**: A instância padrão fornecida no enunciado da tarefa, servindo como base para comparação entre as heurísticas.
- **myciel.5.col**: Uma instância maior com 47 vértices, representando um desafio mediano para as heurísticas.
- **queen9-9.col**: A maior instância utilizada, com 81 vértices e 2112 arés-tas, testando a escalabilidade e eficiência das heurísticas em problemas maiores.

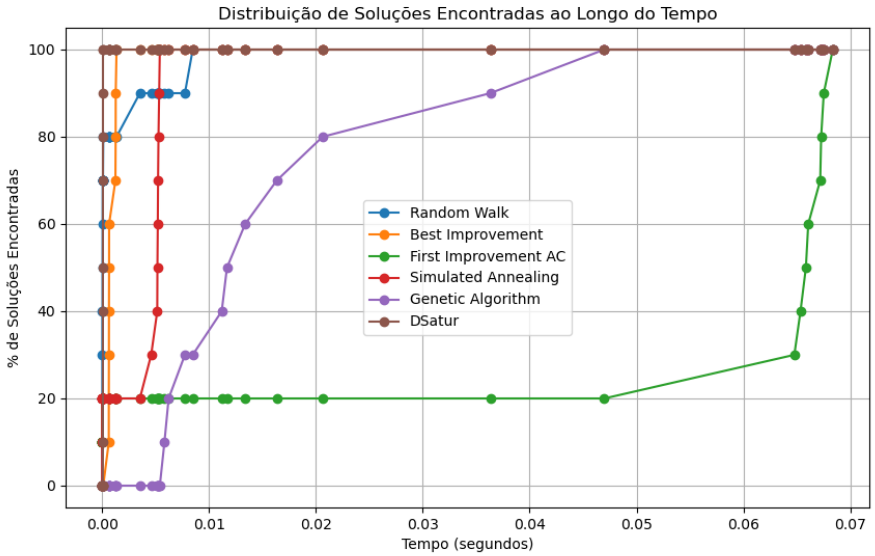


Figura 1: Experimento com a instância default.

Neste experimento, podemos observar que a heurística DSATUR se destacou significativamente, alcançando uma taxa de resolução de 100% das instância quase instantaneamente, com um tempo médio de execução de apenas 0.01 segundos. Isso indica que o DSATUR é altamente eficiente para a instância default, conseguindo encontrar soluções ótimas rapidamente.

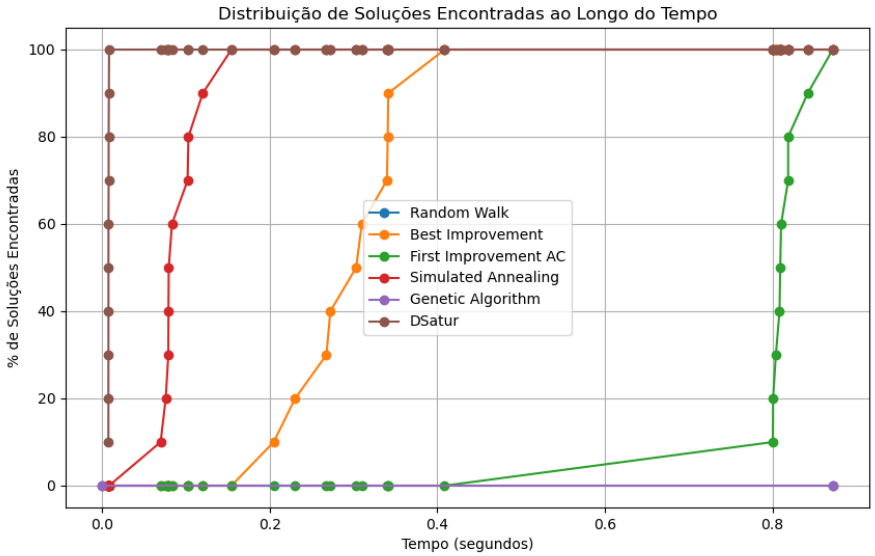


Figura 2: Experimento com a instância myciel.5.col.

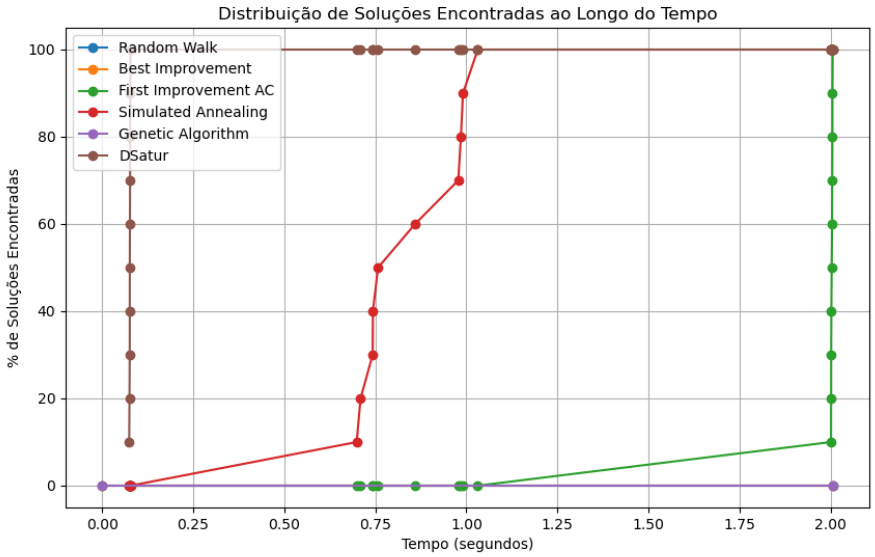


Figura 3: Experimento com a instância queen9-9.col.

Nos dois exeperimentos subsequentes, com as instâncias myciel.5.col e queen9-9.col, alguns algorítmos não encontraram soluções viáveis dentro do

limite de tempo estabelecido. E por isto não aparecem nos gráficos. Esses resultados ressaltam a importância de considerar a complexidade do problema e a adequação das heurísticas utilizadas para diferentes instâncias.

Nos experimentos com instâncias maiores, como myciel.5.col e queen9-9.col, observamos que a heurística DSATUR continuou a se destacar, mantendo uma taxa de resolução de 100% em ambas as instâncias. Isso demonstra a robustez e eficiência do DSATUR mesmo em problemas mais complexos.

# 6 Conclusão

O trabalhou explorou o problema NP-difícil da coloração de grafos por meio de implementações de diversas heurísticas de busca local e metaheurísticas como (RW, BI, FI-RS, FI-AC, SA, GA) além da heurística gulosa DSATUR. O objetivo principal foi complementar a implementação dada como referência e analisar a adequação e eficiência dessas abordagens em diferentes instâncias de grafos.

Em relação aos resultados experimentais, a heurística gulosa DSATUR demonstrou ser, de longe, o método mais robusto e eficiente entre os avaliados. Em todas as três instâncias testadas (Default, myciel.5.col e queen9-9.col), o DSATUR atingiu uma taxa de resolução de 100% das execuções, com um tempo médio de execução quase instantâneo (e.g., 0.01 segundos para a instância **default**). Este desempenho superior é inerente à natureza gulosa do DSATUR, que toma decisões localmente ótimas (priorizando vértices mais saturados) que tendem a levar rapidamente a uma coloração válida. Em contraste, as heurísticas de busca local (RW, BI, FI-RS, FI-AC) e as metaheurísticas (SA, GA) enfrentaram dificuldades notáveis nas instâncias maiores (**myciel.5.col** e **queen9-9.col**). O fato de "algoritmos não terem encontrado soluções viáveis dentro do limite de tempo"para instâncias de tamanho mediano a grande ressalta o desafio da busca local e global em espaços de solução complexos. Embora essas heurísticas sejam projetadas para otimizar o número de conflitos, sua eficácia é comprometida pelo limite de iterações ou pela dificuldade de escapar de ótimos locais. O desempenho fraco sugere que, para problemas de coloração, a estratégia gulosa de construção (como o DSATUR) é mais eficaz para encontrar rapidamente uma solução válida do que as estratégias de busca e refinamento baseadas na minimização de conflitos.

Futuras melhorias poderiam ser feitas através da integração de uma fase de Busca Local (BI ou FI-AC) após o Algoritmo Genético, utilizando a melhor solução da população como ponto de partida. Outra ideia seria a utilização do DSATUR como método de coloração inicial para todos os algoritmos

de busca, fornecendo um ponto de partida de qualidade superior e reduzindo drasticamente o número inicial de conflitos.

# Referências

# Referências

[1] QEHJA, B.; HAJRIZI, E.; ALI, M. A. A Hybrid Graph-Coloring and Metaheuristic Framework for Dynamic Wireless Sensor Networks. *Preprints*, 2025. Disponível em: https://www.preprints.org/manuscript/202507.0720/v1/download. Acesso em: 26 out. 2025.

[2] BIHANI, O. A Heuristic for Graph Coloring Based on the Ising Model. *Mathematics*, v. 13, n. 18, p. 2976, 2025. Disponível em: https://www.mdpi.com/2227-7390/13/18/2976. Acesso em: 26 out. 2025.

[3] YAKUT, S. A robust and efficient algorithm for graph coloring problem. *Journal of King Saud University - Computer and Information Sciences*, 2025. Disponível em: https://www.sciencedirect.com/science/article/pii/S1110866525000696. Acesso em: 26 out. 2025.

[4] DOKEROGLU, T.; BOGAZ, S.; KESKIN, M. An island-parallel ensemble metaheuristic algorithm for graph coloring problems. *arXiv preprint arXiv:2504.15082*, 2025. Disponível em: https://arxiv.org/html/2504.15082v1. Acesso em: 26 out. 2025.