Analysis of Chromatic Number Calculation Algorithms: Exhaustive Search and Greedy Heuristic

Rafael Curado

***Abstract* –This paper presents a comparative analysis of two algorithms for calculating the chromatic number of graphs: an exhaustive search algorithm and a greedy heuristic algorithm. Analyzes the computational complexity of the developed algortithms as well as some metrics, such as the number of basic operations carried out, the execution times, the number of solutions tested and the precision of the greedy heuristic. The algorithm's performance is evaluated through a series of tests with various graph sizes and edge densities.**

***Resumo* –Este artigo apresenta uma análise comparativa de dois algoritmos para calcular o número cromático de grafos: um algoritmo de pesquisa exaustiva e um algoritmo heurístico guloso. Analisa a complexidade computacional dos algoritmos desenvolvidos bem como algumas métricas, como o número de operações básicas realizadas, os tempos de execução, o número de soluções testadas e a precisão da heurística gananciosa. O desempenho do algoritmo é avaliado através de uma série de testes com diversos tamanhos de gráfico e densidades de arestas.**

***Keywords* –Chromatic Number, Graph Coloring, Exhaustive Search, Greedy Heuristic, Computational Complexity**

***Palavras chave* –Número Cromático, Coloração de Grafos, Pesquisa Exaustiva, Heurística Gulosa, Complexidade Computacional**

I. INTRODUCTION

A graph's chromatic number is the smallest number of colors needed for coloring its vertices so that no two neighboring vertices have the same color. This chromatic number is essential to many applications, such as frequency assignment, scheduling, and map coloring problems. Because this problem is computationally demanding, there are several methods for estimating or accurately determining the chromatic number. In this study, we concentrate on two such approaches: a greedy heuristic and exhaustive search.

II. ALGORITHMS FOR CHROMATIC NUMBER CALCULATION

A. Exhaustive Search

The exhaustive search technique determines the bare minimum of colors needed for a valid coloring by iterating over every possible color configuration and examining each one. Even though this method is accurate, it is not practical for bigger graphs since its complexity increases exponentially with the number of vertices. In the code, the function is_valid_coloring checks if a given coloring is valid by ensuring that no adjacent nodes share the same color.

The pseudocode is as follows:

```
For each color count from 1 up to n:
    For each possible coloring of the graph with
        the current color count:
        If the coloring is valid:
            Return the current color count
```

B. Greedy Heuristic

The greedy heuristic method sequentially colors vertices by assigning the lowest color that doesn't conflict with neighbors that have already been colored. Although quicker, this method offers a very effective approximation but does not ensure the minimal chromatic number. Two variants of the greedy heuristic—the top and bottom heuristics—were tested in the implementation and compared in IV-A. While the bottom heuristic arranges the vertices in ascending order, the top heuristic arranges them in decreasing order of degree. Both strategies were tested, and it was found that there was no significant difference in either approach's performance. Therefore, for the rest of the report, the top version of the heuristic was chosen.

The pseudocode is as follows:

```
For each vertex in graph, sorted by degree (
    highest to lowest):
    Get colors of neighboring vertices
    Set color to 0
    While color is used by any neighbor:
        Increment color
    Assign color to vertex
```

III. EXPERIMENTAL METHODOLOGY

In the experiments, graphs with different vertex counts (starting from 4) and edge densities (12.5%, 25%, 50%, and 75%) were used. For every instance of the graph, it was noted:

1. The chromatic number
2. Execution time in milliseconds
3. Number of basic operations
4. Number of configurations tested
5. Precision of the greedy heuristic compared to exhaustive search

Each algorithm was ran several times for every graph configuration in order to get accurate execution time measurements. The results were then averaged to take into consideration possible variations in run durations.

IV. RESULTS AND DISCUSSION

The results of the experiments demonstrate how significantly the greedy heuristic and exhaustive search algorithms perform differently. The exhaustive search's execution durations increased quickly as the number of vertices

and edge density increased, demonstrating the search's exponential complexity.

A. Execution Time Analysis

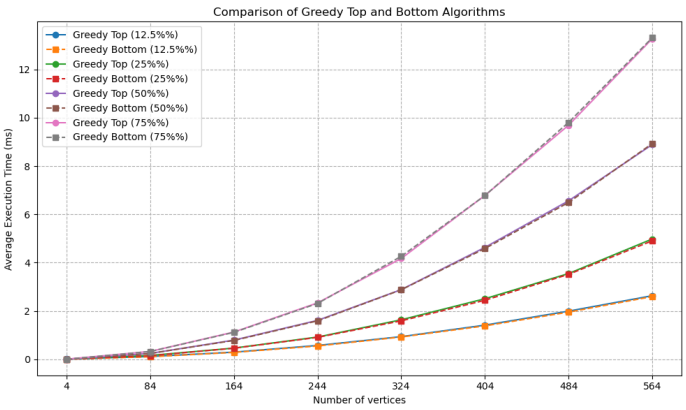

Fig. 1: Greedy algorithm variants execution times

The "Greedy Top" and "Greedy Bottom" heuristics' execution times are almost the same, indicating that there is little difference between them in the results. Both approaches are equally suited for effectively estimating the chromatic number, despite the fact that each variant utilizes a different vertex ordering. This decision has little effect on performance.

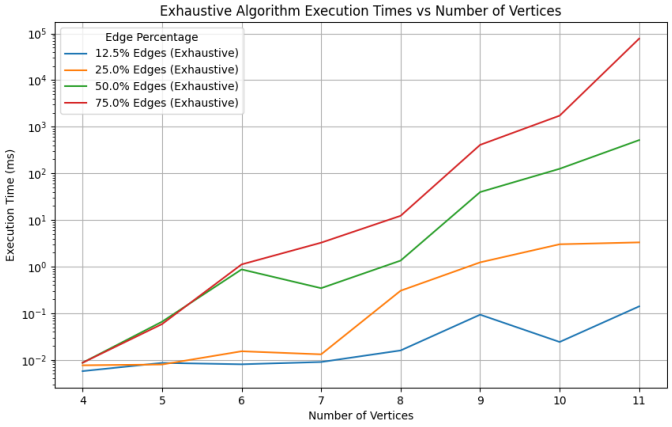

Fig. 2: Exhaustive algorithm execution time

Note: The graph presented in the figure above was analyzed and understood more easily by using logarithmic scales.

The exponential time complexity predicted for exhaustive search is confirmed by the execution times quickly reaching impractical levels as the vertex count rises for edge densities such as 50% and 75%. The exhaustive search strategy becomes impractical for larger instances when graphs with more than 11 vertices start to take an excessive amount of time to finish. Although this method is accurate, it can only be applied to small to medium-sized networks due to its processing complexity.

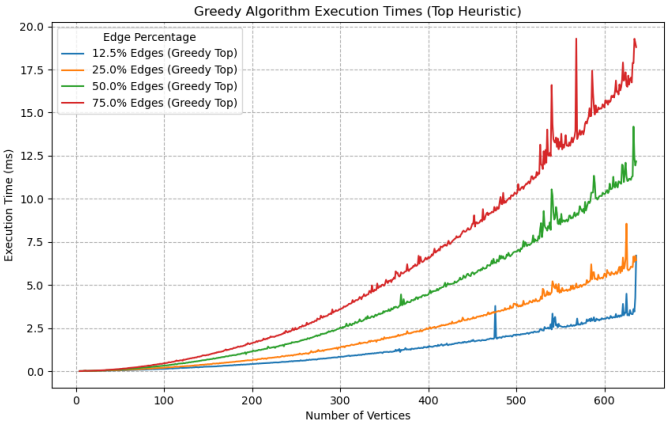

Fig. 3: Greedy algorithm execution time

The greedy heuristic scales much better, with execution time increasing almost linearly. Although it is less accurate in determining the lowest chromatic number, the approach is still effective at 600 vertices and different densities, making it appropriate for big graphs.

B. Exhaustive Search Results

TABLE I: Performance Metrics for Exhaustive Search Algorithm

n	m	χ	**Basic Ops**	**Configs Tested**
4	0 (12.5%)	1	0	1
4	1 (25%)	2	4	4
4	3 (50%)	2	9	5
4	4 (75%)	2	9	5
5	1 (12.5%)	2	4	4
5	2 (25%)	2	5	4
5	5 (50%)	3	143	72
5	7 (75%)	3	119	68
6	1 (12.5%)	2	3	3
6	3 (25%)	2	24	14
6	7 (50%)	4	2315	894
6	11 (75%)	4	3411	1218
7	2 (12.5%)	2	5	4
7	5 (25%)	2	19	9
7	10 (50%)	3	797	415
7	15 (75%)	4	9522	3521
8	3 (12.5%)	2	17	11
8	7 (25%)	3	642	308
8	14 (50%)	3	4099	1452
8	21 (75%)	4	36514	13183
9	4 (12.5%)	2	181	114
9	9 (25%)	3	3295	1338
9	18 (50%)	4	114325	44964
9	27 (75%)	5	1430798	396485
10	5 (12.5%)	2	35	21
10	11 (25%)	3	7282	3422
10	22 (50%)	4	373026	136841
10	33 (75%)	5	6332213	1673621
11	6 (12.5%)	2	238	135
11	13 (25%)	3	7072	3433
11	27 (50%)	4	1631599	544478
11	41 (75%)	6	310937886	67573977

- n: The number of vertices in a graph.
- m: The number of edges in a graph.
- χ: The chromatic number of a graph.

B.1 Basic Operations

Each edge check is considered a basic operation, and the number of basic operations increases with the number of vertices and edges, especially in denser networks. This leads to the observed exponential increase in fundamental operations with greater edge percentages and larger graph sizes.

B.2 Configurations Tested

Each configuration is a unique combination of color assignments to the vertices, and the number of configurations grows exponentially as the number of vertices increases.

C. Greedy Heuristic Results

TABLE II: Performance Metrics for Greedy Heuristic Algorithm

n	m	χ	**Basic Ops**	**Configs Tested**	**Precision**
4	0 (12.5%)	1	0	4	0
4	1 (25%)	2	1	4	0
4	3 (50%)	2	2	4	0
4	4 (75%)	2	2	4	0
5	1 (12.5%)	2	1	5	0
5	2 (25%)	2	2	5	0
5	5 (50%)	3	4	5	0
5	7 (75%)	3	4	5	0
6	1 (12.5%)	2	1	6	0
6	3 (25%)	2	2	6	0
6	7 (50%)	4	7	6	0
6	11 (75%)	4	9	6	0
7	2 (12.5%)	2	2	7	0
7	5 (25%)	2	3	7	0
7	10 (50%)	3	7	7	0
7	15 (75%)	4	13	7	0
8	3 (12.5%)	2	3	8	0
8	7 (25%)	3	6	8	0
8	14 (50%)	4	8	8	1
8	21 (75%)	4	14	8	0
9	4 (12.5%)	2	3	9	0
9	9 (25%)	3	7	9	0
9	18 (50%)	4	16	9	0
9	27 (75%)	5	16	9	0
10	5 (12.5%)	2	3	10	0
10	11 (25%)	3	10	10	0
10	22 (50%)	4	12	10	0
10	33 (75%)	5	22	10	0
11	6 (12.5%)	2	5	11	0
11	13 (25%)	3	6	11	0
11	27 (50%)	4	14	11	0
11	41 (75%)	6	29	11	0
12	8 (12.5%)	2	5	12	?
12	16 (25%)	3	10	12	?
12	33 (50%)	4	18	12	?
12	49 (75%)	7	34	12	?
13	9 (12.5%)	3	7	13	?
13	19 (25%)	3	11	13	?
13	39 (50%)	5	24	13	?
13	58 (75%)	6	34	13	?
14	11 (12.5%)	2	7	14	?
14	22 (25%)	4	11	14	?
14	45 (50%)	5	26	14	?
14	68 (75%)	7	42	14	?
15	13 (12.5%)	3	11	15	?
15	26 (25%)	3	17	15	?
15	52 (50%)	6	27	15	?
15	78 (75%)	7	48	15	?

C.1 Basic Operations

In the greedy heuristic algorithm, basic operations are the comparisons made when assigning colors to each vertex. For each vertex, the method examines the colors of its neighbors to find the smallest available color that isn't used by any adjacent vertices. To perform this check, a while loop is used, and a simple operation is counted for each loop iteration.

As the number of vertices grows, so does the number of fundamental operations, however the growth is more linear than that of the exhaustive search algorithms.

C.2 Configurations Tested

The configurations tested indicate the number of vertices to which a color is applied during execution. The number of configurations evaluated is equal to the number of vertices in the graph because each vertex's color assignment is considered as a distinct configuration.

C.3 Precision

In this analysis, precision is the difference between the chromatic number determined by the greedy heuristic algorithm and the chromatic number found by exhaustive search.

Because exhaustive search could only be performed up to graphs with 11 vertices, the greedy heuristic's precision can only be evaluated up to this limit. With these results, the greedy heuristic technique only failed to identify the precise chromatic number once:

- For a graph with 8 vertices and 50% edge density, the greedy heuristic assigned a chromatic number one unit higher than the real chromatic number, giving this graph a precision of 1.

V. COMPLEXITY ANALYSIS

The algorithms' efficiency and scalability are the main topics of this section. Despite its accuracy, the exhaustive search technique has significant computational costs that increase exponentially with the quantity of the input. While there are certain accuracy trade-offs, the greedy heuristic provides a quicker and more scalable option. The time complexity of each strategy is described in the analysis that follows.

A. Exhaustive Search Formal Complexity Analysis

The exhaustive search algorithm is highly accurate but computationally expensive because it attempts every possible color configuration for a given graph. It has mainly two parts that contribute to its complexity:

1. Generating Color Configurations: This algorithm explores all possible coloring up to n colors for a network with n vertices. Since each vertex can independently accept any of the n colors, the worst-case scenario is that there are n^n potential color configurations.
2. Validity Check for Each Configuration: The algorithm compares the colors of every neighboring vertex to determine whether a given color configuration

is valid. This step has a complexity of $O(m)$ given m edges.

The exhaustive search algorithm has an overall time complexity of:

$$O(n^n \cdot m)$$

B. Greedy Heuristic Formal Complexity Analysis

Instead of examining every potential configuration, the greedy heuristic algorithm selects colors for a network based on vertex order in an effort to color it as efficiently as feasible. The base of its complexity is:

- Sorting Vertices by Degree: Vertices are first sorted by their degree, or the number of edges that connect each vertex, in descending order. $O(n \log n)$ is the complexity of this part, where n is the number of vertices.
- Assigning Colors to Vertices: The algorithm selects the smallest color that isn't being utilized by the vertices that are adjacent to each vertex. In the worst scenario, this requires verifying each of the n vertices up to d adjacent vertices, resulting in a complexity of $O(n \cdot d)$. The worst-case scenario for a dense graph is $O(n^2)$, where d can be as high as n.

The greedy heuristic algorithm's overall complexity is:

$$O(n^2)$$

VI. Comparison of Experimental and Formal Analysis Results

This section examines how the empirical results match the theoretical complexities determined for the greedy heuristic and exhaustive search algorithms. This comparison focuses on how well each algorithm performs in practical situations in contrast to theoretical predictions, with a focus on execution time, basic operations and configurations tested.

A. Execution time

According to the formal complexity analysis, the exhaustive search algorithm's exponential time complexity is $O(n^n \cdot m)$. The experimental results confirmed this prediction, showing that execution times grew significantly with increasing vertex count and edge density. As seen in Figure 2, at higher edge densities (e.g., 50% and 75%), the time required to finish searches for graphs with more than 10 vertices grew too long. This is in line with theoretical expectations, as the exponential nature of the complexity is represented in the rapidly increasing times.

On the other hand, the greedy heuristic's theoretical complexity was $O(n^2)$, suggesting a more scalable method. According to the experimental results, the greedy heuristic scaled approximately linearly with graph size, demonstrating that its time complexity is noticeably better than the exhaustive search for bigger graphs. Figure 3 demonstrates this, showing that even for larger graphs and higher densities, the greedy heuristic's execution time grew linearly. This solid agreement between empirical and theoretical results demonstrates that the greedy algorithm operates more efficiently.

B. Basic Operations and Configurations Tested

As expected, the number of configurations tested by the exhaustive search increased exponentially with the number of vertices, roughly agreeing with the $O(n^n \cdot m)$ complexity. Similarly, the number of basic operations grew exponentially, confirming the formal analysis.

According to its $O(n^2)$ complexity, the greedy heuristic demonstrated more controlled growth in both basic operations and configurations studied. This is in line with predictions because the greedy heuristic simply assigns colors in a single pass through vertices rather than testing numerous combinations thoroughly.

VII. Estimating Execution Time for Larger Instances

Based on theoretical time complexity and analyzed experimental data, it was possible to estimate the execution times of each approach for much larger inputs in order to evaluate how well they scale with rising problem sizes.

A. Exhaustive Search

According to experimental data, the exhaustive algorithm took about 125 miliseconds to finish with an input size of $n = 10$ vertices and $m = 11$ edges (50% edge density). Applying the following formula:

$$T(n) = T(n_0) \cdot \left(\frac{n}{n_0}\right)^2$$

Assuming that $T(n_0)$ is the time for the known input n_0, we calculate that the execution time for $n = 10000$ and 50% of edge density would be around:

$$T(10000) = 125 \cdot \left(\frac{10000}{10}\right)^2 = 125 \cdot 1000^2 = 125000000 \text{ ms}$$

Converting this to hours:

$$\frac{12500000}{3600000} \approx 34.72 \text{ hours}$$

B. Greedy Heuristic

Considering that the greedy algorithm takes around 6.95 miliseconds for $n = 500$ vertices and 50% edge density, according to the formula used before, we can estimate the time for $n = 10000$ vertices:

$$T(10000) = 6.95 \cdot \left(\frac{10000}{500}\right)^2 = 6.95 \cdot 20^2 = 2780 \text{ ms}$$

Converting this to seconds:

$$\frac{2780}{1000} = 2.78 \text{ seconds}$$

VIII. Determining the Largest Graph Feasible for Efficient Processing

Due to limitations in hardware capabilities and execution time, real graph size limits are expected, given the exponential complexity of comprehensive graph search methods. To determine the greatest manageable graph size that

kept a reasonable execution time, incremental testing was done using progressively bigger graphs.

A. Exhaustive Search

Due to its exponential time complexity, this method is only practical for short graphs. The upper limit for this exhaustive search was around $n = 11$ vertices, according to tests. After this, the processing time started to take too long (more than 1 minute) making the exhaustive search unfeasible for larger graphs.

B. Greedy Heuristic

With its more efficient complexity, the greedy heuristic strategy was capable of handling increasingly huge graphs in a fair amount of time. In fact, the time needed to create each graph gradually surpassed the time needed for the heuristic method to solve it as graph sizes grew. Despite the lack of a clear maximum number of nodes, the heuristic was found to scale well, which made it a viable option for larger graphs where the main constraint was the graph generation time rather than the algorithm's processing time.

IX. CONCLUSION

This project analyzed the execution time and other metrics of different algorithms, comparing experimental results with theoretical predictions. The exhaustive approach demonstrated a noticeable performance bottleneck, enhancing the importance of high temporal complexity. In contrast, stronger algorithms performed substantially better with bigger input forms. The experimental results closely matched the formal analysis, demonstrating the predictive potential of complexity analysis. Overall, this work emphasizes the need of algorithm optimization and efficient analysis when dealing with bigger inputs.

REFERENCES

[1] "Chromatic number", 2024, Accessed: 2024-11-8.
URL: https://www.sciencedirect.com/topics/mathematics/chromatic-number

[2] "Graph coloring", 2024, Accessed: 2024-11-9.
URL: https://en.wikipedia.org/wiki/Graph_coloring

[1] [2]