

**try reducing to minimum or removing the literary survey and add more importance to there topics in the proposal**

Here’s the revised proposal focusing on implementation and empirical analysis, with reduced literature survey and increased emphasis on methodological rigor:

**Implementation and Empirical Analysis of the Color-Coding Algorithm for Subgraph Detection**

**Abstract**  
This project implements the color-coding technique introduced by Alon, Yuster, and Zwick[[1]](#fn1) for detecting small subgraphs in large graphs. We focus on practical implementation of both randomized and derandomized variants, benchmarking their performance against theoretical bounds. Experimental evaluation will analyze runtime scaling on worst-case inputs (e.g., sparse graphs with long paths) and naturally distributed graphs (e.g., Erdős-Rényi, planar). The project aims to identify practical optimizations and validate theoretical complexity claims, bridging the gap between algorithmic theory and real-world performance.

**1. Technical Background**

The color-coding method[[1]](#fn1) uses randomized vertex coloring to simplify subgraph detection:

* **Core Mechanism**: Assign colors from to vertices. A subgraph is "colorful" if all vertices have distinct colors, enabling efficient detection via dynamic programming ( time for directed graphs) [1, §3].
* **Derandomization**: Replace probabilistic coloring with -perfect hash families, incurring logarithmic overhead ( time for cycles) [1, §4].
* **Theoretical Guarantees**: For planar/minor-closed graphs, cycles of length can be detected in expected time [1, §5].

This project will test these theoretical claims through systematic implementation and benchmarking.

**2. Implementation Objectives**

**2.1 Algorithm Variants**

1. **Randomized Path/Cycle Detection**
   * Implement dynamic programming for colorful path detection [1, Lemma 3.1].
   * Optimize memory usage for storing color sets ( space).
2. **Derandomized Version**
   * Integrate explicit -perfect hash families using Schmidt-Siegel constructions [1, §4].
   * Precompute hash functions to replace random color sampling.
3. **Generalization to Bounded Tree-Width Subgraphs**
   * Extend implementation to handle subgraphs with tree-width ( time) [1, §6].

**2.2 Performance Targets**

* Achieve deviation from theoretical scaling for .
* Reduce hidden constants in through cache-aware DP table traversal.

**3. Experimental Methodology**

**3.1 Graph Generation**

* **Worst-Case Inputs**: Sparse graphs with maximum path lengths (using recursive backedge-limited DFS).
* **Natural Distributions**:
  + Erdős-Rényi ( with ).
  + Planar graphs via Delaunay triangulation.
  + Power-law networks (Barabási-Albert model).

**3.2 Benchmarking Framework**

1. **Runtime Metrics**
   * Measure for , .
   * Profile memory usage vs. theoretical bounds ().
2. **Statistical Analysis**
   * Fit empirical runtimes to using nonlinear regression.
   * Compare derived constants against theoretical predictions [1, §3-4].
3. **Heuristic Validation**
   * Test early termination for dense subgraphs (color saturation thresholding).
   * Evaluate adaptive color sampling to reduce iterations.

**4. Expected Contributions**

1. **Implementation Artifacts**
   * Open-source C++/Python codebase with modular DP components.
   * Precomputed hash families for .
2. **Empirical Insights**
   * Quantify gap between theory and practice (e.g., observed vs. predicted).
   * Characterize input classes where derandomization overhead exceeds probabilistic gains.
3. **Practical Guidelines**
   * Thresholds for preferring randomized vs. derandomized variants based on .
   * Cache optimization strategies for DP table traversal.

**5. Project Timeline**

|  |  |  |
| --- | --- | --- |
| Phase | Deliverables | Duration |
| Implementation | Core DP engine, hash family integration | 4 weeks |
| Validation | Unit tests on synthetic worst-case graphs | 2 weeks |
| Benchmarking | Runtime profiles across space | 3 weeks |
| Analysis | Statistical modeling of complexity gaps | 3 weeks |

**References**[**[1]**](#fn1) **Alon, N., Yuster, R., & Zwick, U. (1995). Color-Coding. *Journal of the ACM*, 42(4), 844–856. Schmidt, J.P., & Siegel, A. (1990). The spatial complexity of oblivious -probe hash functions. *SIAM Journal on Computing*, 19(5), 775–786.**

This revised proposal minimizes theoretical survey in favor of implementation specifics, with all claims grounded in the core Alon-Yuster-Zwick paper[[1]](#fn1). The experimental design directly tests scalability claims from §3-4 of the original work.

⁂

1. <https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/40595747/6e2491ca-dee8-45a9-a37b-999de9419eab/210332.210337.pdf>