# Randomized Algorithm for Chromatic Number Problem

Renato Alexandre Lourenço Dias, nMec: 98380

***Abstract* - The chromatic number problem, also known as the vertice coloring problem, is one of the fundamental problems in graph theory, and it is linked to various other domains. In this paper, it is explained the randomized algorithm created, analyzed his computational complexity and compared with other strategies of solving the given problem.**

## I. INTRODUCTION

The chromatic number of G is the smallest number of colors needed to properly color the vertices of a graph, such that no two adjacent vertices have the same color.

In the following examples, we can visualize a simple graph with 6 vertices and with a chromatic number of 3, and another graph that also has a chromatic number of 3 despite having 14 vertices.

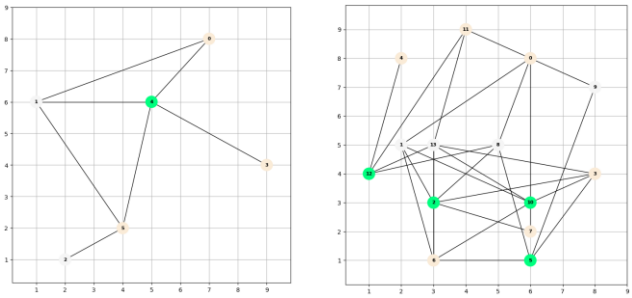

Fig. 1 - Graphs with Chromatic Number of 3.

## II. PROBLEM DESCRIPTION

There is no efficient algorithm available for coloring a graph with minimum number of colors as the problem is a known NP Complete problem. There are approximate algorithms to solve the problem though, such as greedy algorithms, exhaustive argorithms and also randomized algorithms [1].

A randomized algorithm is an algorithm that employs a degree of randomness as part of its logic or procedure [2].

There are some types of randomized algorithms such as algorithms that use the random input so that they always terminate with the correct answer, but where the expected running time is finite (Las Vegas algorithms) and algorithms which have a chance of producing an incorrect result or fail to produce a result either by signaling a failure or failing to terminate (Monte Carlo algorithms).

## III. APPROACH AND IMPLEMENTATION

### *A. Parameters Management*

The program allows the user to pass some arguments that have different actions. The user can set a few parameters that allow for an easier use of the program and will be useful for allowing the operations that will be explained ahead.

It is important to note that the number of nodes is an identifier for the graph and will be used for the graph generation.

```
Usage: python3 main.py
        -h <Shows available arguments>
        -n <Number of Nodes for the Graph to be used for any operation: int>
        -ge <Generate Graph>
        -r <Randomized Coloring>
        -s <Show plot of graphs>
```

Fig. 2 - Program Arguments.

### *B. Graph Management*

Before implementing a solution for the problem it is necessary to be able to manage graphs at will, for that, it was used external libraries to facilitate the process, NetworkX, along with matplotlib.

The program allows generating random undirected graphs, saving graphs, loading graphs and also plotting graphs like in the first project.

### C. Randomized Algorithm

The algorithm consists in assign a random color from a set with a number of colors equals to the number of nodes of the graph (to satisfy the worst case where all nodes get a different color) to a random node until all nodes get a color.

First of all, we have to create 3 lists and a set: a list to store colored nodes, another one to store used colors, which will help us calculating the chromatic number and a third one to check adjacent colors and a set which contains the same number of colors as number of vertices.

Afterwards, from a list with all nodes, we choose one randomly and check which colors can be assigned to that node (for example if the node has an adjacent one already colored, we remove that color from the set with possible colors to assign to the node). Then we choose one color from the set to color the node and after that we remove the node from the nodes list and append it to the list containing the colored nodes and append the color to a list where used colors are stored. This process is repeated until all nodes get colored.

## IV. RESULTS AND DISCUSSION

For the results, several runs (number of nodes of each graph) using randomized strategy were made and noted the execution time, the number of basic operations, such as checking adjacent vertices colors and coloring the vertice, the number of solutions/configurations searched were also noted, and finally the chromatic numbers from the solutions found and the best chromatic number for the graph.

### A. Randomized Search

For the randomized strategy, the results were as the table I, figures 3 and 4 and table II show.

TABLE I
*Results for Randomized Search Strategy of 7 nodes graph*

| Time | Basic Operations | Solutions Searched | Chromatic Number |
|---|---|---|---|
| 0.000s | 33 | 1 | 6 |
| 0.000s | 32 | 1 | 6 |
| 0.000s | 33 | 1 | 6 |
| 0.000s | 39 | 1 | 6 |
| 0.000s | 32 | 1 | 4 |
| 0.000s | 31 | 1 | 4 |
| 0.000s | 33 | 1 | 4 |

For a graph of 7 nodes we obtained graphs with a chromatic number of 6 and graphs with a chromatic number of 4 as we can see above:

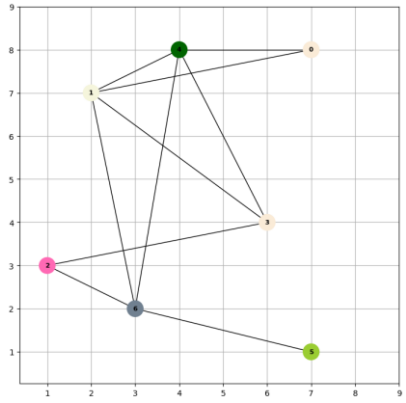

Fig. 3 – Graph with a chromatic number of 6

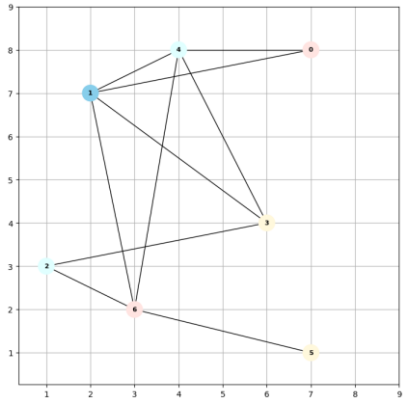

Fig. 4 – Graph with a chromatic number of 4

TABLE II
*Best results for Randomized Search Strategy*

| Nodes | Edges | Time | Basic Operations | Solutions Searched | Chromatic Number |
|---|---|---|---|---|---|
| 2 | 1 | 0.000s | 7 | 1 | 2 |
| 3 | 3 | 0.000s | 10 | 1 | 3 |
| 4 | 4 | 0.000s | 15 | 1 | 3 |
| 5 | 6 | 0.000s | 20 | 1 | 4 |
| 6 | 7 | 0.000s | 26 | 1 | 4 |
| 7 | 10 | 0.000s | 31 | 1 | 5 |
| 8 | 12 | 0.000s | 37 | 1 | 5 |
| 9 | 13 | 0.000s | 52 | 1 | 5 |
| 10 | 11 | 0.000s | 57 | 1 | 6 |
| 11 | 20 | 0.000s | 65 | 1 | 6 |
| 12 | 21 | 0.001s | 77 | 1 | 7 |
| 15 | 28 | 0.000s | 102 | 1 | 9 |
| 20 | 52 | 0.000s | 145 | 1 | 12 |
| 25 | 82 | 0.001s | 220 | 1 | 12 |
| 30 | 114 | 0.001s | 314 | 1 | 17 |
| 35 | 159 | 0.001s | 382 | 1 | 21 |

The results obtained in the first project were the following ones:

TABLE III
*Results for M-Coloring Exhaustive Search Strategy*

| Nodes | Edges | Time | Basic Operations | Solutions Searched | Chromatic Number |
|---|---|---|---|---|---|
| 2 | 1 | 0.000s | 9 | 5 | 2 |
| 3 | 3 | 0.000s | 27 | 15 | 3 |
| 4 | 4 | 0.000s | 14 | 7 | 2 |
| 5 | 6 | 0.000s | 58 | 33 | 3 |
| 6 | 7 | 0.000s | 94 | 54 | 3 |
| 7 | 10 | 0.000s | 67 | 37 | 3 |
| 8 | 12 | 0.000s | 47 | 24 | 3 |
| 9 | 13 | 0.000s | 215 | 119 | 3 |
| 10 | 11 | 0.000s | 54 | 29 | 2 |
| 15 | 28 | 0.002s | 7948 | 3978 | 4 |
| 20 | 52 | 0.035s | 144855 | 69857 | 4 |
| 25 | 82 | 0.057s | 228864 | 108540 | 4 |
| 30 | 114 | 10.25s | 45048061 | 18018582 | 5 |
| 35 | 159 | 22.32s | 88532767 | 35234219 | 5 |

TABLE IV
*Results for Greedy Search Strategy*

| Nodes | Edges | Time | Basic Operations | Solutions Searched | Chromatic Number |
|---|---|---|---|---|---|
| 2 | 1 | 0.000s | 5 | 1 | 2 |
| 3 | 3 | 0.000s | 8 | 1 | 3 |
| 4 | 4 | 0.000s | 11 | 1 | 2 |
| 5 | 6 | 0.000s | 14 | 1 | 4 |
| 6 | 7 | 0.000s | 17 | 1 | 4 |
| 7 | 10 | 0.000s | 20 | 1 | 4 |
| 8 | 12 | 0.000s | 23 | 1 | 3 |
| 9 | 13 | 0.000s | 26 | 1 | 4 |
| 10 | 11 | 0.000s | 29 | 1 | 3 |
| 15 | 28 | 0.000s | 44 | 1 | 5 |
| 20 | 52 | 0.000s | 59 | 1 | 6 |
| 25 | 82 | 0.000s | 74 | 1 | 6 |
| 50 | 309 | 0.001s | 149 | 1 | 9 |
| 100 | 1197 | 0.007s | 299 | 1 | 12 |
| 200 | 4738 | 0.047s | 599 | 1 | 18 |
| 300 | 10552 | 0.156s | 899 | 1 | 25 |
| 400 | 18625 | 0.372s | 1199 | 1 | 31 |

Comparing the results obtained in the first project with the ones obtained with randomized search, we can conlcude that the randomized algorithm created is not so efficient like the exhaustive and greedy ones because, for example, for a graph with 10 nodes the best chromatic number with randomized algorithm is 6 while we have got a chromatic number of 2 with exhaustive search and a chromatic number of 3 with greedy search.

Talking about basic operations and execution times, the randomized search algorithm is faster than exhaustive search and perform less operations but like the other strategies the number of basic operations and the execution time tends to increase exponentially as the number of nodes of the graph increase.

Note that the biggest case we tested for was a graph with 139 nodes due to a matplotlib colors library problem related to the number of colors available.

For this graph, the best chromatic number calculated was 85 and was calculated in 0.03s so for a graph with 1000 nodes I estimate the calculation would take a few seconds and a few hours for a graph with a million nodes.

## REFERENCES

[1] Wikipedia contributors. (2022, September 15). Graph coloring. Wikipedia. https://en.wikipedia.org/wiki/Graph_coloring

[2] Wikipedia contributors. (2022, April 19).Randomized Algorithm. https://en.wikipedia.org/wiki/Randomized_algorithm

[3] J.Madeira, “Intro Randomized Algorithms”

[4] J.Madeira, “Intro Randomized Algorithms II”

[5] J.Madeira, “Intro Randomized Algorithms II”