Arithmetic Circuit Complexity of $S^*_{n,k}$ and Multilinear Monomial Counting

Joint work with V. Arvind[1] Rajit Datta[2]
Partha Mukhopadhyay[2]

[1]Institute of Mathematical Sciences(HBNI), India
[2]Chennai Mathematical Institute, India

Workshop on Arithmetic Complexity Theory, 2019

Outline

1. Multilinear Monomial Detection and Counting.

Outline

1. Multilinear Monomial Detection and Counting.
2. Our Approach.

Outline

1. Multilinear Monomial Detection and Counting.
2. Our Approach.
3. ABP construction of $S^*_{n,k}$.

Outline

1. Multilinear Monomial Detection and Counting.
2. Our Approach.
3. ABP construction of $S^*_{n,k}$.
4. Beating the Brute Force.

Multilinear Monomial Detection and Counting

Koutis and Williams [**Kou08**, **Wi09**, **KW16**] introduced and studied two algorithmic problems on arithmetic circuits.

Multilinear Monomial Detection and Counting

Koutis and Williams [**Kou08**, **Wi09**, **KW16**] introduced and studied two algorithmic problems on arithmetic circuits.

Definition (k-MMD)

Given as input an arithmetic circuit C computing a polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$, decide whether f has a non-zero degree-k multilinear monomial.

Multilinear Monomial Detection and Counting

Koutis and Williams [**Kou08**, **Wi09**, **KW16**] introduced and studied two algorithmic problems on arithmetic circuits.

Definition (k-MMD)

Given as input an arithmetic circuit C computing a polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$, decide whether f has a non-zero degree-k multilinear monomial.

Definition ((k,n)-MLC)

Given as input an arithmetic circuit C computing a polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$, compute the sum of the coefficients of all degree-k multilinear monomials in the polynomial f.

Multilinear Monomial Detection and Counting

- k-path: Given a graph G, does there exist a path of length k?

Multilinear Monomial Detection and Counting

- k-path: Given a graph G, does there exist a path of length k?
- k-Tree: Given a graph G and tree T of size k, does there exist a copy of T in G?

Multilinear Monomial Detection and Counting

- k-path: Given a graph G, does there exist a path of length k?
- k-Tree: Given a graph G and tree T of size k, does there exist a copy of T in G?
- t-Dominating Set: Given graph G, does there exist a set of size k that dominates at least t nodes in G?

Multilinear Monomial Detection and Counting

- k-path: Given a graph G, does there exist a path of length k?
- k-Tree: Given a graph G and tree T of size k, does there exist a copy of T in G?
- t-Dominating Set: Given graph G, does there exist a set of size k that dominates at least t nodes in G?
- m-Dimensional k-Matching: Given mutually disjoint sets U_i, $i \in [m]$ and a collection $\mathcal{C}$ of m-tuples from $U_1 \times \cdots \times U_m$, does there exist a sub-collection of k mutually disjoint m-tuples in $\mathcal{C}$?

Multilinear Monomial Detection and Counting

- k-path: Given a graph G, does there exist a path of length k?
- k-Tree: Given a graph G and tree T of size k, does there exist a copy of T in G?
- t-Dominating Set: Given graph G, does there exist a set of size k that dominates at least t nodes in G?
- m-Dimensional k-Matching: Given mutually disjoint sets U_i, $i \in [m]$ and a collection $\mathcal{C}$ of m-tuples from $U_1 \times \cdots \times U_m$, does there exist a sub-collection of k mutually disjoint m-tuples in $\mathcal{C}$?

Koutis and Williams [**KW16**] obtain a randomized $O^*(2^k)$ algorithm for k-MMD and reduce all these combinatorial problems to k-MMD.

Our Result

Our Result

However, for (k,n)-MLC, nothing better than $\binom{n}{k}$ was known. Alon and Gutner [**AG10**] have shown that, using color-coding technique, one can not obtain better than $O^*(n^{k/2})$.

Our Result

However, for (k,n)-MLC, nothing better than $\binom{n}{k}$ was known. Alon and Gutner [**AG10**] have shown that, using color-coding technique, one can not obtain better than $O^*(n^{k/2})$.

Challenge. To solve (k,n)-MLC beating the brute force.

Our Result

However, for (k,n)-MLC, nothing better than $\binom{n}{k}$ was known. Alon and Gutner [**AG10**] have shown that, using color-coding technique, one can not obtain better than $O^*(n^{k/2})$.

Challenge. To solve (k,n)-MLC beating the brute force.

It will improve the counting version of all these problems in one shot!

Our Result

However, for (k,n)-MLC, nothing better than $\binom{n}{k}$ was known.
Alon and Gutner [**AG10**] have shown that, using color-coding technique, one can not obtain better than $O^*(n^{k/2})$.
Challenge. To solve (k,n)-MLC beating the brute force.
It will improve the counting version of all these problems in one shot!

Theorem

(k,n)-MLC *can be solved in deterministic* $O^*(n^{k/2+c\log k})$ *time for some constant* c.

Our Approach

Our Approach

Definition (Elementary Symmetric Polynomial)

Elementary symmetric polynomial over $\{x_1, \ldots, x_n\}$ of degree k, denoted by $S_{n,k}$, is defined as,

$$S_{n,k}(x_1, \ldots, x_n) = \sum_{\{i_1,\ldots,i_k\} \subseteq [n]} \prod_{j=1}^{k} x_{i_j}.$$

Our Approach

Definition (Elementary Symmetric Polynomial)

Elementary symmetric polynomial over $\{x_1, \ldots, x_n\}$ of degree k, denoted by $S_{n,k}$, is defined as,

$$S_{n,k}(x_1, \ldots, x_n) = \sum_{\{i_1,\ldots,i_k\}\subseteq[n]} \prod_{j=1}^{k} x_{i_j}.$$

Definition (Hadamard Product)

Hadamard product of two polynomials $f, g \in \mathbb{F}[x_1, \ldots, x_n]$ of degree at most d is defined as,

$$f \circ g(x_1, \ldots, x_n) = \sum_m [m]f \cdot [m]g \cdot m.$$

Our Approach

Our Approach

- Note that, taking Hadamard product of a polynomial with $S_{n,k}$ filtrates the degree-k multilinear part of that polynomial.

Our Approach

- Note that, taking Hadamard product of a polynomial with $S_{n,k}$ filtrates the degree-k multilinear part of that polynomial.
- Given a circuit C, (k,n)-$\mathrm{MLC}(C)$ reduces to evaluating $(C \circ S_{n,k})(\vec{1})$.

Our Approach

- Note that, taking Hadamard product of a polynomial with $S_{n,k}$ filtrates the degree-k multilinear part of that polynomial.
- Given a circuit C, (k,n)-$\mathrm{MLC}(C)$ reduces to evaluating $(C \circ S_{n,k})(\vec{1})$.
- However, it is 'hard' to compute even when C is given by a 'small' circuit. For example, given graph $G = (V, E)$, evaluating

$$\left(\sum_{(i,j)\in E} x_i x_j \right)^k \circ S_{n,2k}$$

at $\vec{1}$, yields the number of k-matchings in G.

Detour to Non-commutative Computation

Detour to Non-commutative Computation

Definition (Non-commutative Polynomial Ring)

- Let X be the set of n indeterminates $\{x_1, x_2, \ldots, x_n\}$ and $\mathbb{F}$ be any arbitrary field.
- The non-commutative polynomial ring $\mathbb{F}\langle X\rangle$ is identified with the monoid algebra over $\mathbb{F}$ of the free monoid X^* generated by X.
- So for each ring element $p \in \mathbb{F}\langle X\rangle$, we may write, $p = \sum_{w \in X^*} c_w w$ where each $c_w \in \mathbb{F}$.

Detour to Non-commutative Computation

Definition (Algebraic Branching Program)

- Directed layered acyclic graph.
- One in-degree-0 vertex called *source*, and one out-degree-0 vertex called *sink*.
- Edges only go between consecutive layers i and $i+1$.
- Each edge is labeled by a linear form over variables X.
- The polynomial computed by the ABP is the sum over all source-to-sink directed paths of the product of linear forms that label the edges of the path.

Detour to Non-commutative Computation

Detour to Non-commutative Computation

- Arvind et al.[**AJS09**] show that non-commutative Hadamard product is 'easy' to compute when one of the polynomials is given by an ABP.

Detour to Non-commutative Computation

- Arvind et al.[**AJS09**] show that non-commutative Hadamard product is 'easy' to compute when one of the polynomials is given by an ABP.
- **Idea.** Can we reduce the computation of commutative Hadamard product to non-commutative computations?

Detour to Non-commutative Computation

- Arvind et al.[**AJS09**] show that non-commutative Hadamard product is 'easy' to compute when one of the polynomials is given by an ABP.
- **Idea.** Can we reduce the computation of commutative Hadamard product to non-commutative computations?
- Let us denote $X = \{x_1, \ldots, x_n\}$ to be a set of n commuting variables and $Y = \{y_1, \ldots, y_n\}$ to be a set of n non-commuting variables.

Transformation Theorem

Transformation Theorem

- Given a commutative circuit C computing a polynomial in $\mathbb{F}[X]$, the *noncommutative version* of C, C^{nc} as the noncommutative circuit obtained from C by fixing an ordering of the inputs to each product gate in C and replacing x_i by $y_i, 1 \leq i \leq n$.

Transformation Theorem

- Given a commutative circuit C computing a polynomial in $\mathbb{F}[X]$, the *noncommutative version* of C, C^{nc} as the noncommutative circuit obtained from C by fixing an ordering of the inputs to each product gate in C and replacing x_i by $y_i, 1 \leq i \leq n$.
- For a homogeneous degree-k commutative polynomial $f \in \mathbb{F}[X]$ given by circuit C, the *symmetrized polynomial* of f, f^*,is degree-k homogeneous polynomial

$$f^* = \sum_{\sigma \in S_k} \hat{f}^{\sigma},$$

where $\hat{f} \in \mathbb{F}\langle Y \rangle$ computed by C^{nc}.

Transformation Theorem

Transformation Theorem

- For each monomial $m \in X_k$ and each word $m' \in Y^k$ such that $m' \to m$, we have: $[m']f^* = [m]f$.

Transformation Theorem

- For each monomial $m \in X_k$ and each word $m' \in Y^k$ such that $m' \to m$, we have: $[m']f^* = [m]f$.
- Notice that $[m]f = \sum_{\hat{m} \to m} [\hat{m}]\hat{f}$.

$$[m']f^* = \sum_{\hat{m}^\sigma} [\hat{m}^\sigma]\hat{f} = \sum_{\hat{m} \to m} [\hat{m}]\hat{f} = [m]f.$$

Transformation Theorem

- Let C_1, C_2 be two circuits for a homogeneous degree-k polynomial $f, g \in \mathbb{F}[X]$. Given any $\vec{a} \in \mathbb{F}^n$,

$$
\begin{aligned}
(f^* \circ C_2^{nc})(\vec{a}) &= \sum_{m'} [m']f^* \cdot [m']C_2^{nc} \cdot m'(\vec{a}) \\
&= \sum_{m} \sum_{m' \to m} [m']f^* \cdot [m']C_2^{nc} \cdot m'(\vec{a}) \\
&= \sum_{m} [m]f \sum_{m' \to m} [m']C_2^{nc} \cdot m'(\vec{a}) \\
&= \sum_{m} [m]f \cdot m(\vec{a}) \sum_{m' \to m} [m']C_2^{nc} \\
&= (C_1 \circ C_2)(\vec{a}).
\end{aligned}
$$

(k, n)-MLC and Arithmetic Complexity of $S^*_{n,k}$.

(k,n)-MLC and Arithmetic Complexity of $S^*_{n,k}$.

- Nisan [**Ni91**] defined

$$S^*_{n,k} = \sum_{\{i_1,\ldots,i_k\}\subseteq[n]} \sum_{\sigma\in S_k} \prod_{j=1}^{k} x_{i_{\sigma(j)}}.$$

(k,n)-MLC and Arithmetic Complexity of $S^*_{n,k}$.

- Nisan [**Ni91**] defined

$$S^*_{n,k} = \sum_{\{i_1,\ldots,i_k\}\subseteq[n]} \sum_{\sigma\in S_k} \prod_{j=1}^{k} x_{i_{\sigma(j)}}.$$

- Recall that, given a circuit C, (k,n)-MLC(C) reduces to evaluating $(C \circ S_{n,k})(\vec{1})$.

(k,n)-MLC and Arithmetic Complexity of $S^*_{n,k}$.

- Nisan [**Ni91**] defined

$$S^*_{n,k} = \sum_{\{i_1,\ldots,i_k\}\subseteq[n]} \sum_{\sigma\in S_k} \prod_{j=1}^{k} x_{i_{\sigma(j)}}.$$

- Recall that, given a circuit C, (k,n)-MLC(C) reduces to evaluating $(C \circ S_{n,k})(\vec{1})$.
- Given a circuit C, (k,n)-MLC(C) reduces to evaluating $(C^{nc} \circ S^*_{n,k})(\vec{1})$.

(k,n)-MLC and Arithmetic Complexity of $S^*_{n,k}$.

- Nisan [**Ni91**] defined

$$S^*_{n,k} = \sum_{\{i_1,\ldots,i_k\}\subseteq[n]} \sum_{\sigma\in S_k} \prod_{j=1}^{k} x_{i_{\sigma(j)}}.$$

- Recall that, given a circuit C, (k,n)-MLC(C) reduces to evaluating $(C \circ S_{n,k})(\vec{1})$.
- Given a circuit C, (k,n)-MLC(C) reduces to evaluating $(C^{nc} \circ S^*_{n,k})(\vec{1})$.
- Using the result of Arvind et al.[**AJS09**], it now reduces to 'explicit' ABP construction of $S^*_{n,k}$.

ABP construction for $S^*_{n,k}$

ABP construction for $S^*_{n,k}$

A key ingredient to our algorithm for (k,n)-MLC is new explicit circuit upper bound for $S^*_{n,k}$.

ABP construction for $S^*_{n,k}$

A key ingredient to our algorithm for (k,n)-MLC is new explicit circuit upper bound for $S^*_{n,k}$.

Definition (Explicit Circuit Upper Bound)

A family $\{f_n\}_{n>0}$ of degree-k polynomials has $q(n,k)$-*explicit upper bounds* if there is an $O^*(q(n,k))$ time-bounded algorithm $\mathcal{A}$ that on input $\langle 0^n, k \rangle$ outputs a circuit C_n of size at most $q(n,k)$ computing f_n.

ABP construction for $S^*_{n,k}$

A key ingredient to our algorithm for (k,n)-MLC is new explicit circuit upper bound for $S^*_{n,k}$.

Definition (Explicit Circuit Upper Bound)

A family $\{f_n\}_{n>0}$ of degree-k polynomials has $q(n,k)$-*explicit upper bounds* if there is an $O^*(q(n,k))$ time-bounded algorithm $\mathcal{A}$ that on input $\langle 0^n, k\rangle$ outputs a circuit C_n of size at most $q(n,k)$ computing f_n.

Hence, if $\{f_n\}$ has $q(n,k)$-explicit upper bounds then f_n can be evaluated in time $O^*(q(n,k))$.

ABP construction for $S^*_{n,k}$

ABP construction for $S^*_{n,k}$

Theorem

The family of symmetrized elementary polynomials $\{S_{n,k}\}_{n>0}$ has $\binom{n}{\downarrow k/2}$-explicit upper bounds over rationals and finite fields.

We use $\binom{n}{\downarrow r}$ to denote $\sum_{i=0}^{r} \binom{n}{i}$.

Nisan's result [**Ni91**] only assures the existence of an ABP for $S^*_{n,k}$ with $\binom{n}{\downarrow k/2}$ many nodes.

ABP construction for $S^*_{n,k}$

ABP construction for $S^*_{n,k}$

- Let us denote F as the family of subsets of $[n]$ of size exactly $k/2$ and $\downarrow \mathbb{F}$ as the family of subsets of $[n]$ of size at most $k/2$.

ABP construction for $S^*_{n,k}$

- Let us denote F as the family of subsets of $[n]$ of size exactly $k/2$ and $\downarrow \mathbb{F}$ as the family of subsets of $[n]$ of size at most $k/2$.

- For a subset $S \subset [n]$, we define $m_S = \prod_{j \in S} x_j$. Define

$$f_A = \sum_{\sigma \in S_{k/2}} \prod_{j=1}^{k/2} x_{i_{\sigma(j)}}$$

where $A \in F$ and $A = \{i_1, i_2, \ldots, i_{k/2}\}$, otherwise for subsets $S \notin F$, we define $f_S = 0$.

ABP construction for $S^*_{n,k}$

- Let us denote F as the family of subsets of $[n]$ of size exactly $k/2$ and $\downarrow \mathbb{F}$ as the family of subsets of $[n]$ of size at most $k/2$.
- For a subset $S \subset [n]$, we define $m_S = \prod_{j \in S} x_j$. Define

$$f_A = \sum_{\sigma \in S_{k/2}} \prod_{j=1}^{k/2} x_{i_{\sigma(j)}}$$

  where $A \in F$ and $A = \{i_1, i_2, \ldots, i_{k/2}\}$, otherwise for subsets $S \notin F$, we define $f_S = 0$.
- For each $S \in \downarrow \mathbb{F}$, let us define $\hat{f}_S = \sum_{S \subseteq A} f_A$ where $A \in F$.

ABP construction for $S^*_{n,k}$

ABP construction for $S^*_{n,k}$

Lemma

$$S^*_{n,k} = \sum_{S \in \downarrow \mathbb{F}} (-1)^{|S|} \hat{f}_S^2.$$

ABP construction for $S^*_{n,k}$

Lemma

$$S^*_{n,k} = \sum_{S \in \downarrow \mathbb{F}} (-1)^{|S|} \hat{f}_S^2.$$

Proof.

$$S^*_{n,k} = \sum_{A \in F} \sum_{B \in F} [A \cap B = \emptyset] f_A f_B$$

$$= \sum_{A \in F} \sum_{B \in F} \sum_{S \in \downarrow \mathbb{F}} (-1)^{|S|} [S \subseteq A \cap B] f_A f_B$$

$$= \sum_{S \in \downarrow \mathbb{F}} (-1)^{|S|} \left(\sum_{A \in F} [S \subseteq A] f_A \right)^2 = \sum_{S \in \downarrow \mathbb{F}} (-1)^{|S|} \hat{f}_S^2.$$

ABP construction for $S^*_{n,k}$

ABP construction for $S^*_{n,k}$

Lemma

There is an $\binom{n}{\downarrow k/2}$*-explicit multi-output ABP* B_1 *that outputs the collection* $\{f_A\}$ *for each* $A \in F$.

ABP construction for $S^*_{n,k}$

Lemma

There is an $\binom{n}{\downarrow k/2}$-explicit multi-output ABP B_1 that outputs the collection $\{f_A\}$ for each $A \in F$.

Proof.

- Note that, for each $A \in F$, f_A is the symmetrized polynomial m^*_A as already defined.

ABP construction for $S^*_{n,k}$

Lemma

There is an $\binom{n}{\downarrow k/2}$-explicit multi-output ABP B_1 that outputs the collection $\{f_A\}$ for each $A \in F$.

Proof.

- Note that, for each $A \in F$, f_A is the symmetrized polynomial m^*_A as already defined.
- Note that, $m^*_S = \sum_{j \in S} m^*_{S \setminus \{j\}} \cdot x_j$. Now, the construction of the ABP is obvious.

$\square$

ABP construction for $S^*_{n,k}$

ABP construction for $S^*_{n,k}$

Lemma

There is an $\binom{n}{\downarrow k/2}$-explicit multi-output ABP B_2 that outputs the collection $\{\hat{f}_S\}$ for each $S \in \downarrow \mathbb{F}$.

ABP construction for $S^*_{n,k}$

Lemma

There is an $\binom{n}{\downarrow k/2}$-explicit multi-output ABP B_2 that outputs the collection $\{\hat{f}_S\}$ for each $S \in \downarrow \mathbb{F}$.

Proof.

- Following [**BHKK09**], we define $\hat{f}_{i,S} = \sum_{S \subseteq A} f_A$ where $S \subseteq A$ and $A \cap [i] = S \cap [i]$. Note that, $\hat{f}_{n,S} = f_S$ and $\hat{f}_{0,S} = \hat{f}_S$.

ABP construction for $S^*_{n,k}$

Lemma

There is an $\binom{n}{\downarrow k/2}$*-explicit multi-output ABP* B_2 *that outputs the collection* $\{\hat{f}_S\}$ *for each* $S \in\downarrow \mathbb{F}$*.*

Proof.

- Following [**BHKK09**], we define $\hat{f}_{i,S} = \sum_{S \subseteq A} f_A$ where $S \subseteq A$ and $A \cap [i] = S \cap [i]$. Note that, $\hat{f}_{n,S} = f_S$ and $\hat{f}_{0,S} = \hat{f}_S$.
- From the definition, it is clear that $\hat{f}_{i-1,S} = \hat{f}_{i,S} + \hat{f}_{i,S\cup\{i\}}$ if $i \notin S$ and $\hat{f}_{i-1,S} = \hat{f}_{i,S}$ if $i \in S$.

□

ABP construction for $S^*_{n,k}$

ABP construction for $S^*_{n,k}$

For a noncommutative polynomial $f \in \mathbb{F}\langle X \rangle$ of degree k, such that $f = \sum_{m \in X^k} [m]f \cdot m$, define reverse of f,$f^R = \sum_{m \in X^k} [m]f \cdot m^R$ where m^R is the reverse of the word m.

ABP construction for $S^*_{n,k}$

For a noncommutative polynomial $f \in \mathbb{F}\langle X \rangle$ of degree k, such that $f = \sum_{m \in X^k} [m]f \cdot m$, define reverse of f,$f^R = \sum_{m \in X^k} [m]f \cdot m^R$ where m^R is the reverse of the word m.

Lemma

[Reversing an ABP] Suppose B is a multi-output ABP with r sink nodes where ith sink node computes $f_i \in \mathbb{F}\langle X \rangle$ for each $i \in [r]$. Then one can construct an ABP of twice the size of B that computes the polynomial $\sum_{i=1}^{r} f_i \cdot L_i \cdot f_i^R$ where L_i are affine linear forms.

ABP construction for $S^*_{n,k}$

For a noncommutative polynomial $f \in \mathbb{F}\langle X \rangle$ of degree k, such that $f = \sum_{m \in X^k} [m]f \cdot m$, define reverse of f,$f^R = \sum_{m \in X^k} [m]f \cdot m^R$ where m^R is the reverse of the word m.

Lemma

[Reversing an ABP] Suppose B is a multi-output ABP with r sink nodes where ith sink node computes $f_i \in \mathbb{F}\langle X \rangle$ for each $i \in [r]$. Then one can construct an ABP of twice the size of B that computes the polynomial $\sum_{i=1}^{r} f_i \cdot L_i \cdot f_i^R$ where L_i are affine linear forms.

Proof.

Connect the ABP with its mirror image. □

ABP construction for $S^*_{n,k}$

ABP construction for $S^*_{n,k}$

- Applying the construction of the previous lemma to the multi-output ABP B_2 with $L_S = (-1)^{|S|}$ we obtain an ABP that computes the polynomial $\sum_S (-1)^{|S|} \hat{f}_S \cdot \hat{f}_S^R$.

ABP construction for $S^*_{n,k}$

- Applying the construction of the previous lemma to the multi-output ABP B_2 with $L_S = (-1)^{|S|}$ we obtain an ABP that computes the polynomial $\sum_S (-1)^{|S|} \hat{f}_S \cdot \hat{f}_S^R$.
- Since $\hat{f}_S$ is a symmetrized polynomial, we note that $\hat{f}_S^R = \hat{f}_S$ and we conclude that this ABP computes $S^*_{n,k}$.

ABP construction for $S^*_{n,k}$

- Applying the construction of the previous lemma to the multi-output ABP B_2 with $L_S = (-1)^{|S|}$ we obtain an ABP that computes the polynomial $\sum_S (-1)^{|S|} \hat{f}_S \cdot \hat{f}_S^R$.
- Since $\hat{f}_S$ is a symmetrized polynomial, we note that $\hat{f}_S^R = \hat{f}_S$ and we conclude that this ABP computes $S^*_{n,k}$.
- That yields a $O(k\binom{n}{\downarrow k/2})$ size ABP.

Homogeneity is an Issue

- Note that, our ABP for $S^*_{n,k}$ is not homogeneous.
- Homogenization makes the number of edges quadratic to the number of nodes.
- Hence, we can not use the result of [**AJS09**] directly.

Definition ({0,1}-Homogeneous ABP)

At each layer, the edges are either all 0-edges or all 1-edges.

A Generalization of ABP-ABP Hadamard Product

Lemma

- *B_1 be an ABP of width w_1, ℓ_1 layers and each node has at most d_1 incoming edges.*

A Generalization of ABP-ABP Hadamard Product

Lemma

- *B_1 be an ABP of width w_1, ℓ_1 layers and each node has at most d_1 incoming edges.*
- *B_2 be an ABP of width w_2, ℓ_2 layers and each node has at most d_2 incoming edges.*

A Generalization of ABP-ABP Hadamard Product

Lemma

- *B_1 be an ABP of width w_1, ℓ_1 layers and each node has at most d_1 incoming edges.*
- *B_2 be an ABP of width w_2, ℓ_2 layers and each node has at most d_2 incoming edges.*
- *$B_1 \circ B_2$ can be computed by an ABP B of size at most $w_1 w_2(\ell_1 + \ell_2)$ and edges at most $d_1 d_2 w_1 w_2(\ell_1 + \ell_2)$.*

A Generalization of ABP-ABP Hadamard Product

Lemma

- B_1 *be an ABP of width* w_1, ℓ_1 *layers and each node has at most* d_1 *incoming edges.*
- B_2 *be an ABP of width* w_2, ℓ_2 *layers and each node has at most* d_2 *incoming edges.*
- $B_1 \circ B_2$ *can be computed by an ABP* B *of size at most* $w_1 w_2(\ell_1 + \ell_2)$ *and edges at most* $d_1 d_2 w_1 w_2(\ell_1 + \ell_2)$.
- B *can be computed in deterministic* $O^*(d_1 d_2 w_1 w_2(\ell_1 + \ell_2))$ *time.*

A Generalization of ABP-ABP Hadamard Product

Proof Idea. Use padding layers so that B_1, B_2 have same number of layers and for each layer, both compute polynomials of same degree.

$$f_i' \circ g_j' = \left(\sum_{s \in S_{1,i}} f_s \cdot L_{s,i}^{\{1\}} \right) \circ \left(\sum_{s \in S_{2,j}} g_s \cdot L_{s,j}^{\{2\}} \right)$$

$$= \left(\sum_{(s,s') \in S_{1,i} \times S_{2,j}} (f_s \circ g_{s'}) \cdot (L_{s,i}^{\{1\}} \circ L_{s',j}^{\{2\}}) \right).$$

Beating the Brute Force

- Our ABP solves (k,n)-$\textsc{MLC}$ when the input polynomial is given by an ABP.

Beating the Brute Force

- Our ABP solves (k,n)-$\textsc{MLC}$ when the input polynomial is given by an ABP.
- What can we say when the input polynomial is given by the circuits?

Beating the Brute Force

- Our ABP solves (k,n)-$\textsc{MlC}$ when the input polynomial is given by an ABP.
- What can we say when the input polynomial is given by the circuits?
- We can not use the result of [**AJS09**] directly.

Beating the Brute Force

- Our ABP solves (k,n)-MLC when the input polynomial is given by an ABP.
- What can we say when the input polynomial is given by the circuits?
- We can not use the result of [**AJS09**] directly.
- A circuit of size s computing a polynomial of degree k can be converted to an ABP of size $s^{O(\log k)}$.

Thank You