Comparative Analysis of Algorithms for the Graph Coloring Problem: Minimizing Chromatic Number

AUTHORS
Nabila Zahra, Simal Anjum, Muhammad Youshay

AFFILIATIONS
Algorithms Design and Analysis, Habib University

INTRODUCTION

Graph coloring, a key challenge in combinatorial optimization, requires colors to be assigned to graph vertices so no adjacent vertices share the same color. This NP-complete problem is crucial for applications like software optimization and network management.

We have examined three algorithms, Brute Force, Greedy Algorithm, and Ant Colony Optimization (ACO) to address this problem. Through computational tests and algorithmic analysis, we examine the theoretical complexity, efficacy, and practical utility of each algorithm across different graph sizes This research highlights practical insights and theoretical contributions to solving the graph coloring problem.

METHODOLOGY

To compare the performance of the three algorithms (Brute Force, Greedy Algorithm and Ant Colony Optimization) we implemented these algorithms using Python on a machine with an Intel Core i5-63000U CPU @ 2.4GHz, 8 GB RAM, running Windows 11. The datasets used are:

- *smallest_data.col: 10 nodes, 15 edges (self-generated)*
- *smaller_data.col: 20 nodes, 30 edges (self-generated)*
- *queen11_11.col: 121 nodes, 3,960 edges (from Stanford GraphBase)*
- *le450_15b.col: 450 nodes, 8,169 edges (Leighton graph)*

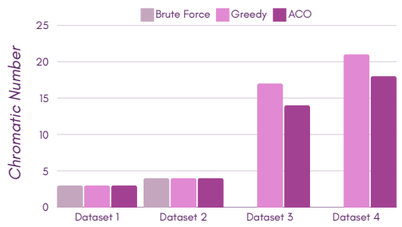
Due to computational demands, each algorithm was run three times per dataset, and the average performance metrics were calculated.

THEORETICAL COMPLEXITIES

Brute Force	$O(n^n \times n^2)$ **Exponential Time Complexity**	• Optimal Solution guaranteed • Limitations on large dataset
Greedy Algorithm	$O(n\Delta)$ $\Delta = Maximum\ degree\ of\ vertices$ **Polynomial Time Complexity**	• Not always optimal solution on complex graphs • Computationally efficient
Ant Colony Optimization	$O(t \cdot m \cdot n^2)$ $t = No.\ of\ iterations \quad m = No.\ of\ ants$ $n = No.\ of\ vertices$ **Polynomial Time Complexity**	• Balance between solution quality and computational effort. • Finds near optimal solutions

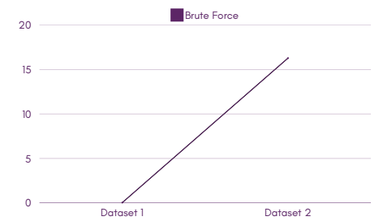
RESULTS

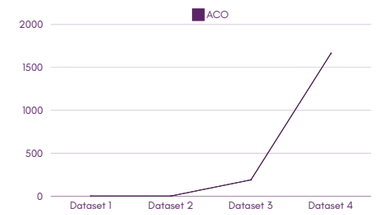
Chromatic Number Comparision



The bar chart shows the difference in chromatic numbers in the datasets between algorithms. The optimized result was given by ACO for large datasets but for smaller datasets, each algorithm gave the same result in terms of chromatic number.

Empirical Complexities





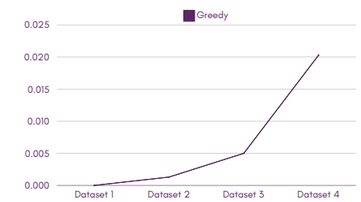


Table 1: Computational Complexity Comparison (Seconds)

Algorithm	D-1	D-2	D-3	D-4
BF	0.0208	16.2699	-	-
GA	0.0	0.0013	0.0050	0.0203
ACO	1.4223	1.4700	189.5601	1663.47

Brute force was found to be not be able to perform on larger datasets. Greedy approach was more computationally efficient that ACO for large datasets.

Dataset 1 : Chromatic Number 3 for each

Brute Force

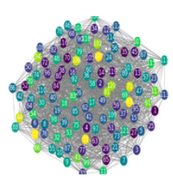
Greedy Algorithms

Ant Colony Optimisation

Dataset 3:

No graph generated for brute force due to limited computational resources

Brute Force



Greedy Algorithms
Chromatic Number: 17

Ant Colony Optimisation
Chromatic Number: 14

ANALYSIS

Brute Force Approach

Computational Complexity: Exhibits exponential growth in computational time with increasing graph size, rendering it inefficient for large datasets.

Performance Analysis: Effective on Small Datasets as it achieves optimal solutions quickly on smaller graphs but struggles with larger graphs such as queen11_11.col and le450_15b.col, often failing to complete in a reasonable timeframe.

Greedy Algorithm

Computational Complexity: Operates in polynomial time, typically $O(n\Delta)$, where n is the number of vertices and Δ is their maximum degree.

Performance Analysis: Efficiently processes graphs of all sizes, handling large datasets effectively. Often does not yield the minimum number of colors, especially in denser graphs, indicating a trade-off between speed and solution quality.

Ant Colony Optimization (ACO)

Computational Complexity: Moderately high at $O(t \cdot m \cdot n^2)$, where t is the number of iterations, m the number of ants, and n the number of vertices, due to overheads, particularly under complex problem sets.

Performance Analysis: Slower than the Greedy Algorithm but faster than Brute Force, ACO finds more optimized solutions suitable for large datasets. It's additional computational time is justified with higher solution quality.

CONCLUSION

The choice of algorithm for graph coloring depends on the graph's size, complexity, and specific needs. The Brute Force approach guarantees optimal solutions but it is unsuitable for large graphs due to its high computational demand. The Greedy Algorithm is computationally efficient but may not always provide the best solution for complex graphs. Ant Colony Optimization (ACO) offers a balance, performing well on larger, more intricate graphs.

RELATED LITERATURE

[1] Velin Kralev and Radoslava Kraleva. 2023. An analysis between different algo- rithms for the graph vertex coloring problem. Int. J. Electr. Comput. Eng. 13, 3 (June 2023), 2972-2980. DOI:https://doi.org/10.11591/ijece.v13i3.pp2972-2980
[2] M. Bessedik, R. Laib, A. Boulmerka, and H. Drias, "Ant Colony System for Graph Coloring Problem," in Proc. International Conference on Computational Intelli- gence for Modelling, Control and Automation and International Conference on In- telligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06), (2005), pp. 786-791, DOI: 10.1109/CIMCA.2005.1631360
[3] Knuth, D. E. (n.d.). The Stanford graphbase: A platform for combi- natorial computing. Knuth: The Stanford GraphBase. https://www-cs- faculty.stanford.edu/ knuth/sgb.html
[4] Leighton, F. T. (1979). A graph coloring algorithm for large scheduling prob- lems. Journal of Research of the National Bureau of Standards, 84(6), 489. https://doi.org/10.6028/jres.084.024
[5] Costa, D. and Hertz, A., 1997. Ants can colour graphs. Journal of the Operational Research Society, 48(3), (March 1997),pp.295-305. DOI:10.1057/palgrave.jors.2600357