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] for detecting small subgraphs in large graphs. We focus on practical implementations 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, thereby bridging the gap between algorithmic theory and real-world performance.

1 Technical Background

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

- **Core Mechanism:** Vertices of the graph are randomly assigned colors from the set $\{1, \ldots, k\}$. A subgraph is termed *colorful* if all its vertices receive distinct colors. This property allows efficient detection via dynamic programming. In particular, as shown in Theorem 3.3 of [1], a colorful path of length $k-1$ can be found in $2^{O(k)} \cdot V$ expected time in undirected graphs and $2^{O(k)} \cdot E$ expected time in directed graphs
- **Cycle Detection:** Similarly, as per Theorem 3.4 in [1], a colorful cycle of size k can be detected in $2^{O(k)} \cdot V$ (or $2^{O(k)} \cdot E$) expected time.
- **Derandomization:** The randomized algorithms can be derandomized using a k-perfect family of hash functions. Constructions based on Schmidt-Siegel [2] yield such families with only an extra $O(\log V)$ factor in the worst-case running time.
- **Generalization to Bounded Tree-Width:** The method also generalizes to detecting subgraphs with bounded tree-width. For instance, Theorem 6.3 in [1] shows that for a subgraph with tree-width t, a detection algorithm runs in $O(2^{O(k)} \cdot V^{t+1})$ time.

2 Implementation Objectives

2.1 Algorithm Variants

- **Randomized Path/Cycle Detection:**
  - Implement the dynamic programming routine for finding colorful paths as described in Lemma 3.1 of [1].
  - Clearly distinguish the bounds: $2^{O(k)} \cdot V$ for undirected and $2^{O(k)} \cdot E$ for directed graphs.
- **Derandomized Version:**
  - Incorporate explicit k-perfect hash families (e.g., via Schmidt-Siegel constructions [2]) to replace random color assignments.
  - Note that this derandomization adds an extra $O(\log V)$ factor in worst-case time.
- **Generalization to Bounded Tree-Width Subgraphs:**
  - Extend the implementation to handle subgraphs with bounded tree-width (with runtime $O(2^{O(k)} \cdot V^{t+1})$, where t is the tree-width).

- **Alternative Approach – Random Orientations:**
  - Briefly explore the random orientations method (Section 2 in [1]) as an alternative technique, which orients edges according to a random permutation and then finds simple paths with competitive runtime.

2.2 Performance Targets

- Achieve empirical runtimes that deviate by less than 10% from the theoretical $2^{O(k)}$ scaling for moderate values of k (e.g., $k \leq 15$).
- Optimize memory usage in the dynamic programming tables (theoretical bound $O(k \cdot 2^k \cdot V)$) via cache-aware programming.

3 Experimental Methodology

3.1 Graph Generation

- **Worst-Case Inputs:** Generate sparse graphs designed to have long paths (e.g., via recursive backedge-limited DFS).
- **Natural Distributions:**
  - Erdős-Rényi graphs $G(n, p)$ with $p = \Theta(1/n)$.
  - Planar graphs generated via Delaunay triangulation.
  - Power-law networks (e.g., using the Barabási-Albert model).

3.2 Benchmarking Framework

- **Runtime Metrics:**
  - Measure the running time $T(k, V)$ for various combinations of k (e.g., 5, 10, 15) and graph sizes V (e.g., 10^3, 10^4, 10^5).
  - Profile memory consumption against theoretical expectations.
- **Statistical Analysis:**
  - Fit empirical runtimes to the model $c \cdot 2^{ak} \cdot V^b$ using nonlinear regression.
  - Compare the derived constants a, b, and c with theoretical predictions from [1].
- **Heuristic Validation:**
  - Evaluate early termination strategies (e.g., color saturation thresholds) and adaptive color sampling to reduce iterations.

4 Conclusion

This project aims to bridge the gap between theory and practice for the color-coding method in subgraph isomorphism problems. By implementing both the randomized and derandomized algorithms—and by exploring alternative methods like random orientations—we intend to validate theoretical bounds while gaining practical insights into performance optimization. The comprehensive empirical analysis is expected to not only support existing theoretical results (e.g., those in [1]) but also to suggest directions for further research and optimization in real-world graph applications.

References

[1] N. Alon, R. Yuster, and U. Zwick, "Color-Coding," *Journal of the ACM*, vol. 42, no. 4, pp. 844–856, 1995.

[2] J. P. Schmidt and A. Siegel, "The spatial complexity of oblivious k-probe hash functions," *SIAM Journal on Computing*, vol. 19, no. 5, pp. 775–786, 1990.