Randomized Algorithms for Chromatic Number Approximation: A Comparative Study

Rafael Curado

***Abstract* –This study compares two strategies for calculating the chromatic number of graphs: a developed random greedy algorithm and the random sequential approach provided by the NetworkX package. The examination focuses on the performance of these methods using important indicators such as the number of basic operations performed, execution times, the number of unique solutions tested, and the correctness of the resulting chromatic numbers. To comprehensively examine the methods, a diverse range of graph examples with varying sizes, edge densities, and benchmark graphs from internet repositories were used. While the formal complexity of the random greedy algorithm is thoroughly examined, the NetworkX random sequential approach is only discussed in the results and discussion in order to provide a comparative viewpoint based on its experimental performance. This combination of theoretical research and computational experimentation provides a full assessment of the random greedy algorithm's scalability and efficiency.**

***Resumo* –Este estudo compara duas estratégias para calcular o número cromático de grafos: um algoritmo aleatório ganancioso desenvolvido e a abordagem sequencial aleatória fornecida pelo pacote NetworkX. O exame centra-se no desempenho destes métodos utilizando indicadores importantes como o número de operações básicas realizadas, tempos de execução, o número de soluções únicas testadas e a exatidão dos números cromáticos resultantes. Para examinar de forma abrangente os métodos, foi utilizada uma gama diversificada de exemplos de gráficos com tamanhos variados, densidades de arestas e gráficos de referência de repositórios da Internet. Embora a complexidade formal do algoritmo aleatório ganancioso seja examinada minuciosamente, a abordagem sequencial aleatória do NetworkX é apenas discutida nos resultados e na discussão, a fim de fornecer um ponto de vista comparativo baseado em seu desempenho experimental. Esta combinação de pesquisa teórica e experimentação computacional fornece uma avaliação completa da escalabilidade e eficiência do algoritmo ganancioso aleatório.**

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

***Palavras chave* –Número Cromático, Coloração de Grafos, Pesquisa Exaustiva, Heurística Gulosa, Algoritmos Aleatórios, 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 examine the possibility of randomized algorithms finding a balance between accuracy of solution and computing economy.

II. ALGORITHMS FOR CHROMATIC NUMBER CALCULATION

A. *Random Greedy*

The Random Greedy Algorithm iteratively applies a greedy coloring process on random vertex orderings in order to estimate the chromatic number. The algorithm determines the chromatic number for the present configuration, colors the graph greedily based on the unique random order of vertices it generates in each trial. To improve exploration, it avoids repeating previously tested orders and keeps note of the best chromatic number achieved across several attempts. The approach is appropriate for larger graphs because it effectively strikes a balance between computing effort and solution quality.

The pseudocode is as follows:

```
Initialize bestChromaticNumber to a very large
    value
Initialize testedOrders as an empty set

Repeat for a fixed number of trials:
    Generate a random order of vertices not seen
        before
    Add this order to testedOrders

    Initialize coloring as empty
    For each vertex in the random order:
        Get colors of neighboring vertices
        Set color to 0
        While color is used by any neighbor:
            Increment color
        Assign color to vertex

    Compute the chromatic number (maximum color
        used + 1)
    Update bestChromaticNumber if the new
        chromatic number is smaller

Return bestChromaticNumber
```

B. *NetworkX Random Sequential*

NetworkX's Random Sequential technique is a greedy coloring method in which the vertices are randomly sorted prior to applying the greedy coloring approach. Each vertex is given the smallest color that isn't already being utilized by its neighbors for a specific random order. This randomness makes it a non-deterministic but simple method of chromatic number estimation because it can produce various results across several runs.

III. EXPERIMENTAL METHODOLOGY

Using the same graph generation procedures as the first project, graphs with different vertex counts (beginning at 4) and edge densities (12.5%, 25%, 50%, and 75%) were used in this experiment. Two sets of benchmark graphs were also included: the Sedgewick & Wayne graphs, namely the SWtinyG and SWmediumG instances, and the Facebook graph from Stanford's SNAP dataset (ego-Facebook). The chromatic number, execution time (in milliseconds), number of basic operations, number of configurations evaluated, and the precision of the new algorithms in comparison to the exhaustive search algorithm were all noted for every graph configuration. However, because the exhaustive technique proved too computationally demanding for bigger graphs, the precision comparison was limited to graphs with up to 11 vertices. To account for variations in execution times, each algorithm was run several times for each graph configuration. The results were averaged to guarantee precise time measurements and minimize runtime fluctuations.

IV. RESULTS AND DISCUSSION

This section is broken into two parts: the first focuses on results from generated graph instances, which were the same used in the first project, and the second on benchmark graph instances.

A. *Generated Graphs Instances*

The generated graphs, which ranged from 4 to more vertices and had varied edge densities (12.5%, 25%, 50%, and 75%), allowed for controlled experiments to assess algorithm performance across a variety of graph sizes and densities.

A.1 Chromatic Number

Both randomized algorithms (Random Greedy and NetworkX's Random Sequential) generated chromatic numbers similar to the greedy algorithm, with just slight changes. In circumstances where edge densities were higher, like 50% and 75%, as seen in Figure 3, the random algorithms occasionally produced somewhat superior results, demonstrating their adaptability to denser edge configurations. However, for less dense graphs with edge densities of 12.5%, as seen in Figure 1 and 25%, the three techniques produced similar chromatic numbers. This shows that at lower edge densities, the problem's basic complexity limits the possibility for randomized solutions to outperform the greedy approach.

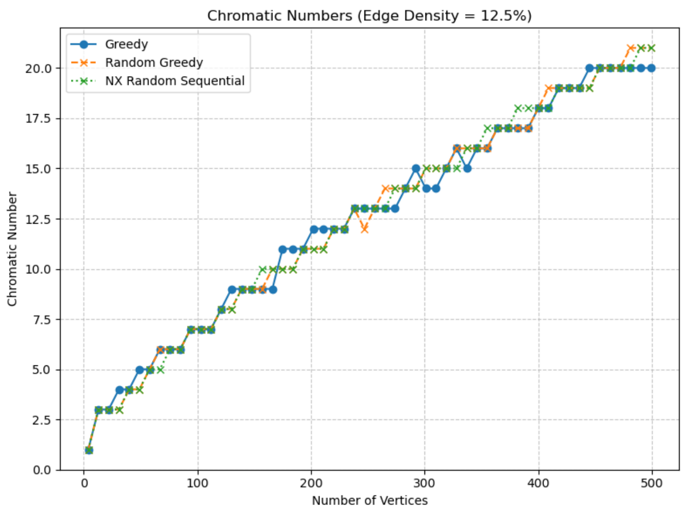

Fig. 1: Chromatic Numbers for 12.5% Edge Density

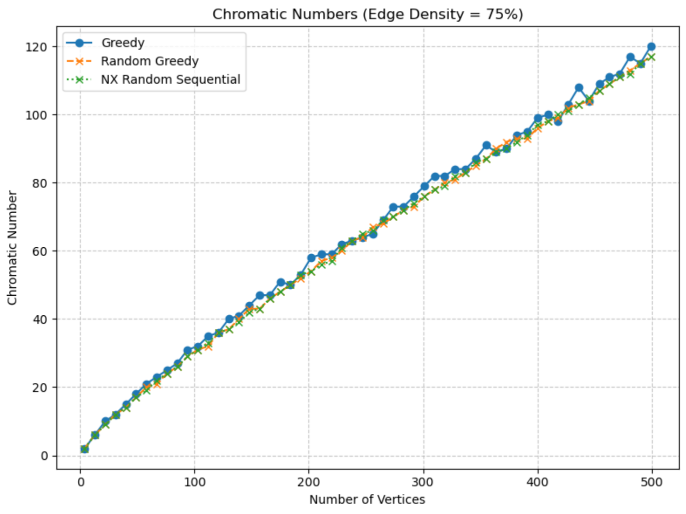

Fig. 2: Chromatic Numbers for 75% Edge Density

The comparisons of the three algorithms—Greedy, Random Greedy, and NetworkX's Random Sequential—provided demonstrating results. Table I compares each algorithm's performance based on chromatic number differences:

TABLE I: Performance Comparison of Chromatic Numbers

Comparison	Mean Difference	Won	Tied	Lost
Random Greedy vs Greedy	-0.68	1,003	770	215
NX Random Sequential vs Greedy	-0.67	1,016	759	213
Random Greedy vs NX Random Sequential	-0.01	307	1,380	301

To assess the accuracy of the Random Greedy and NetworkX Random Sequential algorithms, their chromatic numbers were compared to those acquired from the exhaustive search for graphs with up to 11 vertices. The findings are summarized in Table II.

TABLE II: Performance Metrics for Random Greedy Algorithm

n	m	χ	**Basic Ops**	**Configs Tested**	**Precision**
4	0 (12.5%)	1	0	96	0
4	1 (25%)	2	24	96	0
4	3 (50%)	2	54	96	0
4	4 (75%)	2	48	96	0
5	1 (12.5%)	2	30	150	0
5	2 (25%)	2	43	150	0
5	5 (50%)	3	106	150	0
5	7 (75%)	3	127	150	0
6	1 (12.5%)	2	36	216	0
6	3 (25%)	2	82	216	0
6	7 (50%)	4	222	216	0
6	11 (75%)	4	279	216	0
7	2 (12.5%)	2	51	294	0
7	5 (25%)	2	151	294	0
7	10 (50%)	3	248	294	0
7	15 (75%)	4	362	294	0
8	3 (12.5%)	2	108	384	0
8	7 (25%)	3	242	384	0
8	14 (50%)	3	399	384	0
8	21 (75%)	4	545	384	0
9	4 (12.5%)	2	139	486	0
9	9 (25%)	3	311	486	0
9	18 (50%)	4	605	486	0
9	27 (75%)	5	909	486	0
10	5 (12.5%)	2	216	600	0
10	11 (25%)	3	398	600	0
10	22 (50%)	4	739	600	0
10	33 (75%)	5	1149	600	0
11	6 (12.5%)	2	252	726	0
11	13 (25%)	3	468	726	0
11	27 (50%)	4	1051	726	0
11	41 (75%)	6	1721	726	0

Note: Precision is the difference between the chromatic number determined by the analyzed algorithm and the chromatic number found by exhaustive search.

Both random techniques matched the optimal chromatic numbers found by exhaustive search for tested graphs ($\leq$11 vertices).

A.2 Execution Time

The better chromatic numbers came at the expense of longer execution times for both random algorithms, with the random greedy approach being the most computationally expensive.

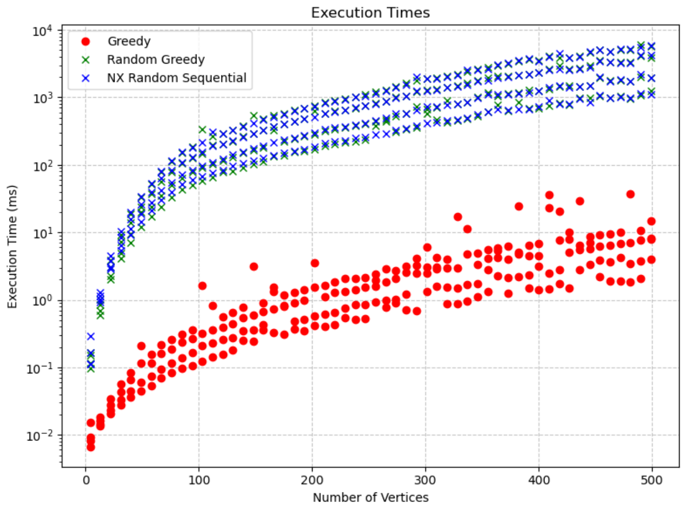

Fig. 3: Execution Times for Generated Graphs

A.3 Basic Operations and Configurations Tested

For generated graph instances, the Greedy algorithm remains the most simple and efficient, needing significantly fewer basic operations and tested setups than the other approaches. The Random Greedy algorithm has a significantly higher number of operations, indicating a more complex decision-making process. While the NetworkX Random Sequential algorithm does not provide particular counts for the operations or configurations evaluated, its execution times are comparable to those of the Random Greedy method, indicating a significant computational effort.

B. Benchmark Graph Instances

The benchmark graph instances, which included SWtinyG, SWmediumG, and the ego-Facebook graph, offered a realistic basis for testing the algorithms. These graphs differ from randomly generated instances in structure and edge distribution, providing a diverse test set.

B.1 Chromatic Number

For the Facebook graphs, the Greedy algorithm often provides lower chromatic numbers than the other two algorithms, as Fig. 4 shows. This suggests that the Greedy algorithm frequently selects a more optimal coloring. The Random Greedy algorithm performs slightly worse than Greedy, with some graphs having a minor increase in chromatic number, such as when it yields a chromatic number of 22 for the 224-vertice graph, compared to Greedy's 21. The NetworkX Random Sequential algorithm often produces the largest chromatic numbers, particularly as graph size rises. with example, with 747 vertices, it returns a chromatic number of 80, compared to Greedy's 75.

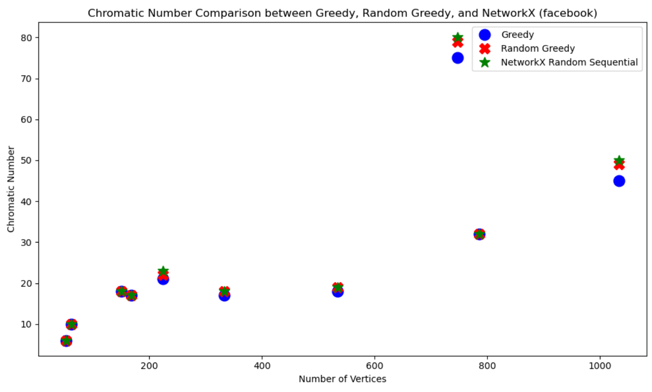

Fig. 4: Chromatic Numbers for Facebook Graphs

For the SW repository, which consists of two graphs—SWtinyG (a little graph) and SWmediumG (a larger graph), the performance of the three algorithms reveals varied levels of computing efficiency and effectiveness in determining chromatic numbers.

For SWtinyG, the three algorithms—Greedy Heuristic, Random Greedy, and NetworkX Random Sequential—all reach the same chromatic number. This demonstrates that the graph's simplicity enables all algorithms to discover equally effective solutions.

For SWmediumG, the Greedy Heuristic provides a slightly greater chromatic number than Random Greedy and NetworkX Random Sequential. This shows that, while the Greedy Heuristic focuses on computing efficiency, its solutions may not necessarily be ideal for increasingly complicated graph structures. Random Greedy and NetworkX Random Sequential, with their more extensive techniques, produce marginally better outcomes.

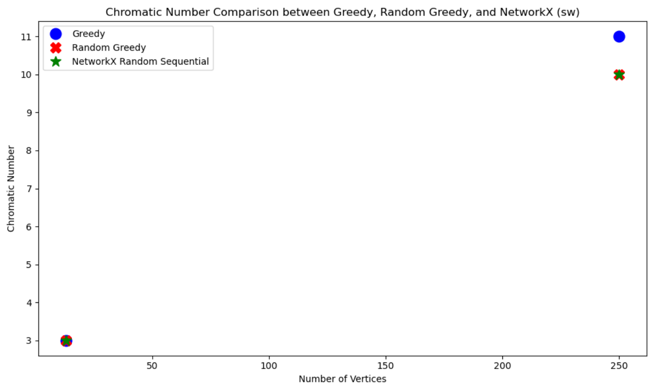

Fig. 5: Chromatic Numbers for SW Graphs

B.2 *Execution Time*

Looking at execution time, the Greedy algorithm is the fastest of the three. Greedy takes about 9.1 milliseconds for a graph with 1034 vertices, whereas the Random Greedy algorithm takes 2408 milliseconds and the NetworkX technique takes 3430. For smaller graphs, the Greedy algorithm is likewise much faster—it takes only 0.0579ms for the 52-vertex graph, compared to 14.5ms for Random Greedy and 18.7ms for NetworkX. The Random Greedy algorithm's execution time increases significantly as the graph size expands. For example, with 1034 vertices, it takes more than 2400ms, indicating its low scalability. Similarly, the NetworkX Random Sequential algorithm exhibits long execution times that increase significantly with graph size, but it is usually quicker than Random Greedy.

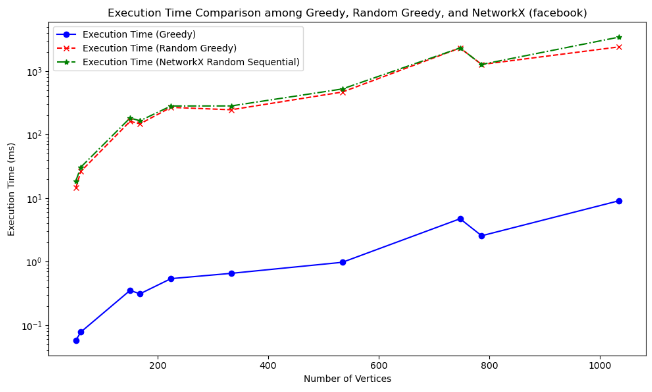

Fig. 6: Execution Times for Facebook Graphs

The execution times of these algorithms vary greatly. The Greedy Heuristic is by far the fastest approach for both graphs, requiring least effort and processing resources. Even with the larger SWmediumG, it remains extremely efficient.

In contrast, Random Greedy and NetworkX Random Sequential are more slower, especially with SWmediumG. NetworkX Random Sequential, in particular, has the longest execution time, owing to its computationally expensive nature. While these algorithms may produce better chromatic numbers, their applicability for bigger graphs is restricted due to their high processing needs.

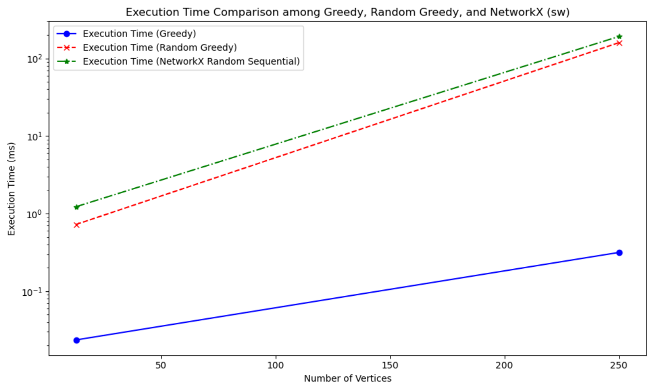

Fig. 7: Execution Times for SW Graphs

B.3 *Basic Operations and Configurations Tested*

In terms of basic operations and tested configurations, the Greedy algorithm is the simplest and most efficient. The Random Greedy method performs far more operations, as evidenced by the 4.4 million operations for the 1034-vertex facebook graph versus Greedy's 9483 operations. The NetworkX Random Sequential algorithm does not disclose particular operation counts, but its execution speeds indicate that it executes a huge number of operations and tested configurations, similar to Random Greedy.

V. Complexity Analysis

The Random Greedy Algorithm is intended to iteratively try vertex coloring with randomized vertex ordering, picking the best chromatic number after a set number of trials. The technique does not examine every possible configuration, instead relying on randomness and greediness to approximate the chromatic number. The difficulty of this method can be separated into key steps:

- Random Vertex Order Generation: In each trial, the algorithm generates a unique random order of the vertices. Generating this random order involves shuffling the n vertices, which takes $O(n)$, and ensuring uniqueness using a hash set, which takes $O(1)$ on average. Across $T = \min(500, 6n)$ trials, the complexity of this step is:
$$O(T \cdot n)$$
- Greedy Coloring: The algorithm uses the greedy heuristic to assign colors to vertices in each random sequence. This involves iterating over all n vertices in the specified order. Check the colors of each vertex's neighbors and assign the smallest available color. In a dense graph, each vertex can have up to $n-1$ neighbors, making the worst-case complexity for this step $O(n^2)$.

Since the greedy coloring is repeated T times, the total complexity for this step is:

$$O(T \cdot n^2)$$

The number of trials, T, is determined by the formula $T = \min(500, 6n)$:

- For small graphs ($n \leq 83$, where $6n \leq 500$), $T = 6n$, and the complexity becomes:
$$O((6n) \cdot n^2) = O(n^3).$$
- For large graphs ($n > 83$, where $6n > 500$), $T = 500$ (constant), and the complexity simplifies to:
$$O(500 \cdot n^2) = O(n^2).$$

VI. Comparison of Experimental and Formal Analysis Results

In terms of temporal complexity, the experimental results show that execution times scale predictably with the number of vertices. For tiny graphs ($n \leq 83$), the observed execution durations reflect the $O(n^3)$ complexity inferred from the formal analysis. For larger graphs ($n \geq 83$), the execution durations settle due to the limited number of trials ($T = 500$) , which is consistent with the $O(n^2)$ theoretical prediction. This correlation demonstrates the theoretical bounds' resilience in describing practical performance.

The experimental results support the formal analysis of the Random Greedy algorithm. While practical results may differ due to randomization and graph-specific features, the theoretical bounds give a dependable framework for comprehending the algorithm's behavior and scaleability.

VII. Estimating Execution Time for Larger Instances

According to experimental data, the random greedy algorithm took about 26 miliseconds to finish with an input size of $n = 50$ vertices and $m = 612$ 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:

$$T(10000) = 26 \cdot \left(\frac{10000}{50}\right)^2 = 26 \cdot 200^2 = 1040000 \text{ ms}$$

Converting this to minutes:

$$\frac{1040000}{60000} \approx 17.3 \text{ minutes}$$

VIII. Determining the Largest Graph Feasible for Efficient Processing

The largest network successfully processed throughout the testing contained 500 vertices and took around 5 seconds to finish. This is the upper limit of computational feasibility for the given configuration, taking into account both execution time and memory use limits. Beyond this size, the resources required are likely to surpass realistic bounds, hence 500 vertices is the effective cutoff for graph processing in this context.

IX. Conclusion

This study examined two randomized algorithms for estimating the chromatic number: a Random Greedy algorithm and NetworkX's Random Sequential approach. When tested across different graph sizes, edge densities, and benchmark datasets, the Random Greedy technique performed competitively, typically providing somewhat higher chromatic values than the Greedy heuristic, especially for denser graphs. However, this enhancement resulted in higher computing expenses.

Both algorithms performed well on small to medium-sized graphs, but their scalability was limited by execution time. Formal examination of the Random Greedy algorithm was consistent with experimental results, confirming that its complexity transitions from $O(n^3)$ for smaller graphs to $O(n^2)$ for bigger ones with restricted trials.

These findings emphasize the potential of randomized methods to improve chromatic number approximation while balancing their computing requirements. Future research could look into hybrid approaches or optimizations to improve scalability and efficiency with larger graphs.

References

[1] "Networkx greedy coloring", 2024, Accessed: 2024-12-1. **URL:** https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.coloring.greedy_color.html#r3570b859f9f4-1

[2] "Networkx graph coloring", 2024, Accessed: 2024-11-29. **URL:** `https://networkx.org/documentation/stable/reference/algorithms/coloring.html`

[3] "Wikipedia graph coloring", 2024, Accessed: 2024-11-19. **URL:** `https://en.wikipedia.org/wiki/Graph_coloring`

[4] "Stanford facebook graphs data", 2024, Accessed: 2024-12-1. **URL:** `https://snap.stanford.edu/data/ego-Facebook.html`

[1] [2] [3] [4]