Paths, Cycles and Permanent

A bridge between Combinatorics and Algebra

Kishlaya Jaiswal
Advisor: Samir Datta

Chennai Mathematical Institute

June 4, 2021

Disjoint Paths

- $DP(k)$ problem: Given a graph G and k pairs of vertices $\{s_i, t_i\}_{i \leq k}$, find paths between each pair such that they are all pairwise disjoint.

Disjoint Paths

- $DP(k)$ problem: Given a graph G and k pairs of vertices $\{s_i, t_i\}_{i \leq k}$, find paths between each pair such that they are all pairwise disjoint.
- k **not fixed**: NP-hard even for undirected planar graphs

Disjoint Paths

- $DP(k)$ problem: Given a graph G and k pairs of vertices $\{s_i, t_i\}_{i \le k}$, find paths between each pair such that they are all pairwise disjoint.
- k **not fixed**: NP-hard even for undirected planar graphs
- So let k be **fixed** but still NP-hard for directed graphs even for $k = 2$

Disjoint Paths

- $DP(k)$ problem: Given a graph G and k pairs of vertices $\{s_i, t_i\}_{i \leq k}$, find paths between each pair such that they are all pairwise disjoint.
- k **not fixed**: NP-hard even for undirected planar graphs
- So let k be **fixed** but still NP-hard for directed graphs even for $k = 2$
- But for directed planar graphs, k-disjoint paths can be found in poly-time [Schrijver 1994]

Disjoint Paths

- $DP(k)$ problem: Given a graph G and k pairs of vertices $\{s_i, t_i\}_{i \le k}$, find paths between each pair such that they are all pairwise disjoint.
- k **not fixed**: NP-hard even for undirected planar graphs
- So let k be **fixed** but still NP-hard for directed graphs even for $k = 2$
- But for directed planar graphs, k-disjoint paths can be found in poly-time [Schrijver 1994]
- If G is undirected then k-disjoint paths can be found in $O(n^3)$ for any fixed k [Robertson, Seymour]

Disjoint Paths

- $DP(k)$ problem: Given a graph G and k pairs of vertices $\{s_i, t_i\}_{i \le k}$, find paths between each pair such that they are all pairwise disjoint.
- k **not fixed**: NP-hard even for undirected planar graphs
- So let k be **fixed** but still NP-hard for directed graphs even for $k = 2$
- But for directed planar graphs, k-disjoint paths can be found in poly-time [Schrijver 1994]
- If G is undirected then k-disjoint paths can be found in $O(n^3)$ for any fixed k [Robertson, Seymour]
- What about shortest disjoint paths in undirected graphs?

Disjoint Paths

- $DP(k)$ problem: Given a graph G and k pairs of vertices $\{s_i, t_i\}_{i \leq k}$, find paths between each pair such that they are all pairwise disjoint.
- k **not fixed**: NP-hard even for undirected planar graphs
- So let k be **fixed** but still NP-hard for directed graphs even for $k = 2$
- But for directed planar graphs, k-disjoint paths can be found in poly-time [Schrijver 1994]
- If G is undirected then k-disjoint paths can be found in $O(n^3)$ for any fixed k [Robertson, Seymour]
- What about shortest disjoint paths in undirected graphs?
- A randomized poly-time algorithm for shortest $DP(2)$ was presented by [Björklund, Husfeldt]

Permanent

- For any matrix A over integers, $\det(A) = \sum_{\sigma} \operatorname{sgn}(\sigma) \prod a_{i\sigma(i)}$

Permanent

- For any matrix A over integers, $\det(A) = \sum_{\sigma} \mathrm{sgn}(\sigma) \prod a_{i\sigma(i)}$
- Computing determinant is well-studied in the past and several fast (both parallel and sequential) algorithms are known.

Permanent

- For any matrix A over integers, $\det(A) = \sum_\sigma \operatorname{sgn}(\sigma) \prod a_{i\sigma(i)}$
- Computing determinant is well-studied in the past and several fast (both parallel and sequential) algorithms are known.
- Permanent is defined as the following algebraic analogue of determinant: $\mathrm{perm}(A) = \sum_\sigma \prod a_{i\sigma(i)}$

Permanent

- For any matrix A over integers, $\det(A) = \sum_\sigma \mathrm{sgn}(\sigma) \prod a_{i\sigma(i)}$
- Computing determinant is well-studied in the past and several fast (both parallel and sequential) algorithms are known.
- Permanent is defined as the following algebraic analogue of determinant: $\mathrm{perm}(A) = \sum_\sigma \prod a_{i\sigma(i)}$
- On the contrary, computing permanent is #P-hard [Valiant 1979]

Permanent

- For any matrix A over integers, $\det(A) = \sum_\sigma \mathrm{sgn}(\sigma) \prod a_{i\sigma(i)}$
- Computing determinant is well-studied in the past and several fast (both parallel and sequential) algorithms are known.
- Permanent is defined as the following algebraic analogue of determinant: $\mathrm{perm}(A) = \sum_\sigma \prod a_{i\sigma(i)}$
- On the contrary, computing permanent is #P-hard [Valiant 1979]
- Nevertheless, Valiant also showed that we can compute permanent mod 2^k in $O(n^{4k-3})$, using Gaussian elimination, which is highly sequential.

Permanent

- For any matrix A over integers, $\det(A) = \sum_\sigma \mathrm{sgn}(\sigma) \prod a_{i\sigma(i)}$
- Computing determinant is well-studied in the past and several fast (both parallel and sequential) algorithms are known.
- Permanent is defined as the following algebraic analogue of determinant: $\mathrm{perm}(A) = \sum_\sigma \prod a_{i\sigma(i)}$
- On the contrary, computing permanent is $\#$P-hard [Valiant 1979]
- Nevertheless, Valiant also showed that we can compute permanent mod 2^k in $O(n^{4k-3})$, using Gaussian elimination, which is highly sequential.
- A parallel algorithm to compute permanent $\pmod{2^k}$ was discovered by [Braverman, Kulkarni & Roy]

The Bridge

- The permanent of a matrix can be regarded as the weighted sum of cycle covers of a directed graph.

The Bridge

- The permanent of a matrix can be regarded as the weighted sum of cycle covers of a directed graph.
- Cycle cover of a graph is a subset of edges which is form vertex-disjoint directed cycles covering all the vertices.

The Bridge

- The permanent of a matrix can be regarded as the weighted sum of cycle covers of a directed graph.
- Cycle cover of a graph is a subset of edges which is form vertex-disjoint directed cycles covering all the vertices.
- Given a weighted directed graph (G, w) with vertices labelled $\{v_1, \ldots, v_n\}$, adjacency matrix A_G of G is defined as
$$(A_G)_{ij} = \begin{cases} w(e) & \text{if } e = (v_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases}$$

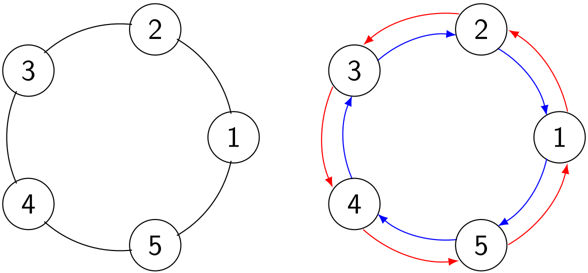
The Bridge

- The permanent of a matrix can be regarded as the weighted sum of cycle covers of a directed graph.
- Cycle cover of a graph is a subset of edges which is form vertex-disjoint directed cycles covering all the vertices.
- Given a weighted directed graph (G, w) with vertices labelled $\{v_1, \dots, v_n\}$, adjacency matrix A_G of G is defined as
  $(A_G)_{ij} = \begin{cases} w(e) & \text{if } e = (v_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases}$
- Denote by $w(C) = \prod_{e \in C} w(e)$ the weight of cycle cover then

$$\mathrm{perm}(A_G) = \sum_{\text{cycle covers } C} w(C)$$

The Bridge

Undirected Graphs



- We view undirected graph as a directed graph with each edge $\{u, v\}$ replaced with two directed edges (u, v) and (v, u) with $w((u, v)) = w((v, u)) = w(\{u, v\})$ (*symmetrical* weights)

Applications

Shortest 2-disjoint paths [BH]

- SDP(2): Given a undirected graph G and 2 pairs of vertices $\{s_i, t_i \mid i \in \{1, 2\}\}$, find disjoint paths between each pair such that the sum of the lengths of these paths is minimum.

Applications

Shortest 2-disjoint paths [BH]

- SDP(2): Given a undirected graph G and 2 pairs of vertices $\{s_i, t_i \mid i \in \{1, 2\}\}$, find disjoint paths between each pair such that the sum of the lengths of these paths is minimum.
- Assign weight x to each edge and add self-loops (of weight 1) on all vertices except terminals

Applications

Shortest 2-disjoint paths [BH]

- SDP(2): Given a undirected graph G and 2 pairs of vertices $\{s_i, t_i \mid i \in \{1,2\}\}$, find disjoint paths between each pair such that the sum of the lengths of these paths is minimum.
- Assign weight x to each edge and add self-loops (of weight 1) on all vertices except terminals
- To force cycles (in a cycle cover) to pass through the given terminals, we introduce *pattern graphs*.

Applications

Shortest 2-disjoint paths [BH]

- SDP(2): Given a undirected graph G and 2 pairs of vertices $\{s_i, t_i \mid i \in \{1, 2\}\}$, find disjoint paths between each pair such that the sum of the lengths of these paths is minimum.
- Assign weight x to each edge and add self-loops (of weight 1) on all vertices except terminals
- To force cycles (in a cycle cover) to pass through the given terminals, we introduce *pattern graphs*.
- A *pattern* P is an ordered pairing of the given terminals.

Applications

Shortest 2-disjoint paths [BH]

- SDP(2): Given a undirected graph G and 2 pairs of vertices $\{s_i, t_i \mid i \in \{1,2\}\}$, find disjoint paths between each pair such that the sum of the lengths of these paths is minimum.
- Assign weight x to each edge and add self-loops (of weight 1) on all vertices except terminals
- To force cycles (in a cycle cover) to pass through the given terminals, we introduce *pattern graphs*.
- A *pattern* P is an ordered pairing of the given terminals.
- A pattern graph G_P is same as G but such that if $(u, v) \in P$ then all outgoing edges from u, except edge (u, v), are deleted.

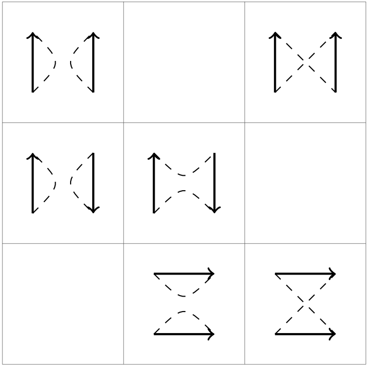
Applications

Shortest 2-disjoint paths [BH]

$$P_0 \quad \begin{array}{cc} t_1 & t_2 \\ \uparrow & \uparrow \\ s_1 & s_2 \end{array}$$

$$P_1 \quad \begin{array}{cc} t_1 & t_2 \\ \uparrow & \downarrow \\ s_1 & s_2 \end{array}$$

$$P_2 \quad \begin{array}{ccc} t_1 & \rightarrow & t_2 \\ & & \\ s_1 & \rightarrow & s_2 \end{array}$$



Smallest exponent with non-zero coefficient in $\mathrm{perm}(A_{P_0}) + \mathrm{perm}(A_{P_1}) - \mathrm{perm}(A_{P_2})$ gives length of shortest 2-disjoint paths

Applications

Shortest disjoint cycles

- *SDC*(l, k): Given a weighted undirected graph G with k-marked vertices, find shortest l-disjoint cycles through these k vertices.

Applications

Shortest disjoint cycles

- $SDC(l, k)$: Given a weighted undirected graph G with k-marked vertices, find shortest l-disjoint cycles through these k vertices.
- Edge variant: Given an undirected graph G with k-marked edges, find l-disjoint cycles through these k-edges.

Applications

Shortest disjoint cycles

- $SDC(l,k)$: Given a weighted undirected graph G with k-marked vertices, find shortest l-disjoint cycles through these k vertices.
- Edge variant: Given an undirected graph G with k-marked edges, find l-disjoint cycles through these k-edges.
- $SDP(2) =_{\mathsf{L}} SDC(2,2)$

Applications

Shortest disjoint cycles

- $SDC(l,k)$: Given a weighted undirected graph G with k-marked vertices, find shortest l-disjoint cycles through these k vertices.
- Edge variant: Given an undirected graph G with k-marked edges, find l-disjoint cycles through these k-edges.
- $SDP(2) =_{\mathsf{L}} SDC(2,2)$
- We were able to find algorithms for shortest disjont cycle $SDC(1,k)$ shortest disjoint cycle and shortest 2-disjoint cycles $SDC(2,k)$, for all $k \geq 1$

Applications

Shortest disjoint cycles

- $SDC(l, k)$: Given a weighted undirected graph G with k-marked vertices, find shortest l-disjoint cycles through these k vertices.
- Edge variant: Given an undirected graph G with k-marked edges, find l-disjoint cycles through these k-edges.
- $SDP(2) =_{\mathsf{L}} SDC(2, 2)$
- We were able to find algorithms for shortest disjont cycle $SDC(1, k)$ shortest disjoint cycle and shortest 2-disjoint cycles $SDC(2, k)$, for all $k \geq 1$
- Although our algorithm for $SDC(1, k)$ is already reminiscient in the work of [Wahlström] on finding a cycle through k-vertices

Applications

Shortest cycle [Wahlström]

- For each binary sequence $b = (b_1 b_2 \dots b_{k-1})$ consider the pattern P_b defined as: $(s_1, t_1) \in P_b$ and $b_{i-1} = 0 \implies (s_i, t_i) \in P_b$ else $(t_i, s_i) \in P_b$

Applications

Shortest cycle [Wahlström]

- For each binary sequence $b = (b_1 b_2 \dots b_{k-1})$ consider the pattern P_b defined as: $(s_1, t_1) \in P_b$ and $b_{i-1} = 0 \implies (s_i, t_i) \in P_b$ else $(t_i, s_i) \in P_b$
- Exponent of smallest term in $\sum_b \mathrm{perm}(A_{P_b}) \pmod 2$ gives the weight of shortest cycle

Applications

Shortest 2-disjoint cycles [DJ]

- Let P_b be the patterns defined previously

Applications

Shortest 2-disjoint cycles [DJ]

- Let P_b be the patterns defined previously
- For each binary sequence $c = (c_1 c_2 \ldots c_{k-2})$ consider the pattern Q_c defined as: $(s_1, s_2), (t_1, t_2) \in Q_c$ and $c_{i-2} = 0 \implies (s_i, t_i) \in Q_c$ else $(t_i, s_i) \in Q_c$

Applications

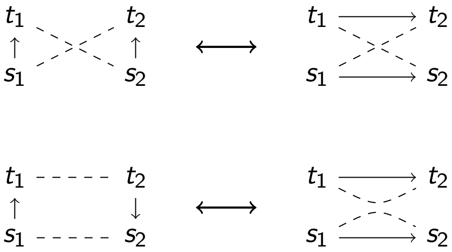
Shortest 2-disjoint cycles [DJ]

- Let P_b be the patterns defined previously
- For each binary sequence $c = (c_1 c_2 \dots c_{k-2})$ consider the pattern Q_c defined as: $(s_1, s_2), (t_1, t_2) \in Q_c$ and $c_{i-2} = 0 \implies (s_i, t_i) \in Q_c$ else $(t_i, s_i) \in Q_c$
- Exponent of smallest term in $\sum_b \mathrm{perm}(A_{P_b}) - \sum_c \mathrm{perm}(A_{Q_c}) \pmod 4$ gives the weight of shortest 2-disjoint cycles separating (s_1, t_1) and (s_2, t_2)

Applications

Shortest 2-disjoint cycles [DJ]

- Let P_b be the patterns defined previously
- For each binary sequence $c = (c_1 c_2 \ldots c_{k-2})$ consider the pattern Q_c defined as: $(s_1, s_2), (t_1, t_2) \in Q_c$ and $c_{i-2} = 0 \implies (s_i, t_i) \in Q_c$ else $(t_i, s_i) \in Q_c$
- Exponent of smallest term in $\sum_b \mathrm{perm}(A_{P_b}) - \sum_c \mathrm{perm}(A_{Q_c}) \pmod 4$ gives the weight of shortest 2-disjoint cycles separating (s_1, t_1) and (s_2, t_2)



Complexity Classes

- Class NC which encaptures the notion of parallel computation, which is defined as follows

Complexity Classes

- Class NC which encaptures the notion of parallel computation, which is defined as follows
- NC^i is the class of decision problems solvable in time $O(\log^i n)$ on a parallel computer with a polynomial number of processors, or the class of decision problems decidable by uniform boolean circuits with a polynomial number of gates of fan-in 2 and depth $O(\log^i n)$

Complexity Classes

- Class NC which encaptures the notion of parallel computation, which is defined as follows
- NC^i is the class of decision problems solvable in time $O(\log^i n)$ on a parallel computer with a polynomial number of processors, or the class of decision problems decidable by uniform boolean circuits with a polynomial number of gates of fan-in 2 and depth $O(\log^i n)$
- $\mathrm{NC} = \bigcup \mathrm{NC}^i$

Complexity Classes

- Class NC which encaptures the notion of parallel computation, which is defined as follows
- NC^i is the class of decision problems solvable in time $O(\log^i n)$ on a parallel computer with a polynomial number of processors, or the class of decision problems decidable by uniform boolean circuits with a polynomial number of gates of fan-in 2 and depth $O(\log^i n)$
- $\mathrm{NC} = \bigcup \mathrm{NC}^i$
- $\oplus\mathrm{L} \subseteq \mathrm{NC}^2$ is the class of decision problems solvable by an NL machine such that:
  - If the answer is 'yes', then the number of accepting paths is odd.
  - If the answer is 'no', then the number of accepting paths is even.

Parallel algorithm for permanent over integers

Intro

- Let $A \in \mathbb{Z}^{n \times n}$

Parallel algorithm for permanent over integers

Intro

- Let $A \in \mathbb{Z}^{n \times n}$
- $\mathrm{perm}(A) = \det(A) \pmod 2$ can be computed in $\oplus$L [MV97]

Parallel algorithm for permanent over integers

Intro

- Let $A \in \mathbb{Z}^{n \times n}$
- $\mathrm{perm}(A) = \det(A) \pmod 2$ can be computed in $\oplus$L [MV97]
- $\mathrm{perm}(A) \pmod 4$?

Parallel algorithm for permanent over integers

Intro

- Let $A \in \mathbb{Z}^{n \times n}$
- $\mathrm{perm}(A) = \det(A) \pmod 2$ can be computed in $\oplus$L [MV97]
- $\mathrm{perm}(A) \pmod 4$?
- Two cases: $\mathrm{perm}(A) \equiv 0 \pmod 2$ or $\mathrm{perm}(A) \not\equiv 0 \pmod 2$

Parallel algorithm for permanent over integers

Intro

- Let $A \in \mathbb{Z}^{n \times n}$
- $\mathrm{perm}(A) = \det(A) \pmod 2$ can be computed in $\oplus$L [MV97]
- $\mathrm{perm}(A) \pmod 4$?
- Two cases: $\mathrm{perm}(A) \equiv 0 \pmod 2$ or $\mathrm{perm}(A) \not\equiv 0 \pmod 2$
- That is: $\det(A)$ is even or odd

Parallel algorithm for permanent over integers

Intro

- Let $A \in \mathbb{Z}^{n \times n}$
- $\mathrm{perm}(A) = \det(A) \pmod 2$ can be computed in $\oplus$L [MV97]
- $\mathrm{perm}(A) \pmod 4$?
- Two cases: $\mathrm{perm}(A) \equiv 0 \pmod 2$ or $\mathrm{perm}(A) \not\equiv 0 \pmod 2$
- That is: $\det(A)$ is even or odd
- Shall reduce odd case to even case by matrix perturbation

Parallel algorithm for permanent over integers

Even case

- Assume $\det(A)$ is even. Then we can find a vector $v \in \mathbb{Z}_2^n$ such that $A^T v = 0 \pmod 2$. Assume WLOG $v_1 = 1$

Parallel algorithm for permanent over integers

Even case

- Assume $\det(A)$ is even. Then we can find a vector $v \in \mathbb{Z}_2^n$ such that $A^T v = 0 \pmod 2$. Assume WLOG $v_1 = 1$
- Let A' be the matrix obtained by replacing the first row with $\sum v_i r_i$ where r_i denotes i^{th} row of A

Parallel algorithm for permanent over integers

Even case

- Assume $\det(A)$ is even. Then we can find a vector $v \in \mathbb{Z}_2^n$ such that $A^T v = 0 \pmod 2$. Assume WLOG $v_1 = 1$
- Let A' be the matrix obtained by replacing the first row with $\sum v_i r_i$ where r_i denotes i^{th} row of A
- Denote by $A[\hat{I}, \hat{J}]$ the matrix obtained from A by removing the I-rows and J-columns

Parallel algorithm for permanent over integers

Even case

- Assume $\det(A)$ is even. Then we can find a vector $v \in \mathbb{Z}_2^n$ such that $A^T v = 0 \pmod 2$. Assume WLOG $v_1 = 1$
- Let A' be the matrix obtained by replacing the first row with $\sum v_i r_i$ where r_i denotes i^{th} row of A
- Denote by $A[\hat{I}, \hat{J}]$ the matrix obtained from A by removing the I-rows and J-columns
- Then

$$\mathrm{perm}(A') = \sum_j \left(\sum_i v_i a_{ij} \right) \mathrm{perm}(A[\hat{1}, \hat{j}])$$

Parallel algorithm for permanent over integers

Even case

- Ha! We have a double summation so we can evaluate this sum in two ways

$$\mathrm{perm}(A') = \sum_i v_i \left(\sum_j a_{ij} \mathrm{perm}(A[\hat{1}, \hat{j}]) \right)$$

Parallel algorithm for permanent over integers

Even case

- Ha! We have a double summation so we can evaluate this sum in two ways

$$\mathrm{perm}(A') = \sum_i v_i \left(\sum_j a_{ij} \mathrm{perm}(A[\hat{1}, \hat{j}]) \right)$$

Parallel algorithm for permanent over integers

Even case

- Ha! We have a double summation so we can evaluate this sum in two ways

$$\mathrm{perm}(A') = \sum_i v_i \left(\sum_j a_{ij} \mathrm{perm}(A[\hat{1}, \hat{j}]) \right)$$

- Let A_i denote the matrix A but with the first row replaced with i^{th} row then

$$\mathrm{perm}(A') = \sum_i v_i \mathrm{perm}(A_i) = \mathrm{perm}(A) + \sum_{i>1} v_i \mathrm{perm}(A_i)$$

Parallel algorithm for permanent over integers

Even case

- Since $A^T v = 0 \pmod{2} \implies \sum_i v_i a_{ij} = 2b_j$ and so

Parallel algorithm for permanent over integers

Even case

- Since $A^T v = 0 \pmod 2 \implies \sum_i v_i a_{ij} = 2b_j$ and so $\mathrm{perm}(A') \pmod 4 = 2\left(\sum_j b_j \mathrm{perm}(A[\hat{1},\hat{j}]) \pmod 2\right)$
- But A_i has two equal rows and we exploit this as follows:

$$\mathrm{perm}(A_i) = \sum_{j,k} a_{ij} a_{ik} \mathrm{perm}(A[\widehat{\{1,i\}},\widehat{\{j,k\}}])$$

Parallel algorithm for permanent over integers

Even case

- Since $A^T v = 0 \pmod 2 \implies \sum_i v_i a_{ij} = 2b_j$ and so $\mathrm{perm}(A') \pmod 4 = 2\left(\sum_j b_j \mathrm{perm}(A[\hat{1},\hat{j}]) \pmod 2\right)$
- But A_i has two equal rows and we exploit this as follows:

$$\mathrm{perm}(A_i) = \sum_{j,k} a_{ij} a_{ik} \mathrm{perm}(A[\widehat{\{1,i\}}, \widehat{\{j,k\}}])$$

- Re-write this as $\mathrm{perm}(A_i) = 2\left(\sum_{j<k} a_{ij} a_{ik} \mathrm{perm}(A[\widehat{\{1,i\}}, \widehat{\{j,k\}}])\right)$

Parallel algorithm for permanent over integers

Even case

$$
\begin{aligned}
&\operatorname{perm}(A) \pmod 4 \\
&\quad = 2\left(\sum_{j=1}^{n} b_j \operatorname{perm}(A[\widehat{\{1\}}, \widehat{\{j\}}]) \pmod 2\right) \\
&\quad - 2\sum_{i=2}^{n} v_i \left(\sum_{\substack{j,k=1 \\ j<k}}^{n} a_{ij}a_{ik} \operatorname{perm}(A[\widehat{\{1,i\}}, \widehat{\{j,k\}}]) \pmod 2\right)
\end{aligned}
$$

Parallel algorithm for permanent over integers

Odd case

- So $\operatorname{perm}(A) \not\equiv 0 \pmod 2$

Parallel algorithm for permanent over integers

Odd case

- So $\mathrm{perm}(A) \not\equiv 0 \pmod 2$
- Expand along i^{th} row

Parallel algorithm for permanent over integers

Odd case

- So $\mathrm{perm}(A) \not\equiv 0 \pmod 2$
- Expand along i^{th} row
- $\mathrm{perm}(A) = \sum_j a_{ij}\mathrm{perm}(A[\hat{i}, \hat{j}]) \not\equiv 0 \pmod 2$

Parallel algorithm for permanent over integers

Odd case

- So $\mathrm{perm}(A) \not\equiv 0 \pmod 2$
- Expand along i^{th} row
- $\mathrm{perm}(A) = \sum_j a_{ij}\mathrm{perm}(A[\hat{i},\hat{j}]) \not\equiv 0 \pmod 2$
- $\exists j$ such that $\mathrm{perm}(A[\hat{i},\hat{j}]) \not\equiv 0 \pmod 2$

Parallel algorithm for permanent over integers

Odd case

- So $\mathrm{perm}(A) \not\equiv 0 \pmod 2$
- Expand along i^{th} row
- $\mathrm{perm}(A) = \sum_j a_{ij}\mathrm{perm}(A[\hat{i},\hat{j}]) \not\equiv 0 \pmod 2$
- $\exists j$ such that $\mathrm{perm}(A[\hat{i},\hat{j}]) \not\equiv 0 \pmod 2$
- Increment a_{ij} by 1 and call the resulting matrix C. So we get

$$\mathrm{perm}(C) = \mathrm{perm}(A) + \mathrm{perm}(A[\hat{i},\hat{j}]) \equiv 0 \pmod 2$$

Parallel algorithm for permanent over integers

Odd case

- So $\mathrm{perm}(A) \not\equiv 0 \pmod 2$
- Expand along i^{th} row
- $\mathrm{perm}(A) = \sum_j a_{ij}\mathrm{perm}(A[\hat{i},\hat{j}]) \not\equiv 0 \pmod 2$
- $\exists j$ such that $\mathrm{perm}(A[\hat{i},\hat{j}]) \not\equiv 0 \pmod 2$
- Increment a_{ij} by 1 and call the resulting matrix C. So we get

$$\mathrm{perm}(C) = \mathrm{perm}(A) + \mathrm{perm}(A[\hat{i},\hat{j}]) \equiv 0 \pmod 2$$

- This gives us a sequential algorithm for computing $\mathrm{perm}(A) \pmod 4$

Parallel algorithm for permanent over integers

Odd case

- $\det(A)$ is odd that means A is non-singular.

Parallel algorithm for permanent over integers

Odd case

- $\det(A)$ is odd that means A is non-singular.
- Every non-singular matrix can be written as $A = PLU$ where L, U are lower,upper triangular matrices, P is a permutation matrix.

Parallel algorithm for permanent over integers
Odd case

- $\det(A)$ is odd that means A is non-singular.
- Every non-singular matrix can be written as $A = PLU$ where L, U are lower,upper triangular matrices, P is a permutation matrix.

Theorem (Okunev, Johnson)

Let A be a non-singular matrix over a field $\mathbb{F}$ then all leading principal minors of A are non-zero iff A admits a LU decomposition

Parallel algorithm for permanent over integers

Odd case

- $\det(A)$ is odd that means A is non-singular.
- Every non-singular matrix can be written as $A = PLU$ where L, U are lower,upper triangular matrices, P is a permutation matrix.

Theorem (Okunev, Johnson)

Let A be a non-singular matrix over a field $\mathbb{F}$ then all leading principal minors of A are non-zero iff A admits a LU decomposition

- Thus $QA = LU$ $(Q = P^{-1})$

Parallel algorithm for permanent over integers

Odd case

- $\det(A)$ is odd that means A is non-singular.
- Every non-singular matrix can be written as $A = PLU$ where L, U are lower,upper triangular matrices, P is a permutation matrix.

Theorem (Okunev, Johnson)

Let A be a non-singular matrix over a field $\mathbb{F}$ then all leading principal minors of A are non-zero iff A admits a LU decomposition

- Thus $QA = LU$ ($Q = P^{-1}$)
- Q is also a permutation matrix and so QA is just the matrix A with it's rows permuted. Hence $\mathrm{perm}(A) = \mathrm{perm}(QA)$

Parallel algorithm for permanent over integers

Eberly

- How to find this permutation matrix?

Parallel algorithm for permanent over integers

Eberly

- How to find this permutation matrix?
- Let A_i be the matrix formed by taking by only taking the first i columns of A

Parallel algorithm for permanent over integers

Eberly

- How to find this permutation matrix?
- Let A_i be the matrix formed by taking by only taking the first i columns of A
- We construct S_i, be the lexicographically first set of indices which correspond to maximal linearly independent rows of A_i

Parallel algorithm for permanent over integers

Eberly

- How to find this permutation matrix?
- Let A_i be the matrix formed by taking by only taking the first i columns of A
- We construct S_i, be the lexicographically first set of indices which correspond to maximal linearly independent rows of A_i
- $rank(A_i) = i \implies |S_i| = i$ and $S_i \subset S_{i+1}$

Parallel algorithm for permanent over integers

Eberly

- How to find this permutation matrix?
- Let A_i be the matrix formed by taking by only taking the first i columns of A
- We construct S_i, be the lexicographically first set of indices which correspond to maximal linearly independent rows of A_i
- $rank(A_i) = i \implies |S_i| = i$ and $S_i \subset S_{i+1}$
- Permute rows of A such that $S_i = [1 \ldots i]$. How?

Parallel algorithm for permanent over integers

Eberly

- How to find this permutation matrix?
- Let A_i be the matrix formed by taking by only taking the first i columns of A
- We construct S_i, be the lexicographically first set of indices which correspond to maximal linearly independent rows of A_i
- $rank(A_i) = i \implies |S_i| = i$ and $S_i \subset S_{i+1}$
- Permute rows of A such that $S_i = [1 \ldots i]$. How?
- Let $S_1 = s_1$ and s_i the unique element in $S_i \setminus S_{i-1}$ for $i > 1$.

Parallel algorithm for permanent over integers

Eberly

- How to find this permutation matrix?
- Let A_i be the matrix formed by taking by only taking the first i columns of A
- We construct S_i, be the lexicographically first set of indices which correspond to maximal linearly independent rows of A_i
- $rank(A_i) = i \implies |S_i| = i$ and $S_i \subset S_{i+1}$
- Permute rows of A such that $S_i = [1 \dots i]$. How?
- Let $S_1 = s_1$ and s_i the unique element in $S_i \setminus S_{i-1}$ for $i > 1$.
- Required permutation is $Q = (n, s_n) \dots (2, s_2)(1, s_1)$ such that all leading principal minors of QA are non-zero

Polynomial permanent

Intro

- Let A be a matrix of integer polynomials

Polynomial permanent

Intro

- Let A be a matrix of integer polynomials
- Björklund and Husfeldt gave a poly-time algorithm to find $\mathrm{perm}(A) \pmod{2^k}$.

Polynomial permanent

Intro

- Let A be a matrix of integer polynomials
- Björklund and Husfeldt gave a poly-time algorithm to find $\mathrm{perm}(A) \pmod{2^k}$.
- We present a parallel algorithm to compute permanent of a matrix over integer polynomials (modulo 2^k) in $\oplus$L

Polynomial permanent

Intro

- Let A be a matrix of integer polynomials
- Björklund and Husfeldt gave a poly-time algorithm to find $\mathrm{perm}(A) \pmod{2^k}$.
- We present a parallel algorithm to compute permanent of a matrix over integer polynomials (modulo 2^k) in $\oplus\mathsf{L}$
- Can we adopt the above [BKR] technique to get a parallel algorithm?

Polynomial permanent

Intro

- Let A be a matrix of integer polynomials
- Björklund and Husfeldt gave a poly-time algorithm to find $\mathrm{perm}(A) \pmod{2^k}$.
- We present a parallel algorithm to compute permanent of a matrix over integer polynomials (modulo 2^k) in $\oplus$L
- Can we adopt the above [BKR] technique to get a parallel algorithm?
- Replacing $\mathbb{Z}$ with $\mathbb{Z}[x]$ fails because $\mathbb{Z}_2$ is a field whereas $\mathbb{Z}_2[x]$ is not!

Polynomial permanent

Intro

- Let A be a matrix of integer polynomials
- Björklund and Husfeldt gave a poly-time algorithm to find $\mathrm{perm}(A) \pmod{2^k}$.
- We present a parallel algorithm to compute permanent of a matrix over integer polynomials (modulo 2^k) in $\oplus$L
- Can we adopt the above [BKR] technique to get a parallel algorithm?
- Replacing $\mathbb{Z}$ with $\mathbb{Z}[x]$ fails because $\mathbb{Z}_2$ is a field whereas $\mathbb{Z}_2[x]$ is not!
- How do we get a field? Quotient $\mathbb{Z}_2[x]$ by an irreducible polynomial $p(x)$

Polynomial permanent

Intro

- Let A be a matrix of integer polynomials
- Björklund and Husfeldt gave a poly-time algorithm to find $\mathrm{perm}(A) \pmod{2^k}$.
- We present a parallel algorithm to compute permanent of a matrix over integer polynomials (modulo 2^k) in $\oplus$L
- Can we adopt the above [BKR] technique to get a parallel algorithm?
- Replacing $\mathbb{Z}$ with $\mathbb{Z}[x]$ fails because $\mathbb{Z}_2$ is a field whereas $\mathbb{Z}_2[x]$ is not!
- How do we get a field? Quotient $\mathbb{Z}_2[x]$ by an irreducible polynomial $p(x)$
- Need a notion of modulo 4 arithmetic extending this field structure

Polynomial permanent

Motivation

- Let $\mathbb{F}$ be a finite field of characteristic 2. Corresponds to modulo 2 arithmetic.

Polynomial permanent

Motivation

- Let $\mathbb{F}$ be a finite field of characteristic 2. Corresponds to modulo 2 arithmetic.
- We recognize the ring $\mathfrak{R} = \mathbb{Z}[x]/(p(x))$

Polynomial permanent

Motivation

- Let $\mathbb{F}$ be a finite field of characteristic 2. Corresponds to modulo 2 arithmetic.
- We recognize the ring $\mathfrak{R} = \mathbb{Z}[x]/(p(x))$
- Modulo 4 corresponds to $\mathfrak{R}/(4) = \mathbb{Z}[x]/(4, p(x))$

Polynomial permanent

Motivation

- Let $\mathbb{F}$ be a finite field of characteristic 2. Corresponds to modulo 2 arithmetic.
- We recognize the ring $\mathfrak{R} = \mathbb{Z}[x]/(p(x))$
- Modulo 4 corresponds to $\mathfrak{R}/(4) = \mathbb{Z}[x]/(4, p(x))$
- Choose $p(x)$ such that its degree is larger than the degree of the permanent polynomial

Polynomial permanent

Motivation

- Let $\mathbb{F}$ be a finite field of characteristic 2. Corresponds to modulo 2 arithmetic.
- We recognize the ring $\mathfrak{R} = \mathbb{Z}[x]/(p(x))$
- Modulo 4 corresponds to $\mathfrak{R}/(4) = \mathbb{Z}[x]/(4, p(x))$
- Choose $p(x)$ such that its degree is larger than the degree of the permanent polynomial
- $x^{2.3^l} + x^{3^l} + 1$ is irreducible over $\mathbb{Z}_2$ for all $l \geq 0$ [JH van Lint] [HV]

Polynomial permanent

Motivation

- Let $\mathbb{F}$ be a finite field of characteristic 2. Corresponds to modulo 2 arithmetic.
- We recognize the ring $\mathfrak{R} = \mathbb{Z}[x]/(p(x))$
- Modulo 4 corresponds to $\mathfrak{R}/(4) = \mathbb{Z}[x]/(4, p(x))$
- Choose $p(x)$ such that its degree is larger than the degree of the permanent polynomial
- $x^{2.3^l} + x^{3^l} + 1$ is irreducible over $\mathbb{Z}_2$ for all $l \geq 0$ [JH van Lint] [HV]
- Can we do better?

Polynomial permanent

Degree reduction via interpolation

- Let $\mathbb{F}$ be a characteristic 2 field of order q

Polynomial permanent

Degree reduction via interpolation

- Let $\mathbb{F}$ be a characteristic 2 field of order q
- $\sum_{a \in \mathbb{F}^*} a^m = \begin{cases} 0 \text{ if } q-1 \nmid m \\ 1 \text{ otherwise} \end{cases}$

Polynomial permanent

Degree reduction via interpolation

- Let $\mathbb{F}$ be a characteristic 2 field of order q
- $\sum_{a \in \mathbb{F}^*} a^m = \begin{cases} 0 \text{ if } q-1 \nmid m \\ 1 \text{ otherwise} \end{cases}$
- Let $f(x) = \sum_{i=0}^{d} c_i x^i$ be an integer polynomial such that $q > d + 2$ then $\sum_{a \in \mathbb{F}^*} a^{q-1-t} f(a) = c_t \pmod 2$

Polynomial permanent

Degree reduction via interpolation

- Let $\mathbb{F}$ be a characteristic 2 field of order q
- $\sum_{a\in\mathbb{F}^*} a^m = \begin{cases} 0 \text{ if } q-1 \nmid m \\ 1 \text{ otherwise} \end{cases}$
- Let $f(x) = \sum_{i=0}^{d} c_i x^i$ be an integer polynomial such that $q > d+2$ then $\sum_{a\in\mathbb{F}^*} a^{q-1-t} f(a) = c_t \pmod 2$
- How do we extract coefficients mod 4?

Polynomial permanent

Degree reduction via interpolation

- We observe that if we instead do the computations over $\mathfrak{R}$ then $\sum_{a\in\mathbb{F}^*} a^m = \begin{cases} 2\alpha_m \text{ if } q-1 \nmid m \\ 2\beta_m + 1 \text{ otherwise} \end{cases}$ where $\alpha_m, \beta_m \in \mathfrak{R}$

Polynomial permanent

Degree reduction via interpolation

- We observe that if we instead do the computations over $\mathfrak{R}$ then $\sum_{a\in\mathbb{F}^*} a^m = \begin{cases} 2\alpha_m \text{ if } q-1 \nmid m \\ 2\beta_m + 1 \text{ otherwise} \end{cases}$ where $\alpha_m, \beta_m \in \mathfrak{R}$
- $\sum_{a_1,a_2\in\mathbb{F}^*}(a_1a_2)^m = \left(\sum_{a\in\mathbb{F}^*} a^m\right)^2 =$
$\begin{cases} 4\alpha_m^2 \text{ if } q-1 \nmid m \\ 4\beta_m^2 + 4\beta_m + 1 \text{ otherwise} \end{cases} = \begin{cases} 0 \pmod 4 \text{ if } q-1 \nmid m \\ 1 \pmod 4 \text{ otherwise} \end{cases}$

Polynomial permanent

Degree reduction via interpolation

- We observe that if we instead do the computations over $\mathfrak{R}$ then $\sum_{a\in\mathbb{F}^*} a^m = \begin{cases} 2\alpha_m \text{ if } q-1 \nmid m \\ 2\beta_m + 1 \text{ otherwise} \end{cases}$ where $\alpha_m, \beta_m \in \mathfrak{R}$
- $\sum_{a_1,a_2\in\mathbb{F}^*}(a_1a_2)^m = \left(\sum_{a\in\mathbb{F}^*} a^m\right)^2 =$ $\begin{cases} 4\alpha_m^2 \text{ if } q-1 \nmid m \\ 4\beta_m^2 + 4\beta_m + 1 \text{ otherwise} \end{cases} = \begin{cases} 0 \pmod 4 \text{ if } q-1 \nmid m \\ 1 \pmod 4 \text{ otherwise} \end{cases}$

Theorem

$$\sum_{a_1,\ldots,a_{2^{k-1}}\in\mathbb{F}^*}(a_1\cdots a_{2^{k-1}})^m = \begin{cases} 0 \pmod{2^k} \textit{ if } q-1 \nmid m \\ 1 \pmod{2^k} \textit{ otherwise} \end{cases}$$

Polynomial permanent

Degree reduction via interpolation

- Let $A(x)$ be a matrix of integer polynomials (in x).

Polynomial permanent

Degree reduction via interpolation

- Let $A(x)$ be a matrix of integer polynomials (in x).
- Compute over $\mathbb{F}$:

$$\sum_{a \in \mathbb{F}^*} a^{q-1-t} \mathrm{perm}(A(a)) = c_t \pmod{2}$$

Polynomial permanent

Degree reduction via interpolation

- Let $A(x)$ be a matrix of integer polynomials (in x).
- Compute over $\mathbb{F}$:

$$\sum_{a \in \mathbb{F}^*} a^{q-1-t} \mathrm{perm}(A(a)) = c_t \pmod 2$$

- Compute over $\mathfrak{R} \pmod{2^k}$:

$$\sum_{a_1, a_2, \ldots \in \mathbb{F}^*} (a_1 a_2 \ldots)^{q-1-t} \mathrm{perm}(A(a_1 a_2 \ldots)) = c_t \pmod{2^k}$$

Polynomial permanent

Small field

- Only need to choose field such that it's order $q > N + 2$ where N is the degree of permanent polynomial

Polynomial permanent

Small field

- Only need to choose field such that it's order $q > N + 2$ where N is the degree of permanent polynomial
- Polynomial size field!

Polynomial permanent

Small field

- Only need to choose field such that it's order $q > N + 2$ where N is the degree of permanent polynomial
- Polynomial size field!
- But how to find this field?

Polynomial permanent

Small field

- Only need to choose field such that it's order $q > N + 2$ where N is the degree of permanent polynomial
- Polynomial size field!
- But how to find this field?
- $x^{2.3^l} + x^{3^l} + 1$ is irreducible over $\mathbb{Z}_2$ for all $l \geq 0$ [JH van Lint] [HV]

Polynomial permanent

Small field

- Only need to choose field such that it's order $q > N + 2$ where N is the degree of permanent polynomial
- Polynomial size field!
- But how to find this field?
- $x^{2.3^l} + x^{3^l} + 1$ is irreducible over $\mathbb{Z}_2$ for all $l \geq 0$ [JH van Lint] [HV]
- Choose $l = \lceil \frac{\log \log N}{\log 3} \rceil$ such that $2^{2.3^l} > N + 2$ where $N = poly(n)$ (and $n =$size of matrix)

Example

- Let $A = \begin{pmatrix} 1 & x+1 & x+2 \\ x & x^2 & x^2+x \\ x^2 & 3 & x^2+3 \end{pmatrix}$, $p(x) = x^6 + x^3 + 1$ be the irreducible polynomial and we want to evaluate $\mathrm{perm}(A) \pmod 4$ over the ring $\mathfrak{R} = \mathbb{Z}[x]/(p(x))$.

Example

- Let $A = \begin{pmatrix} 1 & x+1 & x+2 \\ x & x^2 & x^2+x \\ x^2 & 3 & x^2+3 \end{pmatrix}$, $p(x) = x^6 + x^3 + 1$ be the irreducible polynomial and we want to evaluate $\mathrm{perm}(A) \pmod 4$ over the ring $\mathfrak{R} = \mathbb{Z}[x]/(p(x))$.
- Direct computation gives us $\mathrm{perm}(A) = 2x^5 + 6x^4 + 2x^3 + 12x^2 + 12x$. Now we demonstrate the steps taken by our algorithm.

Example

- **Step 1**: We start by evaluating $\mathrm{perm}(A) \pmod 2$. We directly notice here that $\det(A) = 0 \implies \mathrm{perm}(A) \equiv 0 \pmod 2$

Example

- **Step 1**: We start by evaluating $\mathrm{perm}(A) \pmod 2$. We directly notice here that $\det(A) = 0 \implies \mathrm{perm}(A) \equiv 0 \pmod 2$
- **Step 2**: We solve the equation $A^T v = 0$ over $\mathbb{F}$ by our method as follows: $\begin{pmatrix} 1 & x & x^2 \\ x+1 & x^2 & 1 \\ x & x^2+x & x^2+1 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = 0$

Example

- **Step 1**: We start by evaluating $\mathrm{perm}(A) \pmod 2$. We directly notice here that $\det(A) = 0 \implies \mathrm{perm}(A) \equiv 0 \pmod 2$
- **Step 2**: We solve the equation $A^T v = 0$ over $\mathbb{F}$ by our method as follows: $\begin{pmatrix} 1 & x & x^2 \\ x+1 & x^2 & 1 \\ x & x^2+x & x^2+1 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = 0$
- $\begin{pmatrix} x^3+1 \\ x^5+x \\ 1 \end{pmatrix}$ is a solution to the above equations

Example

- **Step 3**: For each $j = 1, 2, 3$, we find b_j such that $\sum_i v_i a_{ij} = 2b_j \pmod 4$

$$
\begin{aligned}
j = 1: &\quad (x^3+1) + x(x^5+x) + x^2 = 2x^2 \\
j = 2: &\quad (x+1)(x^3+1) + x^2(x^5+x) + 3 = 2x^3 \\
j = 3: &\quad (x+2)(x^3+1) + (x^2+x)(x^5+x) + x^2 + 3 = 2x^3 + 2x^2
\end{aligned}
$$

Example

- **Step 4**: We have the formula

$$\mathrm{perm}(A) \pmod 4 = 2\left(\sum_{j=1}^{3} b_j \mathrm{perm}(A[\widehat{\{3\}}, \widehat{\{j\}}]) \pmod 2\right) - 2\sum_{i=1}^{2} v_i \left(\sum_{\substack{j,k=1 \\ j<k}}^{3} t_{jk} \pmod 2\right)$$

where

$$t_{jk} = a_{ij} a_{ik} \mathrm{perm}(A[\widehat{\{3, i\}}, \widehat{\{j, k\}}])$$

Example

- **Step 4.1**:

$$\mathrm{perm}(A[\widehat{\{3\}}, \widehat{\{1\}}]) = \mathrm{perm}\begin{pmatrix} x+1 & x+2 \\ x^2 & x^2+x \end{pmatrix} = x \pmod 2$$

$$\mathrm{perm}(A[\widehat{\{3\}}, \widehat{\{2\}}]) = \mathrm{perm}\begin{pmatrix} 1 & x+2 \\ x & x^2+x \end{pmatrix} = x \pmod 2$$

$$\mathrm{perm}(A[\widehat{\{3\}}, \widehat{\{3\}}]) = \mathrm{perm}\begin{pmatrix} 1 & x+1 \\ x & x^2 \end{pmatrix} = x \pmod 2$$

$$\implies \sum_{j=1}^{3} b_j \mathrm{perm}(A[\widehat{\{3\}}, \widehat{\{j\}}]) = 0 \pmod 2$$

Example

- **Step 4.2**:

$$\sum_{\substack{j,k=1\\ j<k}}^{3} a_{1j}a_{1k}\mathrm{perm}(A[\widehat{\{1,3\}},\widehat{\{j,k\}}]) = x^3+x^2+x \pmod 2$$

$$\sum_{\substack{j,k=1\\ j<k}}^{3} a_{2j}a_{2k}\mathrm{perm}(A[\widehat{\{2,3\}},\widehat{\{j,k\}}]) = x^4+x^3+x^2 \pmod 2$$

$$\sum_{i=1}^{2} v_i \left(\sum_{\substack{j,k=1\\ j<k}}^{3} a_{ij}a_{ik}\mathrm{perm}(A[\widehat{\{3,i\}},\widehat{\{j,k\}}]) \pmod 2 \right) = x^5+x^4+x^3 \pmod 4$$

Example

- **Step 4.2**:

$$\sum_{\substack{j,k=1\\ j<k}}^{3} a_{1j}a_{1k}\mathrm{perm}(A[\widehat{\{1,3\}},\widehat{\{j,k\}}]) = x^3 + x^2 + x \pmod 2$$

$$\sum_{\substack{j,k=1\\ j<k}}^{3} a_{2j}a_{2k}\mathrm{perm}(A[\widehat{\{2,3\}},\widehat{\{j,k\}}]) = x^4 + x^3 + x^2 \pmod 2$$

$$\sum_{i=1}^{2} v_i \left(\sum_{\substack{j,k=1\\ j<k}}^{3} a_{ij}a_{ik}\mathrm{perm}(A[\widehat{\{3,i\}},\widehat{\{j,k\}}]) \pmod 2 \right) = x^5 + x^4 + x^3 \pmod 4$$

- Therefore, $\mathrm{perm}(A) \pmod 4 = 2x^5 + 2x^4 + 2x^3$ which matches with our direct computation.

Further work

- Can we find shortest 3-disjoint paths using this method?

Further work

- Can we find shortest 3-disjoint paths using this method?
- $SDC(l, k)$ for other values of $l \geq 3$?

Further work

- Can we find shortest 3-disjoint paths using this method?
- $SDC(l, k)$ for other values of $l \geq 3$?
- Permanent over rings with characteristic 2^k, $k \geq 2$?

Thank you!

Questions?