Trabalho Prático: Algoritmo de Welsh-Powell

Cauan Santos Silva
João Paulo Mendonça Andrade

Sumário

1. Introdução
   1.1 O algoritmo da coloração
   1.2 Problema NP
   1.3 Solução heurística

2. Modelagem do problema
   2.1 Representação do Mapa
   2.2 Objetivo da Coloração
   2.3 Complexidade do Problema

3. Discussão do algoritmo
   3.1 Princípios
   3.2 Vantagens e Limitações
   3.3 Aplicações

4. Discussão dos Trabalhos Relacionados
   4.1 Trabalhos Relacionados

5. Apresentação dos Resultados
   5.1 Código do algoritmo
   5.2 Interface Gráfica(em forma de grafo)
   5.3 Interface Gráfica(Mapa)

6. Referências

1. Introdução

1.1 Algoritmo de Coloração de Vértices:

Neste trabalho, abordaremos o problema da coloração de vértices em grafos, cujo objetivo é dividir os vértices em partições de modo que vértices pertencentes à mesma partição não compartilham arestas entre si.

1.2 Complexidade NP:

É importante ressaltar que, até o momento, não existe um algoritmo que possa garantir a obtenção do número mínimo de cores para colorir um grafo qualquer. O problema da coloração pertence à classe NP, o que significa que encontrar uma solução ótima tem complexidade exponencial**.**

2. Modelagem do Problema: Coloração de Mapas da América do Sul e dos Estados da Região Nordeste

Neste tópico, discutiremos a modelagem do problema de coloração de vértices com base na representação da pintura de mapas geográficos, especificamente da

América do Sul e dos estados que compõem a região Nordeste do Brasil. A escolha desses mapas como exemplo prático nos permite visualizar de forma tangível a aplicação deste problema na vida real.

2.1 Representação do Mapa:

Para nosso propósito, consideramos que cada país na América do Sul (ou cada estado no Nordeste) é representado como um vértice em um grafo. Vértices adjacentes no grafo correspondem a países (ou estados) que compartilham uma fronteira. Assim, podemos criar uma representação gráfica que modela as conexões entre essas entidades geográficas.

2.2 Objetivo da Coloração:

O objetivo é atribuir cores aos vértices (países ou estados) de tal forma que vértices adjacentes não compartilhem a mesma cor. Isso se traduz na pintura dos países (ou estados) de um mapa de tal forma que países (ou estados) vizinhos tenham cores diferentes.

2.3 Complexidade do Problema:

A complexidade deste problema torna-se evidente quando se considera que encontrar a menor quantidade de cores necessárias para colorir um mapa sem que países (ou Estados) vizinhos tenham a mesma cor é um desafio computacionalmente caro, e não existe um algoritmo eficiente para solucioná-lo em todos os casos.

3. Discussão do Algoritmo de Welsh-Powell

O algoritmo de Welsh-Powell é uma abordagem heurística que se baseia nos seguintes princípios:

3.1 Princípios:

Ordenação de Vértices: Primeiro, os vértices do grafo são ordenados em ordem decrescente de grau. Isso significa que os vértices com o maior número de arestas adjacentes são colocados no início da lista. Isso ajuda a priorizar a

coloração dos vértices mais "conectados", o que pode, em última análise, reduzir o número total de cores necessárias.

Coloração: Em seguida, os vértices são coloridos um por um, seguindo a ordem estabelecida. Para cada vértice, o algoritmo tenta atribuir a menor cor possível que não seja usada por seus vizinhos já coloridos. Isso significa que, se um vértice não puder ser colorido com uma cor, ele receberá a próxima cor disponível.

Repetição: O processo de coloração é repetido para todos os vértices na lista ordenada.

Resultado: O algoritmo produz uma coloração para o grafo, onde cada vértice recebe uma cor de tal forma que vértices adjacentes não compartilham a mesma cor.

3.2 Vantagens e Limitações:

As vantagens do algoritmo de Welsh-Powell incluem sua simplicidade e eficiência em muitos casos práticos. Ele frequentemente produz colorações com um número relativamente baixo de cores.

No entanto, é importante observar que o algoritmo de Welsh-Powell não garante uma solução ótima para o problema de coloração de vértices. Em alguns casos, ele pode usar mais cores do que o necessário para a coloração ótima. A qualidade da solução depende em grande parte da ordem em que os vértices são considerados e do grafo em si.

3.3 Aplicações:

O algoritmo de Welsh-Powell é frequentemente usado em aplicações práticas, como programação de horários, alocação de recursos e design de circuitos, onde é necessário atribuir recursos (ou cores) de forma eficiente, evitando conflitos entre entidades relacionadas (ou vértices adjacentes). Mesmo que não produza soluções ótimas em todos os casos, é uma ferramenta valiosa para resolver problemas de coloração de vértices de maneira eficaz.

4. Discussão dos Trabalhos Relacionados

4.1 Trabalhos Relacionados :

Nossas referências se concentram em um trabalho que aborda o algoritmo de coloração e suas diversas aplicações. Entre essas aplicações, destacamos a coloração de mapas da América do Sul, dos Estados dos Estados Unidos e de Portugal, bem como a criação de um quebra-cabeça Sudoku e o planejamento de transporte de produtos químicos.

Para a modelagem de nosso trabalho, escolhemos como exemplo a coloração do mapa da América do Sul, substituindo os outros exemplos de coloração mencionados pelo mapa dos estados da região Nordeste. Isso nos permitiu aplicar o conceito de coloração de vértices em um contexto geográfico mais específico e relevante para nossa pesquisa

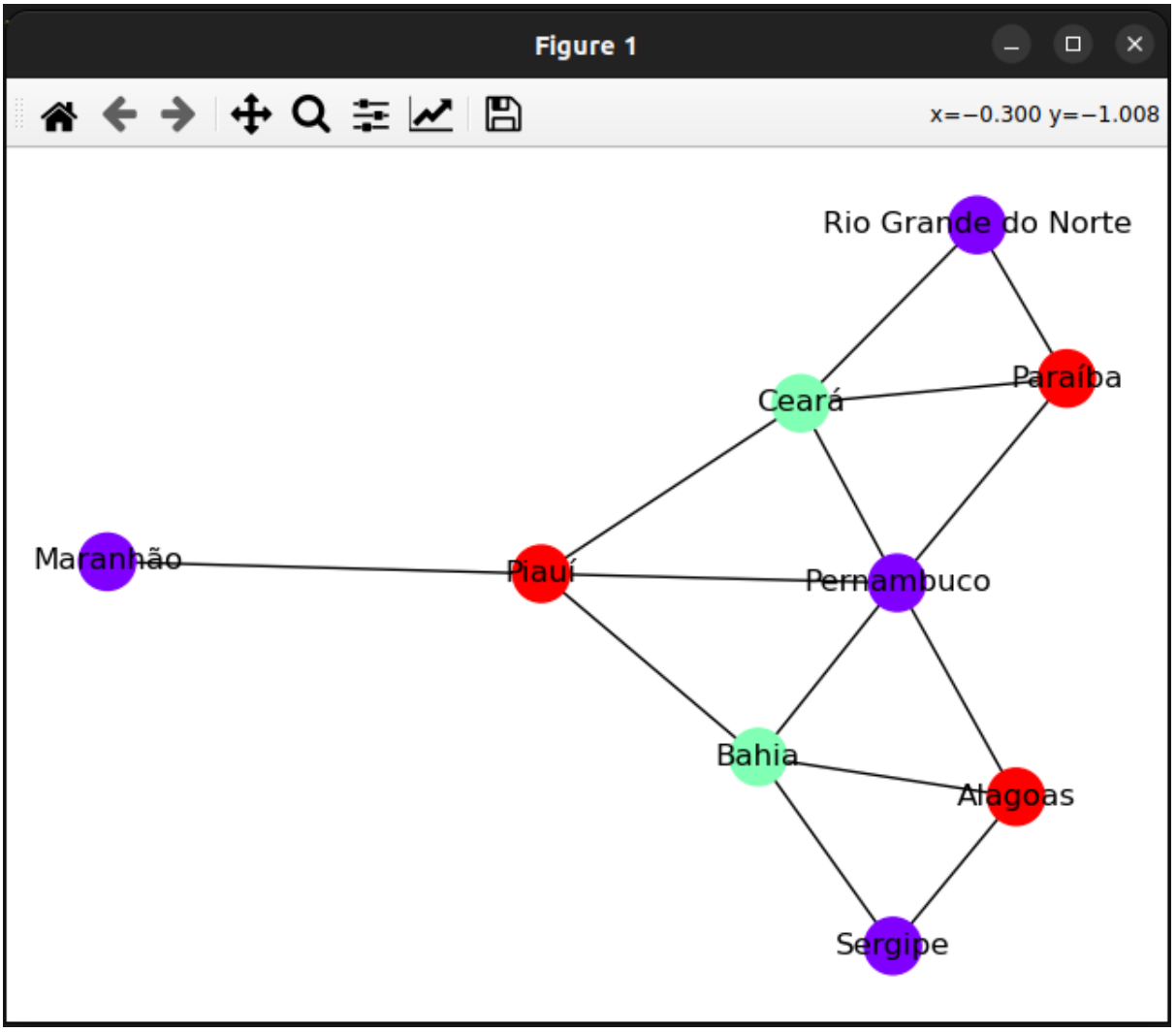
5. Apresentação dos Resultados

5.1. Código do algoritmo:

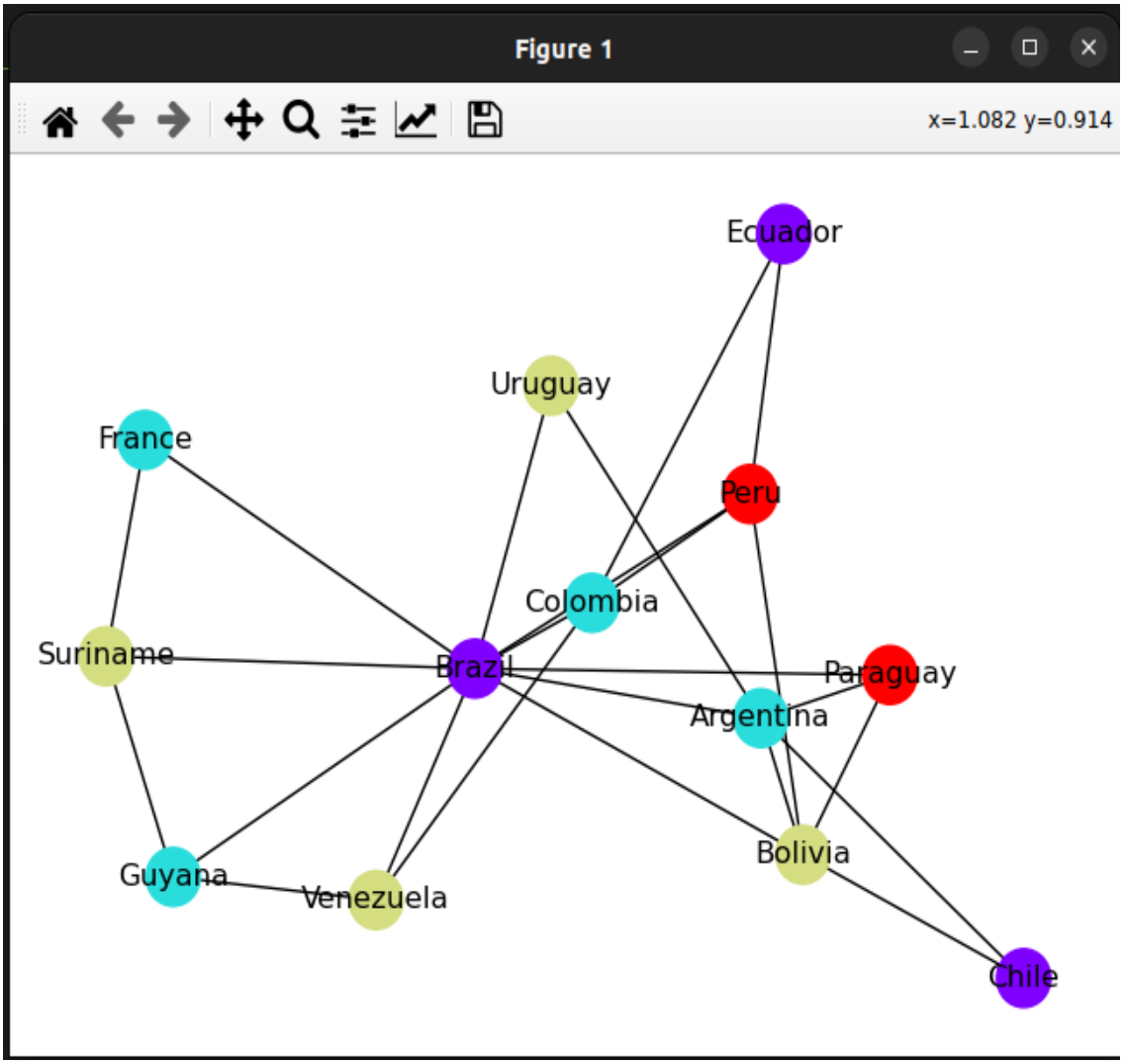
```
def calcula_grau(self, vertice):
    if vertice in self.grafo:
        return self.grafo[vertice].__len__()
    else:
        return 0

def sao_vizinhos(self, vertice_1, vertice_2):
    if vertice_2 in self.grafo[vertice_1]:
        return True
    else:
        return False
def ordena_grau_vertices(self):
    dict_vert = {}
    for vertice in self.grafo:
        vertice_graus = self.calcula_grau(vertice)
        dict_vert[vertice] = vertice_graus
    dict_vert = dict(sorted(dict_vert.items(), key=lambda item: item[1], reverse=True))
    return dict_vert
def imprime_interface(self):
    dicionario = {}
    for chave in self.grafo:
        dicionario[chave] = self.grafo[chave]

    return dicionario
def existe_vizinho_com_aquela_cor(self,dicionario,cor,vertice):
    vertices_ligados=self.retorna_vertices_ligados(vertice)
    ligado=False
    for vert in vertices_ligados:
        if vert  in dicionario[cor]:
            ligado=True
            break
    return ligado

def colorir(self):
    vertices_ordenados=self.ordena_grau_vertices()
    dict_coloracao={}
    cor = 0
    for vertice in vertices_ordenados:
        for i in range(cor+1):
            if i not in dict_coloracao:
                dict_coloracao[i]=[]
            if not self.existe_vizinho_com_aquela_cor(dict_coloracao,i,vertice):
                dict_coloracao[i].append(vertice)
                break
        cor+=1
    return dict_coloracao
```

5.2 Interface Gráfica(em forma de grafo):

Nordeste:

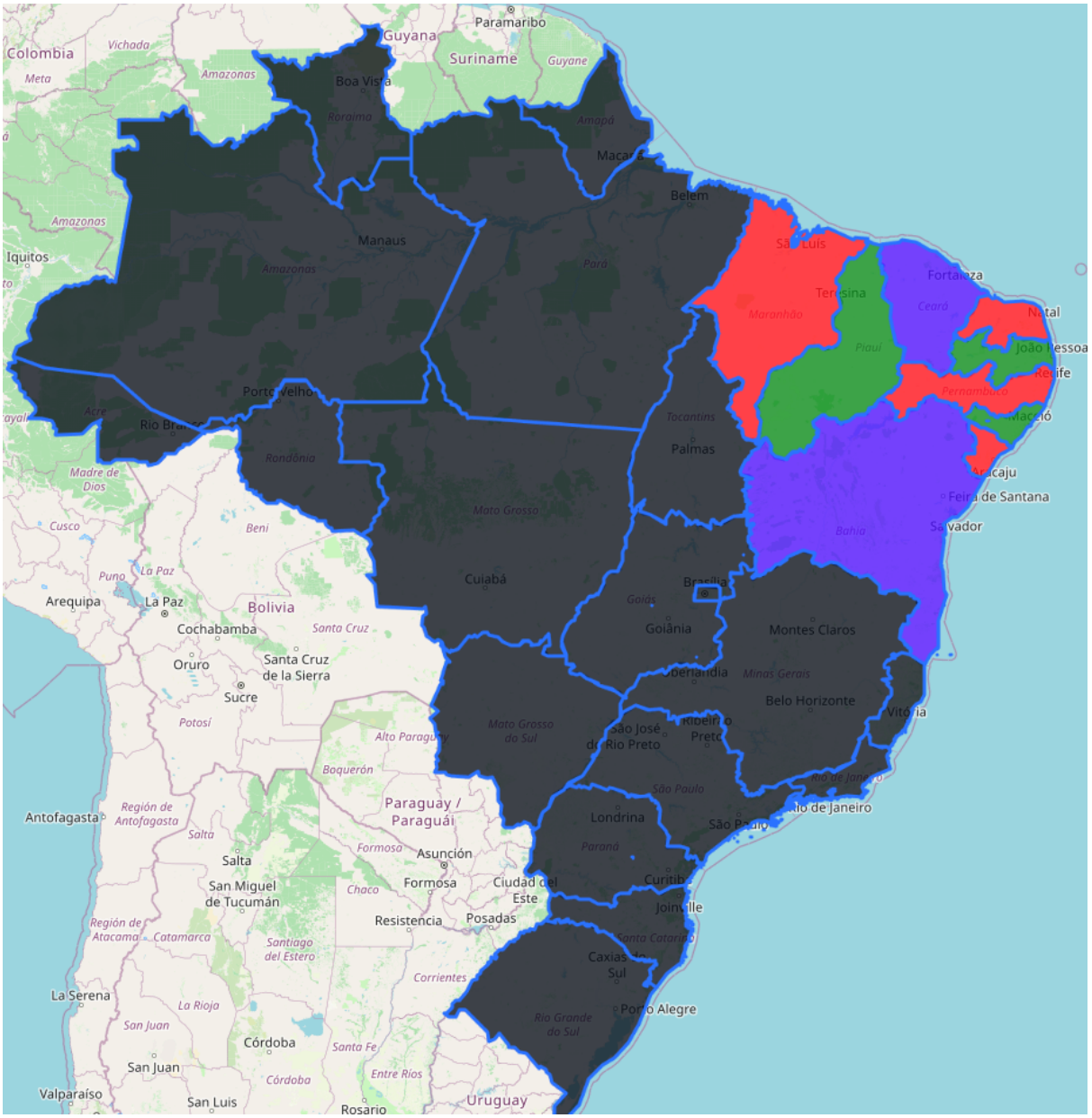


América do Sul:



5.3 Interface Gráfica(Mapa):

Nordeste:



América do Sul:


Caracas
Panamá
Medellín
Venezuela
Colombia
Manaus
Fortaleza
Perú
Lima
Brasil
Salvador
Bolivia
Brasília
Santa
Belo Horizonte
Rio de
Janeiro
Paraguay /
São Paulo
Curitiba
Córdoba
Uruguay
Buenos
Aires
Argentina
South Georg
and the
South Sandwi
Islands

Referências:

https://core.ac.uk/download/pdf/38424385.pdf