UNIVERSIDADE FEDERAL DE RORAIMA

CENTRO DE CIÊNCIA E TECNOLOGIA

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

DCC606– ANALISE DE ALGORITMOS– 2025

PROF. DR. HEBERT OLIVEIRA ROCHA

CLEILLYSON OSMAR SOUZA DINIZ DE ALMEIDA

SARAH EVELYN DO VALE DA SILVA

Geração de Cronogramas com Coloração Aproximada de Grafos

BOA VISTA, RR

2025

CLEILLYSON OSMAR SOUZA DINIZ DE ALMEIDA

SARAH EVELYN DO VALE DA SILVA

Geração de Cronogramas com Coloração Aproximada de Grafos

Trabalho da disciplina de Análise de Algoritmos do ano de 2025.1 apresentado à Universidade Federal de Roraima do curso de Bacharelado em ciência da computação.

Docente: Prof. Dr. Hebert O. Rocha

BOA VISTA, RR

2025

LISTA DE ABREVIATURAS E SIGLAS

DSATUR *Degree of saturation*

SUMÁRIO

1. Introdução

Coloração de grafos é uma técnica que consiste em aplicar cores aos vértices seguindo a regra de que dois vértices ligados por uma aresta não podem ter a mesma cor, a tarefa é usar a menor quantidade de cores, coloração de grafos pode ser usada em várias áreas mas o trabalho vai focar em resolver os problemas de conflitos em gerações de cronogramas.

2. Algoritmos

Foram implementados 2 algoritmos, um algoritmo guloso e um algoritmo DSatur os dois tem a mesma entrada. A primeira entrada é N que é o numero de vértices, as próximas entradas são U e V e são as arestas.

2.1. Algoritmo Guloso

Seu funcionamento é simples ele apenas checa a menor cor disponível, colore o vértice com essa cor e passa pro próximo, por percorrer ordenadamente o os vértices seus resultados são dependentes da ordem de inseção ele não vai garantir uma solução ótima mas é eficiente.

Sua complexidade é $O(V^2 + E)$ V é a quantidade de vértices e E é a quantidade de arestas.

2.1.1. Pseudocódigo

```
ALGORITMO ColoracaoGulosa(G: Grafo)
ENTRADA: G - grafo não direcionado com V vértices
SAÍDA: resultado[] - array com a cor de cada vértice

INÍCIO
    // Inicialização
    resultado ← array de tamanho G.numVertices inicializado com -1
    disponivel ← array de tamanho G.numVertices de booleanos

    // Colorir o primeiro vértice com cor 0
    resultado[0] ← 0

    // Colorir os vértices restantes
    PARA u ← 1 ATÉ G.numVertices-1 FAÇA

        // Marcar todas as cores como disponíveis
        PARA i ← 0 ATÉ G.numVertices-1 FAÇA
            disponivel[i] ← VERDADEIRO
        FIM PARA

        // Marcar cores dos vizinhos como indisponíveis
        atual ← G.listaAdjacencia[u]
        ENQUANTO atual ≠ NULO FAÇA
            vizinho ← atual.vertice
            SE resultado[vizinho] ≠ -1 ENTÃO
                disponivel[resultado[vizinho]] ← FALSO
            FIM SE
            atual ← atual.proximo
```

```
        FIM ENQUANTO

        // Encontrar primeira cor disponível
        cor ← 0
        ENQUANTO cor < G.numVertices E disponivel[cor] = FALSO FAÇA
            cor ← cor + 1
        FIM ENQUANTO

        // Atribuir a cor ao vértice
        resultado[u] ← cor

    FIM PARA

    RETORNAR resultado
FIM
```

2.2. Algoritmo Dsatur

Esse algoritmo também tem uma abordagem guloso mas é considerado um pouco mais sofistica pois além de usar apenas o grau ou a posição do vértice, também usa o conceito de grau de saturação. O grau de saturação é definido como o numero de cores diferentes atribuídas aos seus vizinhos.

O primeiro vértice que ele escolhe é o de maior grau e atribui a ele a primeira cor disponível. Depois vai buscar sempre o que tiver o maior grau de saturação e usa o grau do vértice como critério de desempate. E já que é um algoritmo guloso ele sempre vai atribuir a menor cor disponível.

2.2.1. Pseudocódigo

```
ALGORITMO DSatur_Coloracao_Grafo(G)
ENTRADA: Grafo G = (V, E) com n vértices
SAÍDA: Coloração válida do grafo usando o menor número de cores possível

INÍCIO
    // Inicialização
    cores[1..n] ← -1          // Nenhum vértice colorido inicialmente
    colorido[1..n] ← FALSO    // Marca vértices já coloridos
    num_cores ← 0             // Contador de cores utilizadas

    // PASSO 1: Escolher vértice inicial (maior grau)
    v_inicial ← vértice com maior grau em G
    cores[v_inicial] ← 0
    colorido[v_inicial] ← VERDADEIRO
    num_cores ← 1
    vertices_restantes ← n - 1

    // PASSO 2: Colorir vértices restantes usando critério DSatur
    ENQUANTO vertices_restantes > 0 FAÇA

        // Encontrar próximo vértice pelo critério DSatur
        max_saturacao ← -1
        max_grau ← -1
        proximo_vertice ← -1
```

```
        PARA CADA v ∈ V FAÇA
            SE colorido[v] = FALSO ENTÃO
                sat ← Calcular_Grau_Saturacao(v, G, cores)
                grau ← Calcular_Grau(v, G)

                SE (sat > max_saturacao) OU
                  (sat = max_saturacao E grau > max_grau) ENTÃO
                    max_saturacao ← sat
                    max_grau ← grau
                    proximo_vertice ← v
                FIM SE
            FIM SE
        FIM PARA

        // Encontrar menor cor disponível para o vértice escolhido
        cores_usadas[0..n] ← FALSO

        PARA CADA u adjacente a proximo_vertice FAÇA
            SE cores[u] ≠ -1 ENTÃO
                cores_usadas[cores[u]] ← VERDADEIRO
            FIM SE
        FIM PARA

        cor_escolhida ← 0
        ENQUANTO cores_usadas[cor_escolhida] = VERDADEIRO FAÇA
            cor_escolhida ← cor_escolhida + 1
        FIM ENQUANTO

        // Colorir o vértice
        cores[proximo_vertice] ← cor_escolhida
        colorido[proximo_vertice] ← VERDADEIRO
        vertices_restantes ← vertices_restantes - 1

        SE cor_escolhida ≥ num_cores ENTÃO
            num_cores ← cor_escolhida + 1
        FIM SE

    FIM ENQUANTO

    RETORNAR (cores, num_cores)
FIM

FUNÇÃO Calcular_Grau_Saturacao(v, G, cores)
ENTRADA: Vértice v, Grafo G, Array de cores
SAÍDA: Grau de saturação do vértice v

INÍCIO
    cores_distintas ← conjunto vazio

    PARA CADA u adjacente a v FAÇA
        SE cores[u] ≠ -1 ENTÃO
            cores_distintas ← cores_distintas ∪ {cores[u]}
        FIM SE
    FIM PARA

    RETORNAR |cores_distintas|
FIM
```

```
FUNÇÃO Calcular_Grau(v, G)
ENTRADA: Vértice v, Grafo G
SAÍDA: Grau do vértice v

INÍCIO
   grau ← 0
   PARA CADA u ∈ V FAÇA
      SE (v,u) ∈ E ENTÃO
         grau ← grau + 1
      FIM SE
   FIM PARA
   RETORNAR grau
FIM
```

3. Avaliação Experimental

Foi usado como teste o a grade curricular do semestre de 2025.2 do curso de ciências da computação. Foram 16 matérias obrigatárias ofertadas no semestre, modelamos e colocamos para os algoritmos calcularem os horários. Ambos os algorítimos chegaram a 5 cores, mas o algoritmo guloso conseguiu agrupar melhor os resultados.

4. CONCLUSÃO

Os dois algoritmos não entregam a opção ótima mas ao menos o DSatur consegue entregar constantemente uma resposta aproximada de ótimo. A principal limitação é a entrada de dados, pois é preciso que as arestas sejam feitas previamente. Isso dificulta para usuários comuns.

REFERÊNCIAS

FRANKNBERGER, F. F.; BRANDL, M.; LEITE, M. ESTUDO SOBRE POSSÍVEIS SOLUÇÕES PARA O PROBLEMA DA COLORAÇÃO DE GRAFOS. Anais da Feira do Conhecimento Tecnológico e Científico , [S. l.], v. 1, n. 25, 2024. Disponível em: https://publicacoes.ifc.edu.br/index.php/fetec/article/view/6165

Karger, D., Motwani, R., Sudan, M. (1998). Approximate Graph Coloring by Semidefinite Programming.