Derandomized Color Coding via Two-Stage K-Perfect Hashing

Your Name

May 3, 2025

Abstract

This paper presents a derandomized color coding algorithm using two-stage k-perfect hash families for finding colorful paths in graphs. We describe the construction of polynomial hash functions, the composition of two-stage hash families, and the dynamic programming approach for detecting colorful paths.

1 Introduction

Color coding is a powerful technique for detecting small subgraphs, such as paths or cycles, in large graphs. The derandomized version uses k-perfect hash families to systematically assign colors, ensuring coverage of all possible colorings.

2 Single Stage Hashing using GF(p)

Algorithm 1 Derandomized Color Coding to Detect Colorful Paths of Length k

```
Input: Graph G = (V, E), target path length k
Output: True if a colorful path of length k exists, else False
function DerandomizedColorCoding(G, k)
    n ← |V|
    Construct k-perfect hash family ℋ using:
    for a = 1 to k do
        for b = 0 to k − 1 do
            Add h_{a,b}(v) = ((a · v + b) mod p) mod k + 1 to ℋ      ▷ p > n, prime
        end for
    end for
    for each hash function h ∈ ℋ do
        Assign colors c[v] ← h(v) for all v ∈ V
        Initialize DP table dp[v][S] ← false for v ∈ V, S ⊆ {1, . . . , k}
        for v ∈ V do
            dp[v][{c[v]}] ← true
        end for
        for ℓ = 1 to k − 1 do
            for v ∈ V do
                for each subset S ⊆ {1, . . . , k} of size ℓ do
                    if dp[v][S] = true then
                        for each u ∈ neighbors(v) do
                            if c[u] ∉ S then
                                dp[u][S ∪ {c[u]}] ← true
                            end if
                        end for
                    end if
                end for
            end for
        end for
        for v ∈ V do
            if dp[v][{1, . . . , k}] = true then
                return true
            end if
        end for
    end for
    return false
end function
```

3 Polynomial Hash Function Construction

Algorithm 2 Polynomial Hash Function Construction

Require: Degree $k-1$, prime p
Ensure: A hash function $h(x)$ over GF(p)

Generate k random coefficients $a_0, a_1, \ldots, a_{k-1} \in \{0, \ldots, p-1\}$
while $k > 1$ and $a_{k-1} = 0$ **do**
  Re-sample a_{k-1} from $\{0, \ldots, p-1\}$
end while
Define $h(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{k-1} x^{k-1} \textbf{ mod } p$
Use Horner's Rule to evaluate $h(x)$ efficiently

4 K-Perfect Hash Family Construction

Algorithm 3 K-Perfect Hash Family Construction

Require: k: range size, n: universe size, m: number of hash functions
Ensure: Family of m polynomial hash functions

Find smallest prime $p > \max(n, k)$
for $i = 1$ to m **do**
  Generate a polynomial hash of degree $k-1$ over GF(p)
end for
Store all m functions as the hash family

5 Two-Stage K-Perfect Hash Composition

Algorithm 4 Two-Stage K-Perfect Hash Composition

Require: Stage-1: maps $[1, n] \to [0, k^2-1]$, Stage-2: maps $[0, k^2-1] \to [0, k-1]$
Ensure: Composed hash maps $v \in [1, n]$ to color $\in [0, k-1]$

Construct stage-1 hash family with $2k\lceil \log_2 n \rceil$ functions
Construct stage-2 hash family with k^2 functions
function TwoStageHash(v, i, j)
  intermediate $\leftarrow$ Stage1Hash[i](v) mod k^2
  return Stage2Hash[j](*intermediate*) mod k
end function

Algorithm 5 Two-Stage Polynomial Hash Function for Coloring

Require: Vertex identifier $v \in \{1, \ldots, n\}$, indices i_1, i_2, integer k
Ensure: Composed hash maps $v \in [1, n]$ to color $\in [0, k-1]$

```
// Stage 1: Construct family of m_1 = 2k⌈log_2 n⌉ polynomial hash functions f_{i_1}^{(1)}
Find smallest prime p_1 > max(n, k)
for each f_{i_1}^{(1)} do
    Generate k random coefficients a_0, a_1, ..., a_{k-1} ∈ {0, ..., p_1 - 1}
    while k > 1 and a_{k-1} = 0 do
        Re-sample a_{k-1} from {0, ..., p_1 - 1}
    end while
    Define f_{i_1}^{(1)}(x) = a_0 + a_1 x + ... + a_{k-1} x^{k-1} mod p_1
end for

// Stage 2: Construct family of m_2 = k^2 polynomial hash functions f_{i_2}^{(2)}
Find smallest prime p_2 > k^2
for each f_{i_2}^{(2)} do
    Generate k random coefficients b_0, b_1, ..., b_{k-1} ∈ {0, ..., p_2 - 1}
    while k > 1 and b_{k-1} = 0 do
        Re-sample b_{k-1} from {0, ..., p_2 - 1}
    end while
    Define f_{i_2}^{(2)}(y) = b_0 + b_1 y + ... + b_{k-1} y^{k-1} mod p_2
end for

function COLOR(v, i_1, i_2)
    t ← f_{i_1}^{(1)}(v) mod k^2
    return f_{i_2}^{(2)}(t) mod k
end function
```

Algorithm 6 Two-Stage LFSR-Based Color Hashing for k-Perfect Hashing

Require: Vertex identifier $v \in \{1, \dots, n\}$, indices i_1, i_2
Ensure: Color $c(v) \in \{1, \dots, k\}$

Define tap mask $T =$ `0x80200003` for primitive polynomial
Define bit widths $m = 32$, $l_1 = \lceil \log_2(k^2) \rceil$, $l_2 = \lceil \log_2 k \rceil$
Initialize two LFSR hash families:
  $H_1 \leftarrow \text{LFSRHashFamily}(T, m, 2k \cdot \lceil \log_2 n \rceil)$
  $H_2 \leftarrow \text{LFSRHashFamily}(T, m, k^2)$
function $\textsc{Color}(v, i_1, i_2)$
  $h_1 \leftarrow \text{Hash}(H_1, i_1, v, l_1) \bmod k^2$
  $h_2 \leftarrow \text{Hash}(H_2, i_2, h_1, l_2) \bmod k$
  return $h_2 + 1$
end function
function $\textsc{Hash}$(family, func_idx, steps, ℓ)
  $s \leftarrow \text{Seed}[func\_idx \bmod \text{family.size}]$
  for $j = 1$ to `steps` **do**
    $s \leftarrow \text{LFSR}\textsc{Next}(s, T)$
  end for
  $h \leftarrow 0$
  for $i = 0$ to $\ell - 1$ **do**
    $h \leftarrow (h \ll 1) \vee (s \& 1)$
    $s \leftarrow \text{LFSR}\textsc{Next}(s, T)$
  end for
  return h
end function
function LFSR$\textsc{Next}$(state, taps)
  `lsb` $\leftarrow$ `state`$\&1$
  `state` $\leftarrow$ `state` $\gg 1$
  if `lsb` $= 1$ **then**
    `state` $\leftarrow$ `state` $\oplus$ `taps`
  end if
  return `state`
end function

6 Derandomized Color Coding Algorithm

Algorithm 7 Derandomized Color Coding with Two-Stage Hashing

Require: Graph $G = (V, E)$ with vertices $V = \{1, \ldots, n\}$, target path length k

Ensure: Return true if a colorful path of length k exists

```
Build TwoStageKPerfectHash(k, n)
for each i in stage-1 hashes do
    for each j in stage-2 hashes do
        Assign colors c(v) ← TwoStageHash(v, i, j) + 1 for all v ∈ V
        if HasColorfulPath(c) then
            return true
        end if
    end for
end for
return false
function HasColorfulPath(c)
    Initialize DP[v][S] ← false for all v ∈ V, S ⊆ {1, ..., k}
    for each v ∈ V do
        DP[v][{c(v)}] ← true
    end for
    for len = 1 to k − 1 do
        for each v ∈ V do
            for each S s.t. |S| = len and DP[v][S] = true do
                for each neighbor u of v with c(u) ∉ S do
                    DP[u][S ∪ {c(u)}] ← true
                end for
            end for
        end for
    end for
    for each v ∈ V do
        if DP[v][{1, 2, ..., k}] = true then
            return true
        end if
    end for
    return false
end function
```

7 Analysis

Results for different hash functions and their performance on different graph types are presented.

For $K = 8$, the algorithm runs in $O(n^2)$ time. The two-stage hash family construction is efficient, and the dynamic programming approach ensures that we can find colorful paths in linear time.

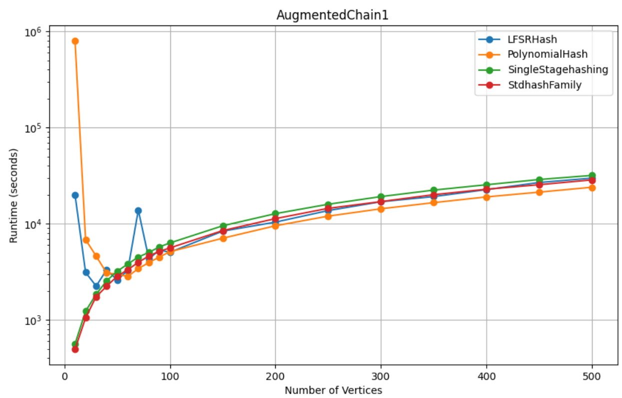

Figure 1: Augmented Chain Graph

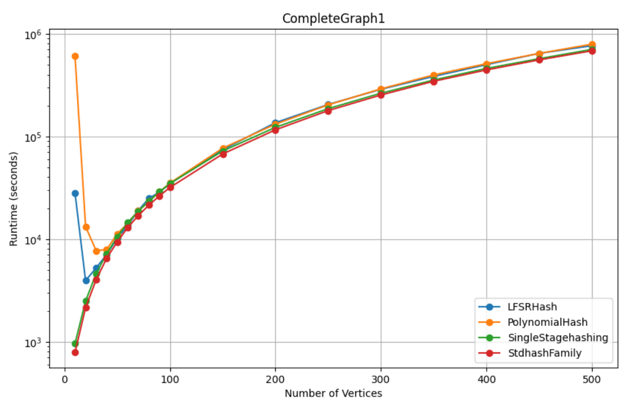

Figure 2: Complete Graph

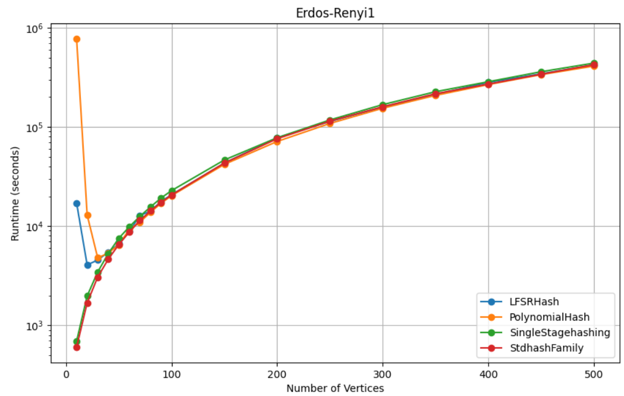

Figure 3: Erdos-Renyi Graph

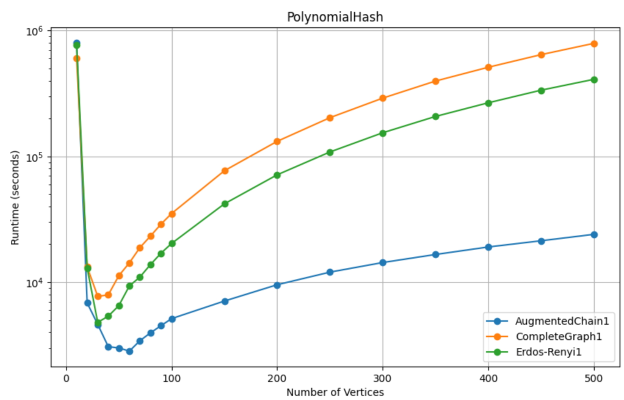

Figure 4: Polynomial Hash Function

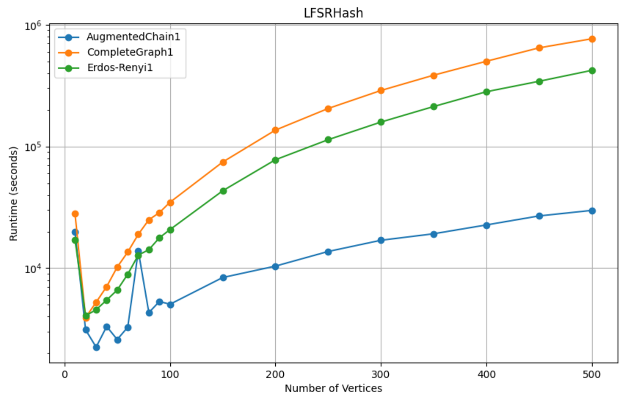

Figure 5: LFSR Hash Function

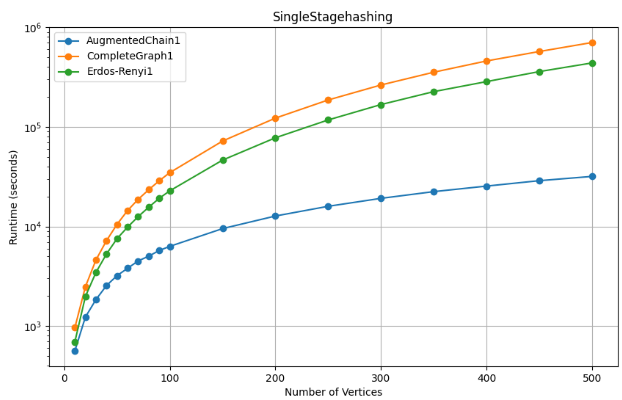

Figure 6: Single Stage Hashing

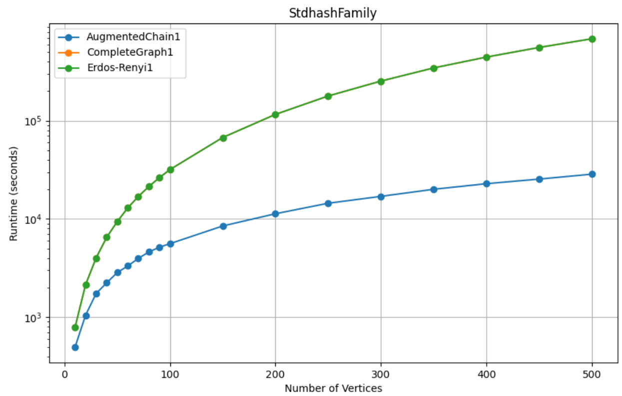

Figure 7: Standard Hash Family

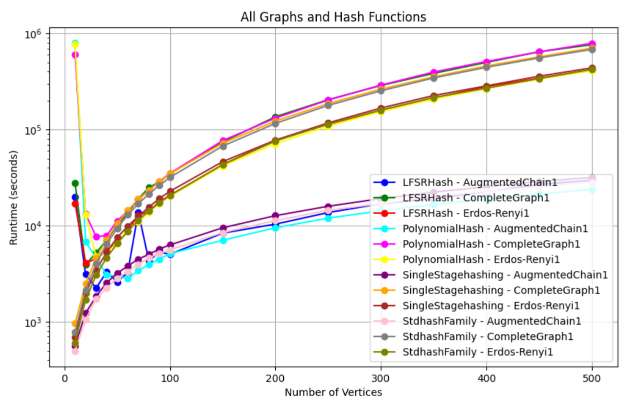

Figure 8: All Graphs and Hash Functions

8 Conclusion

We have described a derandomized color coding algorithm using two-stage k-perfect hash families. This approach efficiently finds colorful paths in graphs and can be extended to other subgraph detection problems.

References