Faster Edge Coloring by Partition Sieving

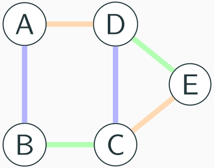
Shyan Akmal (INSAIT, Sofia University)
Tomohiro Koana (Utrecht Univerity & Kyoto University)

STACS 2025

Special thanks to Christian Komusiewicz (Friedrich Schiller University Jena) for travel support.

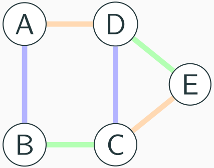
Edge Coloring

Definition: An *edge coloring* of a graph $G = (V, E)$ is an assignment of colors to E so that no two adjacent edges share the same color.



Edge Coloring

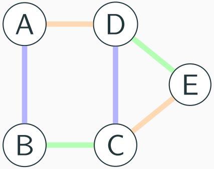
Definition: An *edge coloring* of a graph $G = (V, E)$ is an assignment of colors to E so that no two adjacent edges share the same color.



Chromatic Index $\chi'(G)$: the minimum number of colors needed to edge-color G.

Edge Coloring

Definition: An *edge coloring* of a graph $G = (V, E)$ is an assignment of colors to E so that no two adjacent edges share the same color.



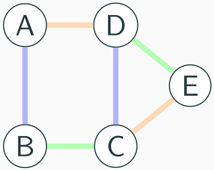
Chromatic Index $\chi'(G)$: the minimum number of colors needed to edge-color G.

Vizing's Theorem: For any simple graph G with maximum degree $\Delta(G)$,

$$\Delta(G) \leq \chi'(G) \leq \Delta(G) + 1.$$

Edge Coloring

Definition: An *edge coloring* of a graph $G = (V, E)$ is an assignment of colors to E so that no two adjacent edges share the same color.



Chromatic Index $\chi'(G)$: the minimum number of colors needed to edge-color G.

Vizing's Theorem: For any simple graph G with maximum degree $\Delta(G)$,

$$\Delta(G) \leq \chi'(G) \leq \Delta(G) + 1.$$

NP-hardness: Determining the chromatic index is NP-hard even for $\Delta = 3$.

Motivation: Runtime

Edge Coloring: A fundamental special case of graph coloring (on line graphs).

Motivation: Runtime

Edge Coloring: A fundamental special case of graph coloring (on line graphs).

State of the art: Coloring can be solved in $O^*(2^n)$ time via subset convolution.

Motivation: Runtime

Edge Coloring: A fundamental special case of graph coloring (on line graphs).

State of the art: Coloring can be solved in $O^*(2^n)$ time via subset convolution.

Central question:

Can Coloring be solved in $O^*((2-\varepsilon)^n)$ time (when k is fixed)?

Motivation: Runtime

Edge Coloring: A fundamental special case of graph coloring (on line graphs).

State of the art: Coloring can be solved in $O^*(2^n)$ time via subset convolution.

Central question:

$$\text{Can Coloring be solved in } O^*((2-\varepsilon)^n) \text{ time (when } k \text{ is fixed)?}$$

Implications for Edge Coloring:

- Since vertex coloring can be solved in $O^*(2^n)$ time, the reduction implies that edge coloring can be solved in $O^*(2^m)$ time (but requires exponential space).

Motivation: Runtime

Edge Coloring: A fundamental special case of graph coloring (on line graphs).

State of the art: Coloring can be solved in $O^*(2^n)$ time via subset convolution.

Central question:

Can Coloring be solved in $O^*((2-\varepsilon)^n)$ time (when k is fixed)?

Implications for Edge Coloring:

- Since vertex coloring can be solved in $O^*(2^n)$ time, the reduction implies that edge coloring can be solved in $O^*(2^m)$ time (but requires exponential space).
- For bounded-degree graphs, a refined subset convolution technique yields an $O^*(2^{(1-\varepsilon)m})$-time algorithm, where $\varepsilon = 1/2^{\Theta(\Delta)}$.

Motivation: Space

Bjorklund et al. [JCSS 2017] developed randomized polynomial-space solutions for edge coloring:

- For general graphs, edge coloring can be solved in $O^*(2^m)$ time.
- For regular graphs, edge colroing can be solved in $O^*(2^{m-n/2})$ time.

Motivation: Space

Bjorklund et al. [JCSS 2017] developed randomized polynomial-space solutions for edge coloring:

- For general graphs, edge coloring can be solved in $O^*(2^m)$ time.
- For regular graphs, edge colroing can be solved in $O^*(2^{m-n/2})$ time.

Our question:

Can edge coloring be solved faster than $O^*(2^m)$ time and polynomial space?

Motivation: Space

Bjorklund et al. [JCSS 2017] developed randomized polynomial-space solutions for edge coloring:

- For general graphs, edge coloring can be solved in $O^*(2^m)$ time.
- For regular graphs, edge colroing can be solved in $O^*(2^{m-n/2})$ time.

Our question:

Can edge coloring be solved faster than $O^*(2^m)$ time and polynomial space?

Our contribution:

Edge coloring can be solved in randomized $O^*(2^{m-3n/5})$ time and polynomial space.

Outline

Algorithm outline:

- **Step 1. Polynomial construction:**
  Design a polynomial that can be efficiently evaluted.
- **Step 2. Sieving:**
  We test whether a monomial satisfying certain properties exists.

Outline

Algorithm outline:

- **Step 1. Polynomial construction:**
  Design a polynomial that can be efficiently evaluted.
- **Step 2. Sieving:**
  We test whether a monomial satisfying certain properties exists.

Talk:

- We first review a simpler $O^*(2^m)$ time algorithm.
- We then discuss our improved algorithm, which refines both steps.

$O^*(2^m)$-time algorithm for edge coloring: Polynomial

Polynomial construction.

Define a variable x_e for each edge $e \in E$; let $X = \{x_e\}_{e \in E}$.

Define a polynomial $P(X)$ over a field $\mathbb{F}$ of characteristic 2:

$$P(X) = \sum_{M_1,\ldots,M_k} \prod_{i=1}^{k} \prod_{e \in M_i} x_e,$$

where $M_1, \ldots, M_k$ are matchings with $|M_1| + \cdots + |M_k| = m$.

$O^*(2^m)$-time algorithm for edge coloring: Polynomial

Polynomial construction.

Define a variable x_e for each edge $e \in E$; let $X = \{x_e\}_{e \in E}$.

Define a polynomial $P(X)$ over a field $\mathbb{F}$ of characteristic 2:

$$P(X) = \sum_{M_1, \dots, M_k} \prod_{i=1}^{k} \prod_{e \in M_i} x_e,$$

where $M_1, \dots, M_k$ are matchings with $|M_1| + \cdots + |M_k| = m$.

Edge coloring can be reformulated as: Is there a collection of k matchings $M_1, \dots, M_k$ that covers the graph, i.e., $M_1 \cup \cdots \cup M_k = E$?

This is equivalent to checking whether $P(X)$ contains the monomial $\prod_{e \in E} x_e$.

$O^*(2^m)$-time algorithm for edge coloring: Polynomial

Polynomial construction.

Define a variable x_e for each edge $e \in E$; let $X = \{x_e\}_{e \in E}$.

Define a polynomial $P(X)$ over a field $\mathbb{F}$ of characteristic 2:

$$P(X) = \sum_{M_1,\dots,M_k} \prod_{i=1}^{k} \prod_{e \in M_i} x_e,$$

where $M_1, \dots, M_k$ are matchings with $|M_1| + \cdots + |M_k| = m$.

Edge coloring can be reformulated as: Is there a collection of k matchings $M_1, \dots, M_k$ that covers the graph, i.e., $M_1 \cup \cdots \cup M_k = E$?

This is equivalent to checking whether $P(X)$ contains the monomial $\prod_{e \in E} x_e$.

Polynomial Evaluation.

Given $\mathbf{a} = \{a_e \in \mathbb{F} \mid e \in E\}$, we can evaluate $P(\mathbf{a})$ in polynomial time since $P(X)$ can be expressed as a product of the Pfaffians of the Tutte matrix.

A monomial $x_1^{d_1} x_2^{d_2} \cdots x_n^{d_n}$ is multilinear if each individual degree is at most 1 (i.e., $d_i \leq 1$ for all i).

$O^*(2^m)$-time algorithm for edge coloring: Sieving

A monomial $x_1^{d_1} x_2^{d_2} \cdots x_n^{d_n}$ is multilinear if each individual degree is at most 1 (i.e., $d_i \leq 1$ for all i).

Multilinear sieving. [Björklund et al., JCSS 2017]

For a polynomial $P(X)$ over a field of char. 2, we can determine whether $P(X)$ contains a multilinear monomial of degree ℓ using randomized $O^*(2^\ell)$ evaluations of $P(X)$.

$O^*(2^m)$-time algorithm for edge coloring: Sieving

A monomial $x_1^{d_1} x_2^{d_2} \cdots x_n^{d_n}$ is multilinear if each individual degree is at most 1 (i.e., $d_i \leq 1$ for all i).

Multilinear sieving. [Björklund et al., JCSS 2017]

For a polynomial $P(X)$ over a field of char. 2, we can determine whether $P(X)$ contains a multilinear monomial of degree ℓ using randomized $O^*(2^\ell)$ evaluations of $P(X)$.

Our goal is to determine whether $P(X)$ contains the monomial $\prod_{e \in E} x_e$, where

$$P(X) = \sum_{M_1,\ldots,M_k} \prod_{i=1}^{k} \prod_{e \in M_i} x_e.$$

We test whether $P(X)$ contains a multilinear term of degree m.

$O^*(2^m)$-time algorithm for edge coloring: Sieving

A monomial $x_1^{d_1} x_2^{d_2} \cdots x_n^{d_n}$ is multilinear if each individual degree is at most 1 (i.e., $d_i \leq 1$ for all i).

Multilinear sieving. [Björklund et al., JCSS 2017]

For a polynomial $P(X)$ over a field of char. 2, we can determine whether $P(X)$ contains a multilinear monomial of degree ℓ using randomized $O^*(2^\ell)$ evaluations of $P(X)$.

Our goal is to determine whether $P(X)$ contains the monomial $\prod_{e \in E} x_e$, where

$$P(X) = \sum_{M_1, \ldots, M_k} \prod_{i=1}^{k} \prod_{e \in M_i} x_e.$$

We test whether $P(X)$ contains a multilinear term of degree m.

Theorem: Edge coloring can be solved in randomized $O^*(2^m)$ time and poly. space.

Our algorithm: Polynomial

We refine the formulation of $P(X)$ while ensuring efficient evaluation.

We define

$$P(X) = \sum_{M_1,\ldots,M_k} \prod_{i=1}^{k} \prod_{e \in M_i} x_e,$$

Our algorithm: Polynomial

We refine the formulation of $P(X)$ while ensuring efficient evaluation.

We define

$$P(X) = \sum_{M_1,\dots,M_k} \prod_{i=1}^{k} \prod_{e \in M_i} x_e,$$

where $M_1, \dots, M_k$ are matchings satisfying the following additional condition: for each vertex v, every x_e corresponding to an edge e incident to v appears exactly $\deg(v)$ times across the M_i-matchings.

Our algorithm: Polynomial

We refine the formulation of $P(X)$ while ensuring efficient evaluation.

We define

$$P(X) = \sum_{M_1,\ldots,M_k} \prod_{i=1}^{k} \prod_{e \in M_i} x_e,$$

where $M_1, \ldots, M_k$ are matchings satisfying the following additional condition: for each vertex v, every x_e corresponding to an edge e incident to v appears exactly $\deg(v)$ times across the M_i-matchings.

Given $\mathbf{a} = \{a_e \in \mathbb{F} \mid e \in E\}$, we can evaluate $P(\mathbf{a})$ in polynomial time using a generalization of the Cauchy-Binet formula to skew-symmetric matrices (known as the Ishikawa-Wakayama formula).

Our algorithm: Partition Sieving

Let X be a set of variables, and let $P(X)$ be a polynomial.

Let $\mathcal{X} = X_1 \sqcup \cdots \sqcup X_p$ be a partition of X.
Let $\mathbf{d} = (d_1, \ldots, d_p)$ be a tuple of positive integers.

Our algorithm: Partition Sieving

Let X be a set of variables, and let $P(X)$ be a polynomial.

Let $\mathcal{X} = X_1 \sqcup \cdots \sqcup X_p$ be a partition of X.
Let $\mathbf{d} = (d_1, \ldots, d_p)$ be a tuple of positive integers.

We say that $P(X)$ is *compatible with* $(\mathcal{X}, \mathbf{d})$ if, for each $i \in [p]$ and every monomial m in $P(X)$, the degree of m restricted to X_i is exactly d_i.

Our algorithm: Partition Sieving

Let X be a set of variables, and let $P(X)$ be a polynomial.

Let $\mathcal{X} = X_1 \sqcup \cdots \sqcup X_p$ be a partition of X.
Let $\mathbf{d} = (d_1, \ldots, d_p)$ be a tuple of positive integers.

We say that $P(X)$ is *compatible with* $(\mathcal{X}, \mathbf{d})$ if, for each $i \in [p]$ and every monomial m in $P(X)$, the degree of m restricted to X_i is exactly d_i.

Our result (partition sieving):
For a polynomial $P(X)$ over a field of char. 2 that is compatible with $(\mathcal{X}, \mathbf{d})$, we can determine whether $P(X)$ contains a multilinear monomial of degree ℓ using randomized $O^*(2^{\ell-p})$ evaluations of $P(X)$.

Our algorithm: Partition Sieving

Let X be a set of variables, and let $P(X)$ be a polynomial.

Let $\mathcal{X} = X_1 \sqcup \cdots \sqcup X_p$ be a partition of X.
Let $\mathbf{d} = (d_1, \ldots, d_p)$ be a tuple of positive integers.

We say that $P(X)$ is *compatible with* $(\mathcal{X}, \mathbf{d})$ if, for each $i \in [p]$ and every monomial m in $P(X)$, the degree of m restricted to X_i is exactly d_i.

Our result (partition sieving):
For a polynomial $P(X)$ over a field of char. 2 that is compatible with $(\mathcal{X}, \mathbf{d})$, we can determine whether $P(X)$ contains a multilinear monomial of degree ℓ using randomized $O^*(2^{\ell - p})$ evaluations of $P(X)$.

This result is based on the *determinantal sieving* framework.

[Eiben, Koana, and Wahlström, SODA 24]

Our algorithm: Putting Everything Together

Preprocessing: First, delete all degree-1 vertices so the minimum degree is ≥ 2.

Our algorithm: Putting Everything Together

Preprocessing: First, delete all degree-1 vertices so the minimum degree is ≥ 2.

Structural result: A graph with ≥ 8 vertices has a dominating set of size $\leq 2n/5$ [McCuaig & Shepherd J. Graph Theorey 89].

Our algorithm: Putting Everything Together

Preprocessing: First, delete all degree-1 vertices so the minimum degree is ≥ 2.

Structural result: A graph with ≥ 8 vertices has a dominating set of size $\leq 2n/5$ [McCuaig & Shepherd J. Graph Theorey 89].

Partitioning strategy: Let D be a dominating set of size $\geq 2n/5$, and let $C = V \setminus D$. We partition the edges E_X acorss C and D based on their incidence with vertices in C, defining a partition $\{\partial_{E_X}(v)\}_{v \in C}$ of E_X with $\geq 3n/5$ parts.

Our algorithm: Putting Everything Together

Preprocessing: First, delete all degree-1 vertices so the minimum degree is ≥ 2.

Structural result: A graph with ≥ 8 vertices has a dominating set of size $\leq 2n/5$ [McCuaig & Shepherd J. Graph Theorey 89].

Partitioning strategy: Let D be a dominating set of size $\geq 2n/5$, and let $C = V \setminus D$. We partition the edges E_X acorss C and D based on their incidence with vertices in C, defining a partition $\{\partial_{E_X}(v)\}_{v \in C}$ of E_X with $\geq 3n/5$ parts.

We apply partition sieiving to $P(X)$ with degree $|\partial_{E_X}(v)|$ for each $v \in C$. Compatibility is guaranteed by the polynomial design.

Our algorithm: Putting Everything Together

Preprocessing: First, delete all degree-1 vertices so the minimum degree is ≥ 2.

Structural result: A graph with ≥ 8 vertices has a dominating set of size $\leq 2n/5$

[McCuaig & Shepherd J. Graph Theorey 89].

Partitioning strategy: Let D be a dominating set of size $\geq 2n/5$, and let $C = V \setminus D$. We partition the edges E_X acorss C and D based on their incidence with vertices in C, defining a partition $\{\partial_{E_X}(v)\}_{v \in C}$ of E_X with $\geq 3n/5$ parts.

We apply partition sieiving to $P(X)$ with degree $|\partial_{E_X}(v)|$ for each $v \in C$. Compatibility is guaranteed by the polynomial design.

Final theorem: Edge coloring can be solved in randomized $O^*(2^{m-3n/5})$ time.

Concluding remarks

Additional Results.

- For a d-regular graph, a dominating set of size $(H_{d+1}/(d+1))n$ exists, where H_k is the k-th harmonic number. This implies that edge coloring can be solved in $O^*(2^{m-\frac{H_{d+1}}{d+1}n})$ time, which approaches $O^*(2^{m-n})$ as $d \to \infty$.
- Our result extends to the list version of edge coloring.

Concluding remarks

Additional Results.

- For a d-regular graph, a dominating set of size $(H_{d+1}/(d+1))n$ exists, where H_k is the k-th harmonic number. This implies that edge coloring can be solved in $O^*(2^{m-\frac{H_{d+1}}{d+1}n})$ time, which approaches $O^*(2^{m-n})$ as $d \to \infty$.
- Our result extends to the list version of edge coloring.

Open Questions.

- Other applications of partition sieving?
- Is there an algorithm for edge coloring running in $O^*(1.9999^m)$ time?
- Can edge coloring be solved in $O^*(2^{m-n})$ time?
- Currently, only a $2^{\Omega(n)}$ time lower bound under ETH is known.
  Can we establish a $2^{\Omega(m)}$ or $2^{\Omega(n \log n)}$ time lower bound for dense graphs?

Concluding remarks

Additional Results.

- For a d-regular graph, a dominating set of size $(H_{d+1}/(d+1))n$ exists, where H_k is the k-th harmonic number. This implies that edge coloring can be solved in $O^*(2^{m-\frac{H_{d+1}}{d+1}n})$ time, which approaches $O^*(2^{m-n})$ as $d \to \infty$.
- Our result extends to the list version of edge coloring.

Open Questions.

- Other applications of partition sieving?
- Is there an algorithm for edge coloring running in $O^*(1.9999^m)$ time?
- Can edge coloring be solved in $O^*(2^{m-n})$ time?
- Currently, only a $2^{\Omega(n)}$ time lower bound under ETH is known.
  Can we establish a $2^{\Omega(m)}$ or $2^{\Omega(n \log n)}$ time lower bound for dense graphs?

Thank you