**Graph Coloring Analysis Project Report**

**Title Page**

* Project Title
* Course Title and Number
* Name and Student ID
* Submission Date

**Abstract**

This project explores the application of graph coloring to solve scheduling problems, focusing on the efficiency and effectiveness of six different vertex ordering methods. Through the creation of conflict graphs that mimic real-world scheduling scenarios, we analyzed the performance of these methods in terms of running time and coloring efficiency. Key findings reveal significant variations in the total number of colors needed and other metrics, providing insights into the practical applications of these orderings in optimizing scheduling tasks.

**Table of Contents**

Introduction...1

Computing Environment...2

Part 1: Conflict Graph Generation...3

Algorithm Description and Implementation...3

Asymptotic Running Time Analysis...4

Distribution Analysis...5

Part 2: Vertex Ordering and Coloring...6

Vertex Ordering Implementations...6

Running Time Analysis...7

Coloring Algorithm...8

Comparative Analysis of Vertex Ordering...9

Results and Discussion...10

Limitations and Future Work...11

Appendices...12

References...13

**Introduction**

* Description of the graph coloring problem.
* Importance and applications of graph coloring in scheduling and other areas.

Objectives of the project.  
  
Graph coloring is a method used to assign "colors" to elements of a graph subject to certain constraints. In scheduling, vertices can represent tasks, and edges represent conflicts between tasks. This project aims to identify optimal scheduling by minimizing the number of colors (timeslots) required, leveraging different graph coloring strategies.

**Computing Environment**

* Detailed description of the computing environment used for the project.
  + Hardware specifications.
  + Software and development tools (Programming language, IDE, libraries).

**Part 1: Conflict Graph Generation**

* Overview of conflict graph representation and its significance.

#### Algorithm Description and Implementation

We implemented graph generators for COMPLETE, CYCLE, and RANDOM graphs using Python's random module for the UNIFORM distribution and custom functions for SKEWED and a custom distribution focusing on edge density variations.

#### Asymptotic Running Time Analysis

Theoretical running times were O(V^2) for COMPLETE graphs, O(V) for CYCLE graphs, and O(E) for RANDOM graphs, with empirical data collected through timed trials supporting these analyses.

#### Distribution Analysis

Histograms showed a uniform conflict distribution for the UNIFORM method, a higher concentration of conflicts among lower-numbered vertices for SKEWED, and a custom distribution highlighting edge density impacts on vertex conflicts.

**Part 2: Vertex Ordering and Coloring**

#### Vertex Ordering Implementations

We detailed implementations for Smallest Last, Smallest Original Degree Last, Uniform Random, Betweenness Centrality Order (BCO), Largest Degree First (LDF), and Connected Components Count (CCC), focusing on their applicability in graph coloring.

#### Running Time Analysis

Running time analyses included O(V+E) for Smallest Last and Smallest Original Degree Last, with Betweenness Centrality and Connected Components requiring more complex calculations. We used time.perf\_counter() to measure actual execution times.

#### Coloring Algorithm

Our coloring algorithm assigned the minimum feasible color to each vertex based on the ordering provided, ensuring no two adjacent vertices shared the same color. This process was iteratively applied across all orderings.

#### Comparative Analysis of Vertex Ordering

Comparative analyses revealed DSATUR and LDF generally required fewer colors than other methods, with BCO showing potential in specific graph types. The efficiency varied with graph structure and vertex degree distribution.

**Results and Discussion**

We found DSATUR and LDF to be most efficient in minimizing colors, highlighting the importance of saturation and degree in coloring efficiency. Smallest Last offered a balanced performance across different graphs.

**Limitations and Future Work**

Limitations include the simplified handling of complex distributions and graph types. Future work could explore hybrid ordering methods and apply machine learning to predict optimal orderings based on graph characteristics.

**Appendices**

* Source Code Listings
* Additional Graphs and Tables

**References**

* Citations of any external resources, academic papers, or other materials referenced in the report.