Power of Algebraic Methods in Designing Algorithms for Graph Problems

Abhranil Chatterjee, IIT Kanpur

CALDAM Indo-Spanish Pre-Conference School on Algorithms and Combinatorics, February 11, 2025

This Talk

Plan of the Talk:

- **Introduction**
- **Bipartite Perfect Matching**
- **Longest Path**

This Talk

Plan of the Talk:

- **Introduction**
- **Bipartite Perfect Matching**
- **Longest Path**

Mantra. **Algebra Powers Computation.**

This Talk

Plan of the Talk:

- **Introduction** algorithm design using algebraic methods
- **Bipartite Perfect Matching**
- **Longest Path**

Mantra. **Algebra Powers Computation.**

This Talk

Plan of the Talk:

- **Introduction**
- **Bipartite Perfect Matching** designing efficient parallel algorithm
- **Longest Path**

Mantra. **Algebra Powers Computation.**

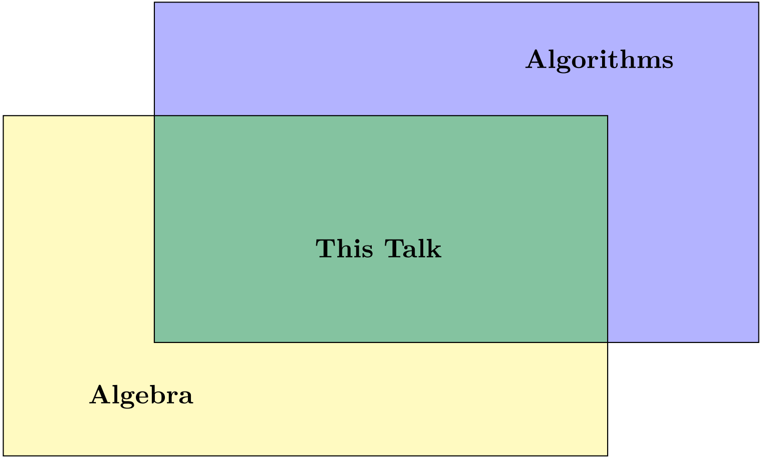
This Talk

Plan of the Talk:

- **Introduction**
- **Bipartite Perfect Matching**
- **Longest Path** designing efficient FPT algorithm

Mantra. **Algebra Powers Computation.**

This Talk



Polynomials

- **Univariate Polynomial:**

$$P(x) = 3x^4 - 2x^3 + x^2 + 5x - 7$$

A univariate polynomial of degree d can have at most d many roots.

- **Multivariate Polynomial:**

$$Q(x, y) = 2x^3y^2 - xy + 4x^2 - 3y^3 + 6$$

$$Q(x_1, \ldots, x_n) = \sum_{(e_1, \ldots, e_n)} \alpha_{e_1, \ldots, e_n} \cdot x_1^{e_1} \cdots x_n^{e_n} \qquad \text{where, } e_1 + \ldots + e_n \leq d.$$

A term $x_1^{e_1} \cdots x_n^{e_n}$ is square-free $\iff \forall i,\ e_i$ is either 0 or 1.

- x^3y^2 is **not** square-free but xy is square-free.

Matrices

Determinant:

$$\det(M) = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}(a_{22}a_{33} - a_{23}a_{32}) - a_{12}(a_{21}a_{33} - a_{23}a_{31}) + a_{13}(a_{21}a_{32} - a_{22}a_{31}).$$

Matrices

Determinant:

$$\det(M) = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}(a_{22}a_{33} - a_{23}a_{32}) - a_{12}(a_{21}a_{33} - a_{23}a_{31}) + a_{13}(a_{21}a_{32} - a_{22}a_{31}).$$

General Formula: $$\det(M) = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} a_{i,\sigma(i)}$$

Matrices

Determinant:

$$\det(M) = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}(a_{22}a_{33} - a_{23}a_{32}) - a_{12}(a_{21}a_{33} - a_{23}a_{31}) + a_{13}(a_{21}a_{32} - a_{22}a_{31}).$$

General Formula: $\det(M) = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} a_{i,\sigma(i)}$

- A matrix M is **invertible** $\iff \det M \neq 0$.
- The **rank** of a matrix is the size of maximum invertible submatrix.

Algebraic Algorithm: General Template

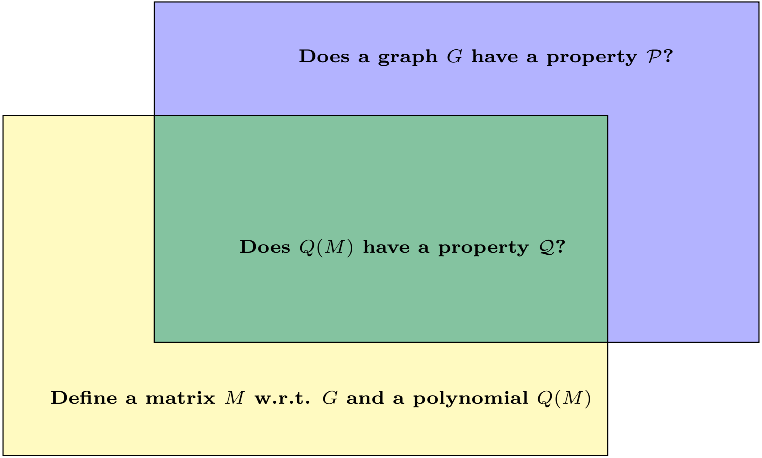
Does a graph G have a property $\mathcal{P}$?

Algebraic Algorithm: General Template

Does a graph G have a property $\mathcal{P}$?

Define a matrix M w.r.t. G and a polynomial $Q(M)$

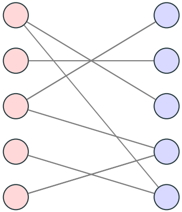
Algebraic Algorithm: General Template

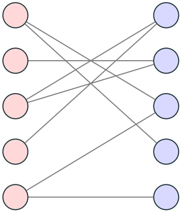


Bipartite Perfect Matching

Perfect Matching in Bipartite Graphs

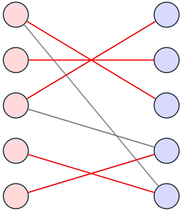
A **perfect matching** in a bipartite graph $G = (L \cup R, E)$, $(L = R = [n])$ is a subset $M \subseteq E$ s.t. every vertex is incident to exactly one edge in M.

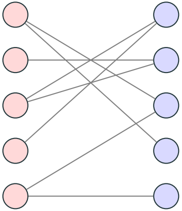




Perfect Matching in Bipartite Graphs

A **perfect matching** in a bipartite graph $G = (L \cup R, E)$, $(L = R = [n])$ is a subset $M \subseteq E$ s.t. every vertex is incident to exactly one edge in M.





Bipartite Perfect Matching: Known Results

- **Polynomial-time algorithm.**

Bipartite Perfect Matching: Known Results

- **Polynomial-time algorithm.** using augmenting path idea

Bipartite Perfect Matching: Known Results

- **Polynomial-time algorithm.**

Can we do better?

Designing Parallel Algorithm

- **Efficient Parallel Algorithm:** solvable in **poly-log time** using **polynomially many processors.**
- For example, **matrix multiplication**, and the **determinant computation** in NC.

Designing Parallel Algorithm

- **Efficient Parallel Algorithm:** solvable in **poly-log time** using **polynomially many processors.**
- For example, **matrix multiplication**, and the **determinant computation** in NC.

Open problem: Design an efficient parallel algorithm for bipartite perfect matching.

Designing Parallel Algorithm

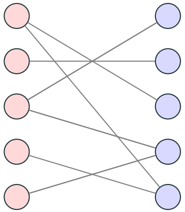
- **Efficient Parallel Algorithm:** solvable in **poly-log time** using **polynomially many processors.** Complexity class: NC.
- For example, **matrix multiplication**, and the **determinant computation** in NC.

Open problem: Design an efficient parallel algorithm for bipartite perfect matching.

Bipartite Perfect Matching: an Algebraic Solution

Consider a bipartite graph $G = (L \cup R, E)$ s.t. $L = R = [n]$ and $E \subseteq L \times R$ and define a symbolic matrix A,

$$A_{i,j} = \begin{cases} x_{i,j}, & (i,j) \in E \\ 0, & (i,j) \notin E \end{cases}$$

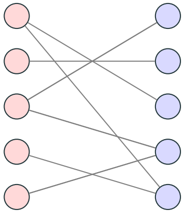


$$A = \begin{bmatrix} 0 & 0 & x_{1,3} & 0 & x_{1,5} \\ 0 & x_{2,2} & 0 & 0 & 0 \\ x_{3,1} & 0 & 0 & x_{3,4} & 0 \\ 0 & 0 & 0 & 0 & x_{4,5} \\ 0 & 0 & 0 & x_{5,4} & 0 \end{bmatrix}.$$

Bipartite Perfect Matching: an Algebraic Solution

Consider a bipartite graph $G = (L \cup R, E)$ s.t. $L = R = [n]$ and $E \subseteq L \times R$ and define a symbolic matrix A,

$$A_{i,j} = \begin{cases} x_{i,j}, & (i,j) \in E \\ 0, & (i,j) \notin E \end{cases}$$



$$A = \begin{bmatrix} 0 & 0 & x_{1,3} & 0 & x_{1,5} \\ 0 & x_{2,2} & 0 & 0 & 0 \\ x_{3,1} & 0 & 0 & x_{3,4} & 0 \\ 0 & 0 & 0 & 0 & x_{4,5} \\ 0 & 0 & 0 & x_{5,4} & 0 \end{bmatrix}.$$

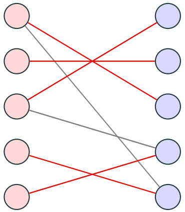
Theorem (Tutte, Edmonds, Lovász ...)

G has a perfect matching $\iff$ $\det A$ is nonzero.

Bipartite Perfect Matching: an Algebraic Solution

Consider a bipartite graph $G = (L \cup R, E)$ s.t. $L = R = [n]$ and $E \subseteq L \times R$ and define a symbolic matrix A,

$$A_{i,j} = \begin{cases} x_{i,j}, & (i,j) \in E \\ 0, & (i,j) \notin E \end{cases}$$

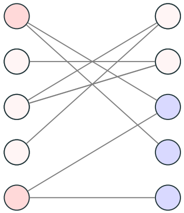


$$A = \begin{bmatrix} 0 & 0 & x_{1,3} & 0 & 0 \\ 0 & x_{2,2} & 0 & 0 & 0 \\ x_{3,1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & x_{4,5} \\ 0 & 0 & 0 & x_{5,4} & 0 \end{bmatrix}.$$

Theorem (Tutte, Edmonds, Lovász ...)

G has a perfect matching $\iff$ $\det A$ is nonzero.

Bipartite Perfect Matching

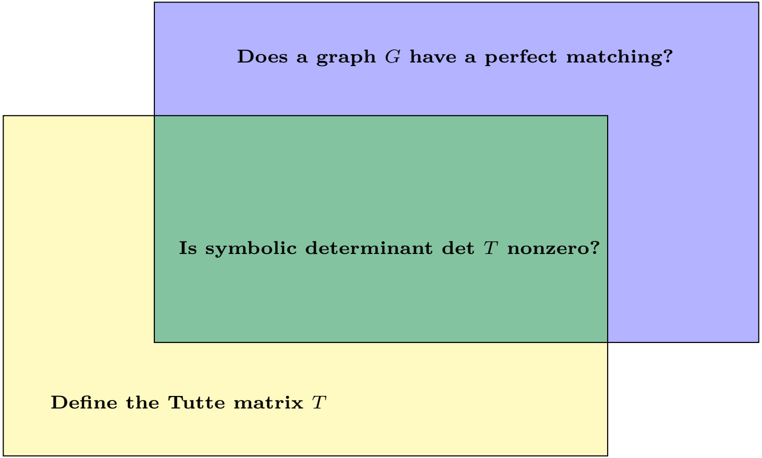


$$A = \begin{bmatrix} 0 & 0 & x_{1,3} & x_{1,4} & 0 \\ 0 & x_{2,2} & 0 & 0 & 0 \\ x_{3,1} & x_{3,2} & 0 & 0 & 0 \\ x_{4,1} & 0 & 0 & 0 & 0 \\ 0 & 0 & x_{5,3} & 0 & x_{5,5} \end{bmatrix}.$$

Theorem (Tutte, Edmonds, Lovász ...)

G has a perfect matching $\iff \det A$ is nonzero.

Bipartite Perfect Matching



Open Problem

Open problem: Design an efficient deterministic parallel algorithm for bipartite perfect matching.

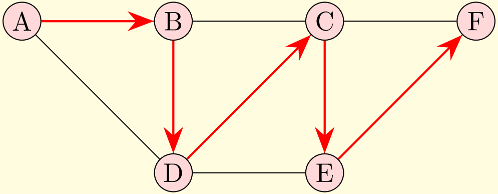
Longest Path

Longest Path in a Graph

Given a graph G, find a **longest** path in G.

Longest Path in a Graph

Given a graph G, find a **longest** path in G.



- The longest path in this graph is A $\rightarrow$ B $\rightarrow$ D $\rightarrow$ C $\rightarrow$ E $\rightarrow$ F.

Longest Path: Known Results

- Shortest Path —> **easy**

  Longest Path —> **hard**.

Longest Path: Known Results

- Shortest Path —> **easy**

  Longest Path —> **hard**.

  Does there exist a path of length exactly k in graph G?

Longest Path: Known Results

- Shortest Path —> **easy**

  Longest Path —> **hard**.

  Does there exist a path of length exactly k in graph G?

- Hardness follows from Hamiltonian path.

Longest Path: Known Results

- **Special Cases**: Longest Path for special graph classes.
- **Approximation Algorithms:** An efficient but inexact solution.
- **FPT Algorithm:** An inefficient but exact solution.

Longest Path: Known Results

- **Color coding:** $4.32^k \text{poly}(n)$ [Alon et al., Huffman]
- **Representative Set:** $2.6^k \text{poly}(n)$ [Fomin et al.]
- **Best Known:** $2.55^k \text{poly}(n)$ [Tsur]

Longest Path: Known Results

- **Color coding:** $4.32^k \text{poly}(n)$ [Alon et al., Huffman]
- **Representative Set:** $2.6^k \text{poly}(n)$ [Fomin et al.]
- **Best Known:** $2.55^k \text{poly}(n)$ [Tsur]

Open problem: Design a deterministic 2^k.poly(n)-time algorithm to find a path of length k in a graph G of size n.

Algebraic Approach

Encode the k-length walks in G by a degree-k polynomial.

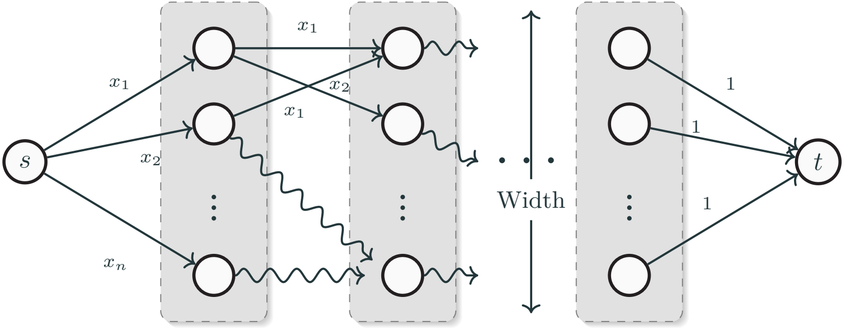
Algebraic Approach

Encode the k-length walks in G by a degree-k polynomial.

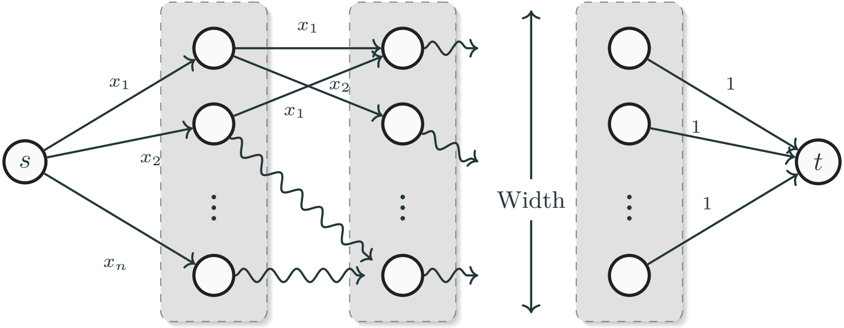
For a graph G of size n, define $n \times n$ matrix A_G:

$$A_G = \begin{cases} x_j, & (i,j) \in E(G) \\ 0, & otherwise. \end{cases}$$

Algebraic Approach

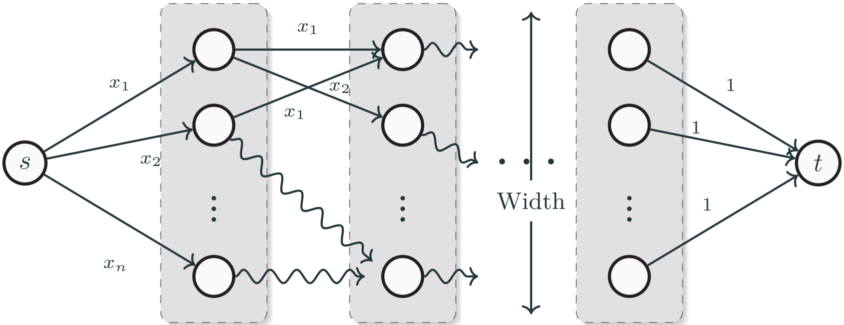


Algebraic Approach



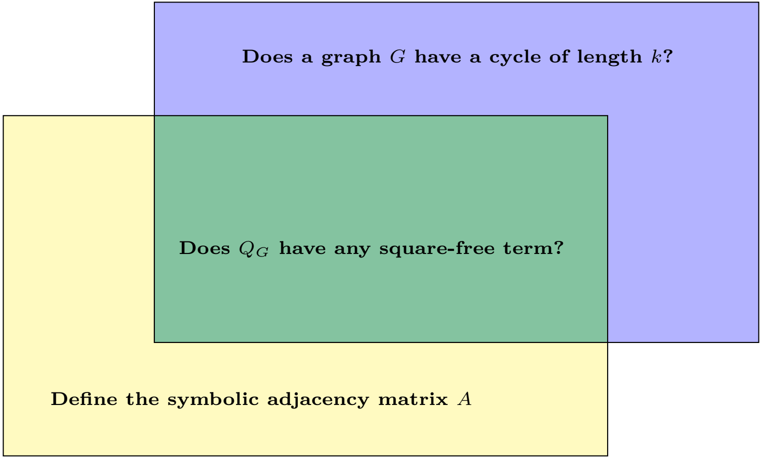
Consider the polynomial $Q_G = (x_1, \ldots, x_n)^T \cdot A_G^{k-1} \cdot \overline{1}$.

Algebraic Approach



A k-path in G corresponds to a square-free term in Q_G.

Longest Cycle



Algebraic Approach

- ***k*-th Elementary Symmetric Polynomial:** Polynomial encoding all k-length **paths**:

$$e_{n,k} = \sum_{T \subseteq [n], |T|=k} \prod_{i \in T} x_i.$$

- Does there exist a **common term**?

- Addressed in the works of:
  - **Pratt [2019]**,
  - **Arvind, C., Datta, Mukhopadhyay [2019]**,
  - **Brand, Pratt [2021]**.

Disjointness Matrix

Define a matrix of size $\binom{n}{k/2}$ such that,

$$D[S,T] = \begin{cases} 1, & S \cap T = \emptyset \\ 0, & otherwise. \end{cases}$$

Complexity depends on the **rank of the disjointness matrix** D.

Disjointness Matrix

Define a matrix of size $\binom{n}{k/2}$ such that,

$$D[S,T] = \begin{cases} 1, & S \cap T = \emptyset \\ 0, & otherwise. \end{cases}$$

Complexity depends on the **rank of the disjointness matrix** D.

Bad News: Disjointness matrix is of full rank.

Algebraic Approach

- ***k*-th Elementary Symmetric Polynomial:** Polynomial encoding all k-length **paths**:

$$\widehat{e}_{n,k} = \sum_{T \subseteq [n], |T|=k} \prod_{i \in T} \alpha_T \cdot x_i.$$

- Does there exist a **common term**?

Weighted Disjointness Matrix

Define a matrix of size $\binom{n}{k/2}$ such that,

$$\widehat{D}[S,T] = \begin{cases} \alpha_{S\cup T}, & S \cap T = \emptyset \\ 0, & otherwise. \end{cases}$$

Complexity depends on the **rank of the weighted disjointness matrix** $\widehat{D}$.

Algebraic Approach

- **Rank**$(D) = 2^k$ [**ACDM'19**].

Algebraic Approach

- $\mathbf{Rank}(D) = 2^k$ [**ACDM'19**].
- $\alpha_T \not\succ 0 \implies$ randomized algorithm.

Algebraic Approach

- **Rank**$(D) = 2^k$ [**ACDM'19**].
- $\alpha_T \not> 0 \implies$ randomized algorithm.
- **Rank**$(D) \leq\ 2.62^k$ where $\forall T,\ \ \alpha_T > 0$ [**BP'21**].

Longest Path: Open Problem

Open problem: Design a 2^k.poly(n)-time algorithm to find a path of length k in a graph G of size n.

Open problem: Minimize the rank of a weighted disjointness matrix.

Summary

Open Problems:

1. **Efficient parallel algorithm for bipartite perfect matching**,
2. **Efficient FPT algorithm for longest path problem**,
3. **Rank of a weighted disjointness matrix**,

Main Idea:

1. **Reduce a graph problem to a problem on polynomials.**

Summary

Open Problems:

1. **Efficient parallel algorithm for bipartite perfect matching**,
   Is algebraic approach helpful?
2. **Efficient FPT algorithm for longest path problem**,
3. **Rank of a weighted disjointness matrix**,

Main Idea:

1. **Reduce a graph problem to a problem on polynomials.**

Summary

Open Problems:

1. **Efficient parallel algorithm for bipartite perfect matching**,
2. **Efficient FPT algorithm for longest path problem**,
3. **Rank of a weighted disjointness matrix**,
   an algebraic approach to design efficient FPT algorithm

Main Idea:

1. **Reduce a graph problem to a problem on polynomials.**

Thank You!