Universidade Estadual do Oeste do Paraná
UNIOESTE – Campus de Foz do Iguaçu
Centro de Engenharias e Exatas
Ciência da Computação
Inteligência Artificial

Adrian Fuchs

Alisson Flecha

Projeto I

Relatório Técnico

Foz do Iguaçu

2020/1

1. Introdução

O presente projeto tem como propósito escolher, implementar e comparar ao menos dois algoritmos de busca para colorir o mapa dos estados do Brasil. Para tal, foram escolhidos os seguintes algoritmos de busca:

- Largura;
- Busca com Heurística de Conflitos Mínimos com *Backtracking;*
- Algoritmo C.

Tem-se como requisito a implementação de uma interface gráfica para que seja possível acompanhar a execução dos algoritmos. Além disso, nosso projeto dispõe de um área com um pseudocódigo onde é possível acompanhar passo-a-passo da execução.

1.1. Problema Proposto

O objetivo do *toy-problem* proposto é colorir o mapa dos estados do Brasil, de modo que onde nenhum estado tem a mesma cor que os seus estados vizinhos. Para isso, o algoritmo pode utilizar um leque de até cinco cores.

1.2. Busca em Largura

Esse algoritmo caracteriza-se pela expansão dos próximos nós ser nos estados vizinhos do estado atual. Ou seja, a busca realiza-se de nível em nível de uma árvore.

1.3. Busca com Heurística de Conflitos Mínimos e *Backtracking*

Já o algoritmo de Busca com Heurística de Conflitos Mínimos expande sua fronteira utilizando um conhecimento prévio do escopo do *toy-problem*. O conhecimento prévio utilizado nesse caso é a quantidade de vizinhos de cada estado e se os mesmos já foram pintados anteriormente.

Todavia, durante a fase de testes, foi identificado que muitas vezes o algoritmo encontrava-se preso em um canto do mapa. Para resolver esse problema, foi implementado também o *backtracking* junto a busca para que caso o algoritmo ficasse preso, ele voltaria para os estados ainda não pintados.

1.4. Algoritmo C

Resultado de um bug na implementação do algoritmo de busca em largura, decidiu-se manter no trabalho, pois ele funciona.

O algoritmo funciona da seguinte forma:

1 - Pega a instância do estado que deveria pintar (no caso o estado inicial);

2 - Verifica quais cores já foram usadas nos vizinhos, e pinta com a primeira disponível;

3 - Chama recursivamente a função dando como entrada os vizinhos não pintados.

2. Instalação e execução

Para poder executar o programa, é necessário ter clonado o repositório https://github.com/FlechaAlisson/ia_trab01 e instalado o Nodemon. Como consequência, é necessário ter o gerenciador de pacotes do Node.js. Para instalar o Node.js, é só ir no link: https://nodejs.org/en/download/.

Tendo o Node.js instalado, é só executar o seguinte comando:

```
$npm install -g nodemon
```

Com o Nodemon pronto, a execução deve-se ser feita pelo terminal de comandos do sistema operacional. Estando na pasta do projeto, deve-se executar o seguinte comando:

```
$nodemon server.js
```

Por fim, a aplicação estará rodando no seguinte link "http://localhost/index.html".

3. Materiais e Métodos

Os experimentos foram executados três vezes em duas máquinas diferentes, tendo o Amapá como estado inicial. Os computadores usados no experimento seguem as seguintes configurações:

	Computador I	**Computador II**
CPU / freq. Clock	Ryzen 5 3600 3.59 GHz	Intel I7-7700 HQ 2.8GHz
RAM	16,0 GB	8,0 GB
Placa de vídeo	Radeon RX 5600XT	GTX 1050 TI

Tabela 01: Configuração das máquinas que foram realizados os experimentos.

Além disso, no momento das execuções, o uso da CPU e da placa de vídeo, junto com a memória RAM estavam com o uso:

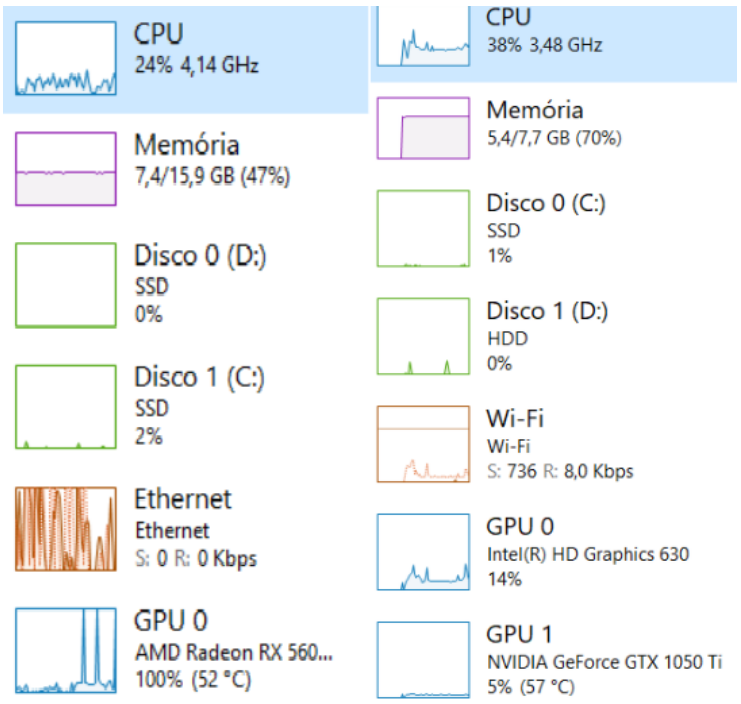

Imagem 01: Gerenciamento de Tarefas do Windows no momento da execução dos experimentos no computador I e II respectivamente. Fonte: Autores.

Por fim, com o objetivo de conseguir mensurar o tempo de execução precisamente, foi desativado o intervalo artificial entre as iterações dos algoritmos.

4. Resultados e Discussão

Tendo isso em mente, segue abaixo um quadro com os desempenho em tempo (microsegundos) de cada algoritmo:

	Busca em Largura	**Conflitos Mínimos com *Backtracking***	**Algoritmo C**
Execução 1	105	105	104
Execução 2	103	107	104
Execução 3	103	104	105
Média	103.67	105.33	104.33
Desvio padrão	0.94	1.24	0.47

Tabela 02: Desempenho de cada algoritmo medidos em microsegundos das execuções no computador I. Fonte: Autores.

	Busca em Largura	**Conflitos Mínimos com *Backtracking***	**Algoritmo C**
Execução 1	105	106	105
Execução 2	104	106	108
Execução 3	106	106	108
Média	105	106	107
Desvio padrão	0.81	0	1.41

Tabela 02: Desempenho de cada algoritmo medidos em microsegundos das execuções no computador II. Fonte: Autores.

A diferença entre os resultados apresentados na Tabela 02 pode ser explicada pelo fato que a Busca com Heurística de Conflitos Mínimos e *Backtracking* precisa checar se o estado escolhido foi visitado anteriormente ou não. Enquanto o algoritmo de Busca em Largura não precisa fazer essa análise para escolher o melhor estado, expandido sempre de forma linear.

Para problemas onde precisam passar por todos os nós de um grafo, a melhor escolha seja o algoritmo mais simples, fazendo valer o Princípio da Navalha de Ockham. Tal Princípio diz que a melhor solução de um problema, é a solução mais simples.

5. Referências Bibliográficas

- Russell, S; Norvig, P. Artificial Intelligence: A Modern Approach, Third Edition;
- LEE, H. D. Estratégias de Busca Não Informada (Cega). [sem ano]. 45 slides;
- LEE, H. D. Estratégias de Busca e Outras Abordagens. [sem ano]. 86 slides;
- LEE, H. D. Introdução à Inteligência Artificial (Parte 1). [sem ano]. 64 slides;
- nodemon. Nodemon.io. Disponível em: <https://nodemon.io/>. Acesso em: 29 Mar. 2021;
- NODE.JS. Download | Node.js. Node.js. Disponível em: <https://nodejs.org/en/download/>. Acesso em: 29 Mar. 2021.