Electronic Colloquium on Computational Complexity, Report No. 71 (2022)

Robustly Separating the Arithmetic Monotone Hierarchy Via Graph Inner-Product

Arkadev Chattopadhyay[*] Utsab Ghosal[†] Partha Mukhopadhyay[‡]

May 13, 2022

Abstract

We establish an ϵ-sensitive hierarchy separation for monotone arithmetic computations. The notion of ϵ-sensitive monotone lower bounds was recently introduced by Hrubeš [Hru20]. We show the following:

- There exists a monotone polynomial over n variables in VNP that cannot be computed by $2^{o(n)}$ size monotone circuits in an ϵ-sensitive way as long as $\epsilon \geq 2^{-\Omega(n)}$.
- There exists a polynomial over n variables that can be computed by polynomial size monotone circuits but cannot be computed by any monotone arithmetic branching program (ABP) of $n^{o(\log n)}$ size, even in an ϵ-sensitive fashion as long as $\epsilon \geq n^{-\Omega(\log n)}$.
- There exists a polynomial over n variables that can be computed by polynomial size monotone ABP but cannot be computed in $n^{o(\log n)}$ size by monotone formulas even in an ϵ-sensitive way, when $\epsilon \geq n^{-\Omega(\log n)}$.
- There exists a polynomial over n variables that can be computed by width-4 polynomial size monotone arithmetic branching programs (ABPs) but cannot be computed in $2^{o(n^{1/d})}$ size by monotone, unbounded fan-in formulas of product depth d even in an ϵ-sensitive way, when $\epsilon \geq 2^{-\Omega(n^{1/d})}$. This yields an ϵ-sensitive separation of constant-depth monotone formulas and constant-width monotone ABPs. It seems that even an ordinary separation of the two classes was not known.

An interesting feature of our separations is that in each case the polynomial exhibited is obtained from a *graph inner-product* polynomial by choosing an appropriate graph topology. The closely related graph inner-product Boolean function for expander graphs was invented by Hayes [Hay11], also independently by Pitassi [Pit09], in the context of *best-partition* multiparty communication complexity.

1 Introduction

While considerable progress has been made in monotone complexity, several fundamental problems remain open. In particular, it is known that monotone lower bounds of a certain kind are enough to imply the major breakthrough of obtaining strong general circuit lower bounds. In Boolean complexity, it has long been known [Val86] that monotone circuit lower bounds for slice functions are sufficient to yield general lower bounds. In the context of arithmetic complexity, Hrubeš [Hru20] recently formulated an analogous result by showing that ϵ-sensitive monotone lower bounds for arbitrarily small but non-zero ϵ yield lower bounds even for non-monotone circuits. More generally, consider F_n to be the full polynomial,

[*] TIFR, Mumbai. Partially supported by a MATRICS grant of the Science and Engineering Research Board, DST, India. arkadev.c@tifr.res.in

[†] CMI,Chennai ghosal@cmi.ac.in

[‡] CMI, Chennai. partham@cmi.ac.in

ISSN 1433-8092

$(1 + x_1 + x_2 + \cdots x_n)^n$, of degree n that obviously has a simple monotone circuit. For any monotone polynomial f, Hrubeš showed that super-polynomial lower bounds on the monotone circuit (branching program, formula) size for $F_n + \epsilon \cdot f$ for arbitrarily small $\epsilon > 0$, yields general lower bounds on circuit (branching program, formula[1]) size for computing f. Setting $F_{n,m} := \prod_{i=1}^{n}(x_{i,1} + \cdots + x_{i,m})$ to be the full set-multilinear polynomial (over $n \times m$ variables) and proving such ϵ-sensitive bounds yields general set-multilinear bounds. Hrubeš argues that proving ϵ-sensitive lower bounds even for moderately small ϵ seems to be non-trivial as it'd require exploiting information of the values of coefficients of monomials appearing in f. Most techniques employed for proving lower bounds in general, and for monotone lower bounds in particular, ignore the specific values of coefficients. They use the structure of the support set of the monomials alone. With respect to such techniques, the determinant and permanent polynomials, two polynomials that Valiant's VP vs. VNP conjecture asserts have very different complexities, remain equivalent. Some recent works that are able to exploit the values of coefficients are that of Yehudayoff [Yeh19] and the work of Srinivasan [Sri20] that builds upon the former. However, these techniques have not yielded so far ϵ-sensitive lower bounds. Such bounds were recently obtained by Chattopadhyay, Datta and Mukhopadhyay [CDM21] and by Chattopadhyay et.al. [CDGM22], adapting techniques from 2-party communication complexity.

Motivated towards gaining a better understanding of ϵ-sensitive computations, we revisit the question of separating the powers of some of the key monotone arithmetic models: circuits, branching programs, formulas and constant-depth (unbounded fan-in) formulas. Arvind, Joglekar and Srinivasan [AJS09] proved that constant-depth monotone formulas are strictly less powerful than monotone formulas of unrestricted depth by considering a specialized polynomial. Hrubeš and Yehudayoff [HY11] showed that elementary symmetric polynomials cannot be computed in polynomial size by monotone formulas. This provides a separation of the powers of monotone formulas from that of monotone ABPs. Later, a different work of Hrubeš and Yehudayoff [HY16] showed a similar separation of the power of monotone ABPs and monotone circuits. This required the construction of an altogether different polynomial. Very recently, Komarath, Pandey and Rahul [KPR22] provided a unified treatment of these separations (not including the separation between constant-depth formulas and formulas of unrestricted depth) by making use of graph homomorphism polynomials. It is natural to ask if these separations can be strengthened/made robust in the following way: can we exhibit a polynomial f that has polynomial size monotone circuits (ABPs) but even $F_n + \epsilon \cdot f$ has no polynomial size monotone ABP (formula)?

The main contribution of this work is to provide the first such robust separations between the power of monotone circuits, ABPs, formulas and constant-depth circuits that are additionally done in a unified way. Our separations build on the connection developed in [CDM21, CDGM22] between randomized communication complexity and ϵ-sensitive lower bounds. In particular, this allows us to exhibit a general framework to define *graph inner-product* polynomials such that simply changing the graph appropriately yields the required polynomial for each of our separations. More precisely, given an undirected graph G on k vertices and a number m, we first define a Boolean function, called the graph inner-product function and denoted by $\text{IP}_G : \{0,1\}^{k \times m} \to \{1, -1\}$. Let $V(G) := \{u_1, \ldots, u_k\}$. We identify each vertex with a variable $\overrightarrow{u_i}$ that takes m-bit binary vectors as values. Then,

$$\text{IP}_G(\overrightarrow{u_1}, \ldots, \overrightarrow{u_k}) := \Big(-1\Big)^{\sum\limits_{(u_i,u_j)\in E(G)} \langle \overrightarrow{u_i}, \overrightarrow{u_j} \rangle},$$

where $\forall (u_i, u_j) \in E(G)$, $\langle \overrightarrow{u_i}, \overrightarrow{u_j} \rangle := \sum_{t\in[m]} u_t^i \cdot u_t^j$ and $\overrightarrow{u_i} = (u_1^i, u_2^i, \ldots, u_m^i) \in \{0,1\}^m$.

[1] The case of formulas comes with the following subtlety: Hrubeš' argument uses a homogeneization trick. Unlike circuits or ABPs, we don't know yet if formulas can be homogeneized without significant blow-up [Sap21, Subsection 5.1]. This seemingly prevents a direct application of Hrubeš' argument to formulas. Nevertheless, using the fact that size s ABPs can be simulated by size $s^{\log s}$ formulas, one concludes quite easily that an ϵ-sensitive monotone lower bound for formulas of the form $n^{(\log n)^{1+\delta}}$ for any $\delta > 0$, is sufficent to imply super-polynomial lower bounds even for general ABPs.

We consider a set-multilinear polynomial over an input matrix X of dimension $k \times n$ with entries $X[i,j] := x_{i,j}$ of indeterminates, and $n = 2^m$. The monomials will naturally encode satisfying assignments to the Boolean graph inner product function defined above. Towards this, define $\mathbb{M}[X]$ to be the set of all set-multilinear monomials of degree k over $X = \{X_i | \, i \in [k]\}$, where $\forall i \; X_i = \{x_{i,j} | \, j \in [n]\}$. With every map $\nu : [k] \to [n]$, we identify a monomial $\kappa_\nu \in \mathbb{M}[X]$ as $m_\nu := \prod_{i=1}^{k} x_{i,\nu(i)}$. This forms a bijection between set $\mathbb{M}[X]$ and $\mathrm{T} = \{\nu | \, \nu : [k] \to [n]\}$. Now each map $\nu \in \mathrm{T}$ can be identified by a k tuple of m-bit vectors $(\overrightarrow{\nu_1}, \ldots, \overrightarrow{\nu_k})$ where for every $i \in [k]$ $\overrightarrow{\nu_i}$ is the binary representation of $\nu(i) \in [n]$. So in this way, given map $\nu : [k] \to [n]$, any set-multilinear degree k monomial $\kappa = x_{1,\nu(1)} \cdots x_{k,\nu(k)}$ corresponds to a k tuple of m-bit vectors $\widetilde{\kappa} = \{\overrightarrow{\nu_1}, \ldots, \overrightarrow{\nu_k}\}$ where each $\overrightarrow{\nu_t}$ is the binary representation of $\nu(t)$.

Let,

$$f_{G,m} := \sum_{\mathrm{IP}_G(\widetilde{\kappa}) = -1} \kappa$$

be called the $\mathrm{IP}_{G,m}$ polynomial.

Further, let

$$F_{k,n} := \prod_{i=1}^{k} (x_{i,1} + \cdots + x_{i,n}),$$

be the full set-multilinear polynomial over $\mathbb{M}[X]$.

We can now state our first theorem that implies the first strongly exponential ϵ-sensitive monotone lower bounds for an explicit (monotone) polynomial in VNP. Recall that $n = 2^m$.

Theorem 1.1. *Let G be a constant-degree expander graph on k vertices. Then, there exists a constant $c > 0$ such that any monotone circuit computing either of the polynomial $F_{k,n} \pm \epsilon \cdot f_{G,m}$ has size $2^{\Omega(km)}$ as long as $\epsilon \geq 2^{-ckm}$.*

Remark 1.1. *Plugging $m = 1$ in Theorem 1.1, we recover the claimed strongly exponential lower bound as long as $\epsilon = 2^{-\Omega(n)}$.*

The VNP upper bound for the polynomial $f_{G,m}$ follows from Valiant's criterion [Val79, Proposition 4]. Our second theorem obtains an ϵ-sensitive separation between monotone circuits and monotone ABPs.

Theorem 1.2. *Let T be the full binary tree on k vertices. Then, $f_{T,m}$ can be computed by monotone circuits of size $O(kn^3)$. On the other hand, there exists a constant $c > 0$ such that any monotone ABP computing either of the polynomial $F_{k,n} \pm \epsilon \cdot f_{T,m}$ has size $k^{\Omega(m)}$ as long as $\epsilon \geq k^{-cm}$.*

We next provide an analogous separation between monotone ABPs and monotone formulas by considering a path.

Theorem 1.3. *Let Γ be a simple path on k vertices. Then, the polynomial $f_{\Gamma,m}$ can be computed by a monotone ABP of size $O(kn)$. On the other hand, there exists a constant $c > 0$ such that any monotone formula computing either of the polynomial $F_{k,n} \pm \epsilon \cdot f_{\Gamma,m}$ has size $k^{\Omega(m)}$ as long as $\epsilon \geq k^{-cm}$.*

Finally, we provide a separation between constant-depth monotone formulas and monotone formulas. In fact, we provide a stronger separation, that of monotone constant-depth formulas and constant-width ABPs.

Theorem 1.4. *Let Γ be a simple path on k vertices. Then, the polynomial $f_{\Gamma,1}$ can be computed by a monotone width-4 ABP of size $O(k)$. On the other hand, there exists a constant $c > 0$ such that any monotone formula of product-depth d computing either of the polynomial $F_{k,2} \pm \epsilon \cdot f_{\Gamma,1}$ has size $2^{k^{\Omega(1/d)}}$ as long as $\epsilon \geq 2^{-ck^{1/d}}$.*

It is known that poly-size bounded-depth monotone formulas can be simulated by poly-size bounded-width monotone ABPs which in turn can be simulated by monotone formulas of unrestricted depth and polynomial size. Our result, thus in particular, shows that the class of polynomials computed by constant-depth monotone formulas of polynomial size are strictly contained (in a strong sense) in the class of polynomials having bounded-width monotone ABPs of polynomial size. As far as we know, such a result was not known before.

1.1 Outline of our Technique

We build upon the insight relating ϵ-sensitive lower bounds with the measure of discrepancy under universal distributions, originating in [CDM21, CDGM22]. Both of these works prove lower bounds against monotone circuits for polynomials that are not known to have any efficient monotone computation. The main concern here is to prove separations in the monotone hierarchy. Thus, the new challenge is to come up with far '*easier polynomials*' to which a discrepancy like measure can still be applied. The canonical example of a 'function' to which the discrepancy method applies in communication complexity is that of Inner-product. However, one important difference between the setting of standard 2-party communication complexity and arithmetic circuits is that while in the former, every rectangle that appears in a rectangular decomposition conforms to the same partition of inputs among the players, in the latter each product polynomial appearing in a decomposition is free to have its own partition of the input variables. Indeed, the standard Inner-product function becomes trivial for the best (w.r.t the players) partition.

This is why we turn to best-partition communication complexity, a slightly non-standard model. Hayes [Hay11], and independently Pitassi [Pit09], designed an appropriate version of the Inner-product function that they called Graph inner-product (see the definition from the Introduction) which they proved is hard against all (balanced) partitions. To do this, they chose the graph to be a constant-degree expander. A standard property of such an expander is that the size of any balanced cut is large. This allows one to say that for every possible partition of inputs among the players, one can induce a large copy of the standard inner-product as a sub-function for which the given partition is the worst for the players. This does not immediately give us a monotone arithmetic circuit lower bound. To get there, we need to argue against *many partitions appearing together* in the decomposition. This is where we use the idea from [CDGM22] of discrepancy w.r.t universal distributions that is a measure which is partition independent. This gives us our Theorem 1.1, an ϵ-sensitive strongly exponential lower bounds against monotone circuits.

How does one tune the complexity of a graph Inner-product polynomial so that it becomes easy for monotone circuits but remains robustly hard against monotone branching programs? The starting point is to look at the (not robust) separation of these two classes by Hrubeš and Yehudayoff [HY16]. They showed that there is a subtle difference between decomposition theorems for ABPs and that of circuits. ABPs give rise to a slightly more structured decomposition where, for every r, we can force each product polynomial to have a partition such that one part has size exactly r. They further showed that for the full binary tree on k vertices, any cut where one side has $r(k)$ vertices, has size $\Omega(k)$. Exploiting this insight, we establish Theorem 1.2 by proving a universal discrepancy bound for all such partitions on the binary-tree inner-product function.

To separate formulas from ABPs, we use the fact that the decomposition theorem for formulas provides even more structure. Here, every polynomial of degree k appearing in a decomposition can be written as a product of about $\log k$-many polynomials rather than two. This corresponds, with the usual caveat of mixed vs. best partition, to the case of the best-partition $\log k$-party number-in-the-hand model of randomized communication complexity. While the goal of having an efficient ABP upper bound for the polynomial forces the corresponding 2-party game to be easy for at least some partition, choosing the graph to be a simple path ends up having the following simple but remarkable feature: for every balanced $\log k$-wise partition of the inputs, one can design a 2-party game with a hard partition. This reduces the task to proving a discrepancy bound for this hard 2-party partition which drives Theorem 1.3.

To separate monotone ABPs of constant width from monotone constant-depth (but unbounded fan-in) formulas, we first note that keeping the path inner-product polynomial becomes easy for ABPs of constant width once we restrict each node in the path to have two (or any constantly many) variables instead of n. Finally, we exploit the super-structured decomposition theorem for constant-depth (but unbounded fan-in) formulas. Each product polynomial now is the product of $k^{1/d}$-many set-multilinear product polynomials, where d is the product-depth of the formula. Using similar ideas as in the proof of Theorem 1.3 with this additional structure, we establish appropriate discrepancy bounds to yield Theorem 1.4.

1.2 Other Related Work

Strongly exponential lower bounds on monotone arithmetic circuits were obtained in several previous works as well (see for example [GS12, RY11, Sri20, CKR20, HY21, CDGM22]). Among them, the idea of using the discrepancy method appears in Raz and Yehudayoff [RY11] and in Chattopadhyay et.al. [CDGM22]. The latter work did prove exponential ϵ-sensitive lower bounds for the spanning tree polynomial that is known to be in VP. Incomparably, our Theorem 1.1 proves a strongly exponential ϵ-sensitive lower bound, but for a polynomial in VNP. ϵ-sensitive lower bounds were not proved in other works.

Some separations related to our Theorems 1.2-1.4 have been obtained in other works. For instance, Dvir et. al. [DMPY12] showed that multilinear branching programs are strictly more powerful that multilinear formulas. As their witnessing polynomial is computable by, in fact, a monotone branching program, their separation subsumes a simple separation of the corresponding monotone models. However, our Theorem 1.3 provides an ϵ-sensitive lower bound for formulas for a polynomial efficiently computed by branching programs. Thus, the two separations seem incomparable. Similarly, the work of Raz and Yehudayoff [RY09], and later by Chillara et.al. [CELS18], provided, among other things, an exponential separation of the multilinear models of formulas and constant-depth, unbounded fan-in formulas via monotone polynomials computed efficiently by monotone formulas of unrestricted depth. For the same reason as before, this separation seems incomparable to the ϵ-sensitive separation provided by our Theorem 1.4, although both imply a separation of the simpler corresponding monotone models.

Finally, Arvind, Joglekar and Srinivasan [AJS09] also obtained a monotone depth-hierarchy theorem. In the process, they also considered the model of bounded-width (monotone) circuits, a model quite related to bounded-width (monotone) branching programs. They proved a monotone width-hierarchy theorem as well. As far as we can tell, their work does not separate the class of polynomials having efficient monotone bounded-depth formulas from the class of polynomials having efficient monotone bounded-width circuits, although it is known that the latter contains the former. Our Theorem 1.4, on the other hand, shows that already width-4 monotone ABPs can compute polynomials that require exponential size to be computed by constant-depth monotone circuits, even in an ϵ-sensitive sense as long as ϵ is inverse exponentially large.

Organization

The paper is organized as follows. We provide some background mainly on communication complexity and monotone algebraic complexity in Section 2. In Section 3, we prove results on discrepancy bounds for graph inner product function. The proof of Theorem 1.1 is given in Section 4 that shows ϵ-sensitive lower bound for monotone circuits. Section 5 contains the proof of Theorem 1.2 which shows the separation between monotone circuits and ABPs. Finally in Section 6, we provide the proofs of Theorem 1.3 and Theorem 1.4 establishing the separations between monotone ABPs vs formulas, and constant width ABPs vs constant depth formulas.

2 Preliminaries

Notation

Let $[n] = \{1, 2, \ldots, n\}$. Polynomials are always considered over $\mathbb{R}[X]$ where $\mathbb{R}$ is the set of reals.

Set-multilinear Polynomials

Let $X = \cup_{i=1}^{k} X_i$ be a set of variables where $X_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,n}\}$. A polynomial $f \in \mathbb{R}[X]$ is set-multilinear if each monomial in f respects the partition given by the set of variables $X_1, X_2, \ldots, X_k$. In other words, each monomial κ in f is of the form $x_{1,j_1} x_{2,j_2} \cdots x_{k,j_k}$. For the purpose of the paper it is also useful to think the variables are from a matrix $M_{k \times n}$ where the i^{th} row is $\{x_{i,1}, x_{i,2}, \ldots, x_{i,n}\}$.

Ordered Polynomials

For a monomial of the form $m = x_{i_1,j_1} x_{i_2,j_2} \cdots x_{i_k,j_k}$ we define the set $I(m) = \{i_1, i_2, \ldots, i_k\}$. If a polynomial f has the same set $I(m)$ for every monomial occurring it it with a nonzero coefficient, then we say that the polynomial is ordered and we write $I(f) = I(m)$ for each m. Clearly, the set-multilinear polynomials are ordered polynomials with $I(f) = \{1, 2, \ldots, k\}$.

2.1 Structure of Monotone Circuits

The main structural result for monotone circuits that we use throughout, is the following theorem.

Theorem 2.1. *[Yeh19, Lemma 1] Let $n > 2$ and $f \in \mathbb{R}[X]$ be an ordered monotone polynomial with $I(p) = [k]$. Let C be a monotone circuit of size s that computes f. Then, we can write*

$$f = \sum_{t=1}^{s} a_t \cdot b_t$$

where a_t and b_t are monotone ordered polynomials with $\frac{k}{3} \leq |I(a_t)| \leq \frac{2k}{3}$ and $I(b_t) = [k] \setminus I(a_t)$. Moreover, $a_t b_t \leq f$ for each $1 \leq t \leq s$, by which we mean that the coefficient of any monomial in $a_t b_t$ is bounded by the coefficient of the same monomial in f.

A partition $P = (A, B)$ of $[k]$ is said to be perfectly balanced if $|A| = |B| = \frac{k}{2}$ and is said to be nearly balanced if $\frac{k}{3} \leq |A|, |B| \leq \frac{2k}{3}$. An ordered product polynomial $a \cdot b$ on n variables is said to be nearly balanced if $\frac{k}{3} \leq |I(a)|, |I(b)| \leq \frac{2k}{3}$.

2.2 Structure of Monotone ABPs

We recall the definition of algebraic branching programs (ABPs).

Definition 2.1 (Algebraic Branching Program)**.** *An* algebraic branching program *(ABP) is a layered directed acyclic graph. The vertex set is partitioned into layers $0, 1, \ldots, k$, with directed edges only between adjacent layers (i to $i+1$). There is a* source *vertex of in-degree 0 in layer 0, and one out-degree-0* sink *vertex in layer k. Each edge is labeled by an affine $\mathbb{F}$-linear form where $\mathbb{F}$ is the underlying field. The polynomial computed by the ABP is the sum over all source-to-sink directed paths of the ordered product of affine forms labeling the path edges.*

The following structure theorem is well-known.

Theorem 2.2. *[HY16, Lemma 3] Let f be a degree k homogeneous monotone set-multilinear polynomial computed by a size s ABP. Then for every $j \in [k]$ there exists s pairs of monotone ordered set-multilinear polynomials $\{g_i, h_i \,|i \in [s]\}$ such that*

$$f = \sum_{i=1}^{s} g_i \cdot h_i$$

where for every i, $|I(g_i)| = j$ and $|I(h_i)| = k - j$. $(I(g_i), I(h_i))$ gives a partition of $[k]$.

2.3 Structure of (Monotone) Set-Multilinear Formulas

Definition 2.2. *(Monotone) Set-Multilinear-*log*-Product Polynomials.*
*A degree k polynomial f defined over a $k \times n$ matrix M of variables is called a (monotone) set-multilinear-*log*-product polynomial if there exists p (monotone) set-multilinear polynomials $f_1, \ldots, f_p$ such that the following holds.*

1. $f = \prod_{i=1}^{p} f_i$.
2. $\forall i \in [p-1],\ \ (\frac{1}{3})^i k \le |I(f_i)| \le (\frac{2}{3})^i k$ *where $I(f_i)$ is the set of rows of M on which the polynomial f_i is defined.*
3. $\forall\, i \ne j,\ \ I(f_i) \cap I(f_j) = \emptyset$.
4. $|I(f_p)| = 1$.

The following structure theorem is well-known and proved in [HY11]. However, we include a self-contained proof in the appendix for completeness.

Theorem 2.3. *[HY11, Lemma 4] Let f be a degree k set-multilinear polynomial computed by a (monotone) formula of size s. Then there exists (monotone) set-multilinear-*log*-product polynomials $g_1, g_2, \ldots, g_{s'}$ such that $s' \le s$,*

$$f = g_1 + g_2 + \cdots + g_{s'}.$$

2.4 Structure of (Monotone) Set-Multilinear Constant Depth Formula

Definition 2.3. *(p, ℓ)-Form*
A degree k (monotone) set-multilinear polynomial f defined over a matrix $M_{k\times n}$ of variables has a (p, ℓ)-from if there exists p (monotone) set-multilinear polynomials $f_1, \ldots, f_p$ such that the following holds.

1. $f = \prod_{i=1}^{p} f_i$.
2. $\forall\ i\ \in\ [p],\ \ |I(f_i)| \ge \ell$, *where $I(f_i)$ is the set of rows of $M_{k\times n}$ on which the polynomial f_i is defined.*
3. $\forall\, i \ne j,\ \ I(f_i) \cap I(f_j) = \emptyset$.

The following theorem is a re-statement of Lemma 9 in [HY11]. For completeness, the proof is included in the appendix.

Theorem 2.4. *[HY11, Lemma 9] Let f be a degree k set-multilinear polynomial computed by a (monotone) formula of size s and product depth d. Let $q > 1$ be a natural number such that $k > (2q)^d$. Then there exists (monotone) set-multilinear-$(q, k(2q)^{-d})$-form polynomials $g_1, g_2, \ldots, g_{s'}$ such that $s' \le s$,*

$$f = g_1 + g_2 + \cdots + g_{s'}.$$

2.5 Communication Complexity

We recall some basic results from communication complexity. The details can be found in [KN06]. Let us very briefly first recall basic notions in the 2-party communication model of Yao. The joint input space of Alice and Bob is $\{0,1\}^m \times \{0,1\}^m$ with each player receiving an m-bit Boolean string, and they want to evaluate a Boolean function $F : \{0,1\}^m \times \{0,1\}^m \rightarrow \{-1,1\}$. One defines a combinatorial rectangle R as a product set $A \times B$, for some $A, B \subseteq \{0,1\}^m$. Put another way, R is just a sub-matrix of the $2^m \times 2^m$ communication matrix M_F of the function F, that Alice and Bob want to compute. The rows of this matrix are indexed by possible inputs of Alice and the columns by the ones of Bob and $M_F(x,y) = F(x,y)$. One of the important notions is discrepancy. For a rectangle R, the discrepancy $\mathrm{Disc}_\delta(F,R) := \left|\delta(R \cap F^{-1}(1)) - \delta(R \cap F^{-1}(-1))\right|$ where δ is a distribution on the input space $\{0,1\}^m \times \{0,1\}^m$. In other words,

$$\mathrm{Disc}_\delta(F,R) = \left| \mathop{\mathbb{E}}_{(x,y)\sim\delta}[F(x,y)R(x,y)] \right|.$$

The discrepancy of F under δ is defined as

$$\mathrm{Disc}_\delta(F) := \max_R \mathrm{Disc}_\delta(F,R).$$

In this work, we will be forced to look at variable partition models. That is, the n input bits will be partitioned among Alice and Bob in multiple ways. Each such partition P has its own set of rectangles, denoted by $\mathcal{R}(P)$. Hence, we define,

$$\mathrm{Disc}_{\delta,P}(F) := \max_{R \in \mathcal{R}(P)} \mathrm{Disc}_\delta(F,R).$$

The 2-party model extends to t-party model naturally. Here, we have t players $P_1, P_2, \dots, P_t$ and the joint input space is $\{0,1\}^{m_1} \times \{0,1\}^{m_2} \times \ldots \times \{0,1\}^{m_t}$. Player P_i receives an input from $\{0,1\}^{m_i}$. Together, they want to compute a function $F : \{0,1\}^{m_1} \times \{0,1\}^{m_2} \times \cdots \times \{0,1\}^{m_t} \rightarrow \{-1,1\}$. We can similarly define a combinatorial rectangle $R = R_1 \times \cdots \times R_t$ where each $R_i \subseteq \{0,1\}^{m_i}$. For any distribution δ over the input space $\{0,1\}^{m_1} \times \{0,1\}^{m_2} \times \cdots \times \{0,1\}^{m_t}$ and a rectangle R we define

$$\mathrm{Disc}^t_\delta(F,R) := \left|\delta(R \cap F^{-1}(1)) - \delta(R \cap F^{-1}(-1))\right|. \tag{1}$$

Now we define the discrepancy of F in the following way,

$$\mathrm{Disc}^t_\delta(F) := \max_R \mathrm{Disc}^t_\delta(F,R)$$

Exactly like in the two-party case, we will be considering the t-party discrepancy w.r.t multiple t-way partitions. Hence, given such a partition P, we analogously define $\mathrm{Disc}^t_{\delta,P}(F)$.
Let us recall the inner product function,

$$\mathrm{IP}_m(x,y) := \Big(-1\Big)^{\sum_{i=1}^m x_i y_i},$$

that is widely studied. It is well-known that the two-party discrepancy of the inner product function is small under the uniform distribution $\mathcal{U}$ over $\{0,1\}^m \times \{0,1\}^m$. This was first proved by Chor and Goldreich [CG88]. A self-contained proof can be found in [KN06].

Theorem 2.1. *[KN06, Example 3.29] Under the uniform distribution $\mathcal{U}$ over $\{0,1\}^m \times \{0,1\}^m$, $\mathrm{Disc}_\mathcal{U}(\mathrm{IP}_m) = 2^{-\Omega(m)}$.*

3 Discrepancy Bound for Graph Inner Product

3.1 Graph Inner Product Function

Given a graph G on k vertices $\{u_1, u_2, \ldots, u_k\}$, we identify each vertex u_i with variable $\overrightarrow{u_i}$ which takes m-bit binary vectors as values. Now we define the following function

$$\mathrm{IP}_G(\overrightarrow{u_1}, \ldots, \overrightarrow{u_k}) = \Big(-1 \Big)^{\sum_{(u_i,u_j)\in E(G)} \langle \overrightarrow{u_i}, \overrightarrow{u_j} \rangle}$$

where $\forall (u_i, u_j) \in E(G)$ and $\langle \overrightarrow{u_i}, \overrightarrow{u_j} \rangle = \sum_{t\in[m]} u^i_t \cdot u^j_t$ and $\overrightarrow{u_i} = (u^i_1, u^i_2, u^i_3, \ldots, u^i_m) \in \{0,1\}^m$.

3.2 Graph Inner Product Polynomial

Consider the input matrix X of dimension $k \times n$ with entries $X[i,j] := x_{i,j}$ of indeterminates, and $n = 2^m$. Define $\mathbb{M}[X]$ to be the set of all set-multilinear monomials of degree k over $X = \{X_i |\ i \in [k]\}$, where $\forall i\ X_i = \{x_{i,j} |\ j \in [n]\}$. With every map $\nu : [k] \to [n]$, we identify a monomial $\kappa_\nu \in \mathbb{M}[X]$ as $\kappa_\nu := \prod_{i=1}^{k} x_{i,\nu(i)}$. This forms a bijection between set $\mathbb{M}[X]$ and $\mathrm{T} = \{\nu |\ \nu : [k] \to [n]\}$.
Now each map $\nu \in \mathrm{T}$ can be identified by a k tuple of m-bit vectors $(\overrightarrow{\nu_1}, \ldots, \overrightarrow{\nu_k})$ where for every $i \in [k]$ $\overrightarrow{\nu_i}$ is the binary representation of $\nu(i) \in [n]$. So in this way, given map $\nu : [k] \to [n]$, any set-multilinear degree k monomial $\kappa = x_{1,\nu(1)} \cdots x_{k,\nu(k)}$ corresponds to a k tuple of m-bit vectors $\overrightarrow{\kappa} = \{\overrightarrow{\nu_1}, \ldots, \overrightarrow{\nu_k}\}$. Let the polynomial,

$$f_{G,m} := \sum_{\mathrm{IP}_G(\overrightarrow{\kappa})=-1} \kappa$$

be denoted as $\mathrm{IP}_{G,m}$ and be called the G-Inner product polynomial.

3.3 Discrepancy and Lower Bound Correspondence

Recently in [CDGM22], a correspondence between ϵ-sensitive monotone lower bound for a $0-1$ coefficient polynomial f an appropriate communication problem has been established via the discrepancy measure. There the authors only considered two-party communication problem. Here we extend it to t-party communication problem for $t \geq 2$. Let $(A_1, A_2, \ldots, A_t)$ be any partition of $[k]$ and let there be t players $P_1, \ldots, P_t$. The players are given a map $\nu : [k] \to [n]$ in a distributed fashion, i.e, P_i gets a map $\nu_i : A_i \to [n]$. They jointly want to decide if the monomial κ_ν is present in polynomial f or not. In other words, the players want to compute a Boolean function, denoted by $C^f : \{0,1\}^{k\times m} \to \{1,-1\}$, where $C^f(\nu(1), \ldots, \nu(k)) = -1$ if monomial κ_ν has coefficient 1 in f and otherwise C^f evaluates to 1. Inspecting the proof in [CDGM22], it is easy to observe that the lower bound technique is independent of the monotone computation model. In particular, it applies to ABPs, formulas and constant depth formulas using their respective structure theorems. that is Theorem 2.2, Theorem 2.3 and Theorem 2.4. More precisely, we can restate their result in the following form.

Theorem 3.1. *[CDGM22, Adaptation of their Theorem 1.3] Let f be any $0-1$ set-multilinear monotone polynomial defined over a matrix of variables of dimension $k \times n$. Let Δ be a distribution over $[k]^n$.*

1. *The monotone circuit complexity of $F_{k,n} - \epsilon \cdot f$ (resp. $F_{k,n} + \epsilon \cdot f$) is at least $\frac{\epsilon}{3\gamma}$ (resp. $\frac{\epsilon}{6\gamma}$) as long as $\epsilon \geq \frac{6\gamma}{1-3\gamma}$ (resp. $\epsilon \geq \frac{6\gamma}{1-12\gamma}$), where $\gamma := \max_P \mathrm{Disc}_{\Delta,P}(C^f)$ and P is any nearly balanced 2-wise partition of $[k]$.*

2. *Let $r \in [k]$. Then the monotone ABP complexity of $F_{k,n} - \epsilon \cdot f$ (resp. $F_{k,n} + \epsilon \cdot f$) is at least $\frac{\epsilon}{3\gamma}$ (resp. $\frac{\epsilon}{6\gamma}$) as long as $\epsilon \geq \frac{6\gamma}{1-3\gamma}$ (resp. $\epsilon \geq \frac{6\gamma}{1-12\gamma}$), where $\gamma := \max_P \mathrm{Disc}_{\Delta,P}(C^f)$. Here the max runs over all 2-wise partitions $P = (A, B)$ of $[k]$ such that $|A| = r$.*

3. *The monotone formula complexity of $F_{k,n} - \epsilon \cdot f$ (resp. $F_{k,n} + \epsilon \cdot f$) is at least $\frac{\epsilon}{3\gamma}$ (resp. $\frac{\epsilon}{6\gamma}$) as long as $\epsilon \geq \frac{6\gamma}{1-3\gamma}$ (resp. $\epsilon \geq \frac{6\gamma}{1-12\gamma}$), where $\gamma := \max_P \mathrm{Disc}^t_{\Delta,P}(C^f)$, P runs over all t-wise partitions with $t = \Omega(\log k)$.*

4. *The monotone product depth d formula complexity of $F_{k,n} - \epsilon \cdot f$ (resp. $F_{k,n} + \epsilon \cdot f$) is at least $\frac{\epsilon}{3\gamma}$ (resp. $\frac{\epsilon}{6\gamma}$) as long as $\epsilon \geq \frac{6\gamma}{1-3\gamma}$ (resp. $\epsilon \geq \frac{6\gamma}{1-12\gamma}$), where $\gamma := \max_P \mathrm{Disc}^t_{\Delta,P}(C^f)$, P runs over all t-wise partitions with $t = \Omega(k^{\frac{1}{d}})$.*

3.4 Good Matching in Graphs

Consider a graph G on vertex set V. For a partition $P = (V_1, V_2)$ of V, let G_P be the induced bipartite graph, i.e., $E(G_P) := \{(u, v) : (u, v) \in E(G), u \in V_1, v \in V_2\}$.

Definition 3.1. *Let G be a graph and $P = (V_1, V_2)$ be any partition of the vertex set. A matching M in the induced bipartite graph G_P is called good if for every pair of edges $(u_i, w_i), (u_j, w_j)$ in M, none of (u_i, u_j), (u_i, w_j), (w_i, w_j) and (u_j, w_i) are in $E(G)$. Further, we define*

$$\tau(G, P) = \max_{M \text{ is good w.r.t } P} |M|.$$

Lemma 3.1. *Let G be any graph with maximum degree d. Then for any partition P of the vertex set of G, we have $\tau(G, P) \geq \frac{|E(G_P)|}{2d^2}$.*

Proof. Consider the graph G_P and build a matching $M \subseteq E(G_P)$ by the following process, starting with an empty M.

1. If $E(G_P)$ is empty, return M. Otherwise, add any edge (u, v) in $E(G_P)$ to M.
2. Remove every edge currently in $E(G_P)$ that is incident to a vertex that is a neighbour of u or v (including themselves) in G.
3. Go back to 1.

Observe that after each completion of step 2, the number of edges newly added to M is 1 and the number of edges removed from $E(G_P)$ is at most $2d^2$ since degree of a vertex in G (and therefore in G_P as well) is at most d. Thus, size of M is at least $\frac{|E(G_P)|}{2d^2}$. It's simple to verify that the pruning done at step 2 ensures M forms a good matching. □

The following lemma gives a lower bound on the size of a good matching in a constant degree expander graph. The proof follows by a simple application of the Expander Mixing Lemma [HLW06, Lemma 2.5] and Lemma 3.1 . Similar arguments also appear in [Pit09, Lemma 4.2] and in [Hay11]. We provide a proof for the sake of completeness in the appendix.

Lemma 3.2. *[Pit09, Hay11] Let d be a constant and G be a d regular expander graph on k vertices with the second largest eigen value of the normalized adjacency matrix $\lesssim \frac{1}{\sqrt{d}}$ and $P = (V_1, V_2)$ be a nearly balanced partition of the vertex set V. Then,*

$$\tau(G, P) = \Omega_d(k).$$

The notation Ω_d hides a constant that depends on d.
The next lemma is about good matching in a full binary tree.

Lemma 3.3. *Given a full binary tree T on k vertices, there exists a number $t \approx \frac{2}{3}k$, such that for any partition $P = (V_1, V_2)$ of the vertex set with $|V_1| = t$,*

$$\tau(T, P) = \Omega(\log k).$$

Proof. In [HY16], it is shown that there exists a number $t \approx \frac{2}{3}k$ such that for any partition $P = (V_1, V_2)$ with $|V_1| = t$, the induced bipartite graph T_P has $\Omega(\log k)$ many edges. Since the maximum degree of T is 3, using Lemma 3.1 $\tau(T, P) = \Omega(\log k)$.

□

3.5 A Communication Problem

Much of what we're going to prove in this section, derives its intuition from the following communication problem in Yao's 2-party model. We state the problem below, although we point out to the reader that if one is just interested in verifying the monotone lower bounds we claim for the various arithmetic models, then it is not necessary to know this communication problem.

Problem 3.1. *Given a graph G with a vertex set V, where V is partitioned into V_1, V_2 and $|V| = k$, we consider the following communication problem in Yao's 2-party model: Alice (Bob) gets the assignment to variables corresponding to vertices of V_1 (V_2). Together they need to evaluate the Boolean function* IP_G *on their joint inputs.*

Having stated the problem, we prove below the key discrepancy upper bound that shows the above problem's communication complexity, w.r.t. any balanced partition, to be high. This consequence of our discrepancy bound may be of independent interest.

Lemma 3.4. *Consider a graph G on vertex set V such that $V = \{u_1, u_2, \ldots, u_k\}$. For a partition $P = (V_1, V_2)$ of V, let G_P be the induced bipartite graph. Under the uniform distribution $\mathcal{U}$ over $\{0, 1\}^{km}$, the following holds,*

$$\mathrm{Disc}_{\mathcal{U},P}(\mathrm{IP}_G) \leq 2^{-\Omega(\tau(G,P)\cdot m)}.$$

Proof. Let M_P be the good matching in G_P such that $\tau(G, P) = |M_P|$. For convenience let $|M_P| = t$. For any edge (u_i, u_j) in the matching M_P define $\overrightarrow{C_i} = \oplus_{u_r \in \mathrm{Nbd}(u_i)\setminus\{u_j\}} \overrightarrow{u_r}$ and $\overrightarrow{C_j} = \oplus_{u_\ell \in \mathrm{Nbd}(u_j)\setminus\{u_i\}} \overrightarrow{u_\ell}$. Here $\mathrm{Nbd}(u_i) = \{u_\ell : (u_i, u_\ell) \in E(G)\}$.
Then, $\mathrm{IP}_G(\overrightarrow{u_1}, \overrightarrow{u_2}, \ldots, \overrightarrow{u_k}) = (-1)^D$ where

$$\begin{aligned} D &= \Big(\sum_{(u_i,u_j)\in M_P} (\langle \overrightarrow{u_i}, \overrightarrow{u_j} \rangle + \langle \overrightarrow{C_i}, \overrightarrow{u_i} \rangle + \langle \overrightarrow{C_j}, \overrightarrow{u_j} \rangle \Big) + \sum_{\substack{u_q,u_r \notin V(M_P) \\ (u_q,u_r)\in E(G)}} \langle \overrightarrow{u_q}, \overrightarrow{u_r} \rangle \\ &= \Big(\sum_{(u_i,u_j)\in M_P} (\langle \overrightarrow{u_i} + \overrightarrow{C_j}, \overrightarrow{u_j} + \overrightarrow{C_i} \rangle + \langle \overrightarrow{C_i}, \overrightarrow{C_j} \rangle \Big) + \sum_{\substack{u_q,u_r \notin V(M_P) \\ (u_q,u_r)\in E(G)}} \langle \overrightarrow{u_q}, \overrightarrow{u_r} \rangle \qquad (2) \\ &= \Big(\sum_{(u_i,u_j)\in M_P} \langle \overrightarrow{u_i'}, \overrightarrow{u_j'} \rangle + c \Big) \end{aligned}$$

Here $\overrightarrow{u_i'} = \overrightarrow{u_i} + \overrightarrow{C_j}$ and $\overrightarrow{u_j'} = \overrightarrow{u_j} + \overrightarrow{C_i}$. Note that

$$c = \sum_{\substack{u_q,u_r \notin V(M_P) \\ (u_q,u_r)\in E(G)}} \langle \overrightarrow{u_q}, \overrightarrow{u_r} \rangle + \sum_{(u_i,u_j)\in M_P} \langle \overrightarrow{C_i}, \overrightarrow{C_j} \rangle.$$

For the partition $P = (V_1, V_2)$ consider an arbitrary rectangle $R \in \mathcal{R}(P)$ in $\{0,1\}^{|V_1| \cdot m} \times \{0,1\}^{|V_2| \cdot m}$. Here for the sake of simplicity we abuse the notation R and denote it as a characteristic function for the rectangle R. Let $\mathcal{U}$ be the uniform distribution over $\{0,1\}^{|V_1| \cdot m} \times \{0,1\}^{|V_2| \cdot m}$ and $\mathcal{U}_m$ be the uniform distribution over $\{0,1\}^m$.
Notice that

$$\begin{aligned} \mathrm{Disc}_{\mathcal{U},P}(\mathrm{IP}_G, R) &= \Big| \mathop{\mathbb{E}}_{\overrightarrow{u_i} \sim \mathcal{U}_m} \Big[\mathrm{IP}_G(\overrightarrow{u_1}, \overrightarrow{u_2}, \ldots, \overrightarrow{u_k}) \cdot R(\overrightarrow{u_1}, \overrightarrow{u_2}, \ldots, \overrightarrow{u_k}) \Big] \Big| \\ &= \Big| \mathop{\mathbb{E}}_{\overrightarrow{u_i} \sim \mathcal{U}_m : u_i \notin M_P} \Big[\mathop{\mathbb{E}}_{\overrightarrow{u_i} \sim \mathcal{U}_m : u_i \in M_P} \Big[\mathrm{IP}_G((\overrightarrow{u_1}, \ldots, \overrightarrow{u_k}) R(\overrightarrow{u_1}, \ldots, \overrightarrow{u_k}) \Big] \Big] \Big| \end{aligned} \tag{3}$$

Let us denote assignments to vertices not in M_P collectively by $\overrightarrow{w} \in \{0,1\}^{(k-2t)m}$, and assignments to vertices in M_P collectively by $\overrightarrow{u} \in \{0,1\}^{2tm}$. Thus, using (2), we can continue (3) as follows:

$$\mathrm{Disc}_{\mathcal{U},P}(\mathrm{IP}_G, R) \leq \mathop{\mathbb{E}}_{\overrightarrow{w}} \Big[\Big| \mathop{\mathbb{E}}_{\overrightarrow{u} \in \{0,1\}^{2tm}} \Big[\Big(-1 \Big)^{\sum_{(u_i,u_j) \in M_P} \langle \overrightarrow{u_i}, \overrightarrow{u_j} \rangle} R^{\overrightarrow{w}}(\overrightarrow{u}) \Big] \Big| \Big] \tag{4}$$

The inner expectation is $\mathrm{Disc}_{\mathcal{U}'}(\mathrm{IP})$ where $\mathcal{U}'$ is the uniform distribution over $\{0,1\}^{tm} \times \{0,1\}^{tm}$. The value of $\mathrm{Disc}_{\mathcal{U}'}(\mathrm{IP})$ is at most $2^{-\Omega(tm)}$ by Theorem 2.1.

□

We will now consider a multi-party communication problem in the number-in-hand (NIH) model from the point of view of best partition communication complexity. While 2-party communication problems are relevant for proving lower bounds against circuits and ABP's, it'd be helpful to use the multi-party model for both unrestricted depth formulas and bounded-depth formulas. As the information bottleneck for players grows with the number of players in the NIH model, this kind of communication problems capture the limitation of formulas, especially w.r.t. ABPs and circuits.

Problem 3.2. *Communication problem Path-IP*
Let there be t players $P_1, P_2, \ldots, P_t$. The problem is defined over a k vertex path Γ with a fixed vertex set $V = \{u_1, u_2, \ldots, u_k\}$ and its partition into sets $V_1, V_2, \ldots, V_t$. Each P_i gets the vertex set V_i. The input to this problem is a map $\pi_i : V_i \to \{0,1\}^m$ given to each P_i which specifies a m bit vector assignment to each vertex in V_i. Together the players want to decide if $\langle \overrightarrow{i_1}, \overrightarrow{i_2} \rangle + \langle \overrightarrow{i_2}, \overrightarrow{i_3} \rangle + \cdots + \langle \overrightarrow{i_{k-1}}, \overrightarrow{i_k} \rangle = 1 \pmod 2$. Here for every $j \in [k]$ $\overrightarrow{i_j} = \pi_\ell(u_j)$ when $u_j \in V_\ell$ (for $\ell \in [t]$).

Observe for this communication problem, rectangles are defined to be t-product sets, i.,e. R is called a rectangle if $R = R_1 \times R_2 \times \cdots \times R_t$ with each $R_i \subseteq \{0,1\}^{k_i m}$ and $k_i = |V_i|\ \forall i \in [t]$. Our goal is to show under uniform distribution $\mathcal{U}$ over $\{0,1\}^{km}$, the discrepancy of any rectangle R is at most $2^{-\Omega(tm)}$. To show this we first show the following lemma.

Lemma 3.5. *Consider a path graph Γ on k vertices. For every partition $\mathcal{P} = (\mathcal{P}_1, \ldots, \mathcal{P}_t)$ of the vertex set of Γ into $2 \leq t < k$ parts, there exists a partition $\widetilde{\mathcal{P}} = (\mathcal{P}_{\mathcal{A}}, \mathcal{P}_{\mathcal{B}})$ into two parts that is a coarsening of $\mathcal{P}$ such that $\tau(\Gamma, \widetilde{\mathcal{P}})$ is $\Omega(t)$.*

Proof. Consider a partition $\mathcal{P} = (\mathcal{P}_1, \ldots, \mathcal{P}_t)$ of the set of vertices $V(\Gamma)$. We create a random coarsening of it, $\widetilde{\mathcal{P}} = (\mathcal{P}_{\mathcal{A}}, \mathcal{P}_{\mathcal{B}})$, as follows: for every $i \in [t]$, toss an independent unbiased coin. If output is head, put the vertices of $\mathcal{P}_i$ in $\mathcal{P}_{\mathcal{A}}$ and otherwise put them in $\mathcal{P}_{\mathcal{B}}$. Since $\mathcal{P}$ partitioned $V(\Gamma)$ into exactly t parts, there are at least $t-1$ edges (u_i, u_{i+1}) of Γ such that u_i and u_{i+1} belong to different $\mathcal{P}_j$s. Denote by $\mathcal{E}$, the set of such edges. Now, for every edge $e = (u, v)$ in the path Γ define a random variable y_e such that

$$y_e = \begin{cases} 1 & \text{if } e \in \mathrm{cut}(\widetilde{\mathcal{P}}), \\ 0 & \text{otherwise.} \end{cases}$$

Here, cut($\widetilde{\mathcal{P}}$) is the set of edges $e = (u, v)$ such that $u \in \widetilde{\mathcal{P}_{\mathcal{A}}}$ and $v \in \widetilde{\mathcal{P}_{\mathcal{B}}}$ or vice-versa. Let the random variable $Y = \sum_e y_e$ be the size of cut($\widetilde{\mathcal{P}}$). Note,

$$\mathbb{E}[Y] = \sum_e \mathbb{E}[y_e] \geq \sum_{e \in \mathcal{E}} \mathbb{E}[y_e] = \frac{t-1}{2} = \Omega(t).$$

Thus, there exists a fixed coarser partition $\widetilde{\mathcal{P}} = (\mathcal{P}_{\mathcal{A}}, \mathcal{P}_{\mathcal{B}})$ of $V(\Gamma)$ such that the induced bipartite graph $\Gamma_{\widetilde{\mathcal{P}}}$ has $\Omega(t)$ many edges. Since Γ has maximum degree 2, by Lemma 3.1 $\tau(\Gamma, \widetilde{\mathcal{P}}) = \Omega(t)$.

□

Lemma 3.6. *Let $t \geq 2$ and let Γ be a path on k vertices. Under the uniform distribution $\mathcal{U}$ over $\{0,1\}^{km}$ for any t-wise partition $\mathcal{P}$, the following holds.*

$$\text{Disc}^t_{\mathcal{U},\mathcal{P}}(\text{IP}_\Gamma) \leq 2^{-\Omega(tm)}$$

Proof. Obtain a coarsening of the partition $\mathcal{P}$ prescribed by Lemma 3.5 to get $\widetilde{\mathcal{P}} = (\mathcal{P}_{\mathcal{A}}, \mathcal{P}_{\mathcal{B}})$. Consider any rectangle $R = R_1 \times \cdots \times R_t$ under partition $\mathcal{P}$. Let $A \subset [t]$ such that for every $i \in A$ the vertices of $\mathcal{P}_i$ goes to $\mathcal{P}_{\mathcal{A}}$. Similarly $B = [t] \setminus A$ and for every $j \in B$ vertices of $\mathcal{P}_j$ goes to $\mathcal{P}_{\mathcal{B}}$. Define, $R_A := \times_{i \in A} R_i$ and $R_B := \times_{j \in B} R_j$ and finally, $\widetilde{R} := R_A \times R_B$. Observe that $\widetilde{R}$ forms a two-dimensional rectangle w.r.t. $\widetilde{\mathcal{P}}$. It is simple to verify,

$$\text{Disc}^t_{\mathcal{U}}(R) = \text{Disc}_{\mathcal{U}}(\widetilde{R}).$$

We know using Lemma 3.4 $\text{Disc}_{\mathcal{U},\widetilde{\mathcal{P}}}(\widetilde{R}) \leq \text{Disc}_{\mathcal{U},\widetilde{\mathcal{P}}}(\text{IP}_\Gamma) \leq 2^{-\Omega(tm)}$. □

4 Monotone Circuit Lower Bound via Expander Graph-IP Polynomial

In this section we prove Theorem 1.1. For the sake of convenience we restate it.

Theorem 1.1. *Let G be a constant-degree expander graph on k vertices. Then, there exists a constant $c > 0$ such that any monotone circuit computing either of the polynomial $F_{k,n} \pm \epsilon \cdot f_{G,m}$ has size $2^{\Omega(km)}$ as long as $\epsilon \geq 2^{-ckm}$.*

Proof. Consider the Boolean function IP_G described in Subsection 3.1 where the underlying graph G is a constant degree expander graph on k vertices. Every vertex gets a m bit binary vector assignment. Let P be any nearly balanced partition of the vertex set V. We first show the following claim.

Claim 4.1. *Under the uniform distribution $\mathcal{U}$ over $\{0,1\}^{km}$, for every nearly balanced partition P on the vertex set V,*

$$\text{Disc}_{\mathcal{U},P}(\text{IP}_G) \leq 2^{-\Omega(km)}.$$

Proof. From Lemma 3.2 we know that for every nearly balanced partition P of V, $\tau(G, P) = \Omega(k)$. Using Lemma 3.4, under the uniform distribution $\mathcal{U}$ over $\{0,1\}^{km}$ we get that, $\text{Disc}_{\mathcal{U},P}(\text{IP}_G) \leq 2^{-\Omega(km)}$.

□

Now the proof follows from the first part of Theorem 3.1. Here the polynomial $f = f_{G,m}$ and C^f is the Boolean function IP_G. From Claim 4.1 it is clear that

$$\gamma = \max_P \text{Disc}_{\mathcal{U},P}(\text{IP}_G) \leq 2^{-\Omega(km)}.$$

The universal distribution Δ is the uniform distribution $\mathcal{U}$ over km bits. Let the value of $\gamma = 2^{-\gamma_0 km}$ for some constant $\gamma_0 > 0$. It is easy to verify that choosing $\epsilon \geq 2^{\frac{-\gamma_0 km}{10}}$ satisfies the condition $\epsilon \geq \frac{6\gamma}{1-3\gamma}$. Hence monotone circuit complexity of $F_{k,n} - \epsilon \cdot f_{G,m}$ is at least $\frac{\epsilon}{3\gamma}$ which is $2^{\Omega(km)}$. The proof for $F_{k,n} + \epsilon \cdot f_{G,m}$ is analogous. □

5 Separation between Monotone Circuits and Monotone ABPs via Tree-IP Polynomial

In this section we prove Theorem 1.2. For the sake of convenience we restate the theorem here.

Theorem 1.2. *Let T be the full binary tree on k vertices. Then, $f_{T,m}$ can be computed by monotone circuits of size $O(kn^3)$. On the other hand, there exists a constant $c > 0$ such that any monotone ABP computing either of the polynomial $F_{k,n} \pm \epsilon \cdot f_{T,m}$ has size $k^{\Omega(m)}$ as long as $\epsilon \geq k^{-cm}$.*

The proof of this theorem is divided in the following two subsections.

5.1 Upper Bound

First we show the upper bound. Let T_u be the sub-tree rooted at node u of T. Below we consider set-multilinear monomials κ such that $I(\kappa) = V(T_u)$. Further, we denote by $\overrightarrow{\kappa}[u]$ the m-bit binary assignment to $\overrightarrow{u}$ by $\overrightarrow{\kappa}$. For every node u in T and $\overrightarrow{a} \in \{0,1\}^m$ and $b \in \{0,1\}$, we define the following polynomials,

$$g_{u,b}^{\overrightarrow{a}} := \sum_{\substack{\mathrm{IP}_{T_u}(\overrightarrow{\kappa})=(-1)^b \\ \overrightarrow{\kappa}[u]=\overrightarrow{a}}} \kappa$$

and

$$g_{u,b} := \sum_{\overrightarrow{a}\in\{0,1\}^m} g_{u,b}^{\overrightarrow{a}}.$$

Thus, $g_{r,1}$ is the output polynomial $f_{T,m}$ where r is the root of T. By induction on the depth of T, we show that for each node u of T and $\overrightarrow{a} \in \{0,1\}^m$, the polynomials $g_{u,0}^{\overrightarrow{a}}, g_{u,1}^{\overrightarrow{a}}$ can be simultaneously computed by a circuit of size at most $O\Big(2^d 2^{3m}\Big)$.

For the base case $d = 1$. Let u be a node with two children v, w. For the purpose of this section, i, j, k are used for the integer values of $\overrightarrow{i}, \overrightarrow{j}, \overrightarrow{k} \in \{0,1\}^m$. Now the polynomials computed at node u are following,

$$g_{u,0}^{\overrightarrow{i}} = \sum_{\substack{\overrightarrow{j},\overrightarrow{k}\in\{0,1\}^m: \\ \langle\overrightarrow{i},\overrightarrow{j}\rangle+\langle\overrightarrow{i},\overrightarrow{k}\rangle=0 \pmod 2}} x_{u,i}x_{v,j}x_{w,k}$$

where for every $\overrightarrow{i}$ the polynomial $g_{u,0}^{\overrightarrow{i}}$ has 2^{2m} monomials. Hence we can compute the polynomials $g_{u,0}, g_{u,o}^{\overrightarrow{i}}$ by a monotone circuit of size at most 2^{3m}. Similarly we compute the polynomial $g_{u,1}$. So the total size of the circuit is 2^{3m+1} and the base case holds.

Consider a vertex u at depth d with children v, w. By inductive hypothesis, we have circuits C_v, C_w, each of size at most $O(2^{d-1}2^{3m})$ computing simultaneously the polynomials $g_{v,0}^{\overrightarrow{i}}, g_{v,1}^{\overrightarrow{j}}$ and $g_{w,0}^{\overrightarrow{i}}, g_{w,1}^{\overrightarrow{j}}$ respectively, for each $\overrightarrow{i}, \overrightarrow{j} \in \{0,1\}^m$.

Now we compute the polynomials $g_{u,0}^{\overrightarrow{i}}, g_{u,1}^{\overrightarrow{j}}$.

It is easy to observe that

$$g_{u,0}^{\overrightarrow{i}} = \sum_{\substack{\overrightarrow{j},\overrightarrow{k}\in\{0,1\}^m: \\ \langle\overrightarrow{i},\overrightarrow{j}\rangle+\langle\overrightarrow{i},\overrightarrow{k}\rangle=0 \pmod 2}} (x_{u,i}g_{v,0}^{\overrightarrow{j}}g_{w,0}^{\overrightarrow{k}}+x_{u,i}g_{v,1}^{\overrightarrow{j}}g_{w,1}^{\overrightarrow{k}})+ \sum_{\substack{\overrightarrow{j},\overrightarrow{k}\in\{0,1\}^m: \\ \langle\overrightarrow{i},\overrightarrow{j}\rangle+\langle\overrightarrow{i},\overrightarrow{k}\rangle=1 \pmod 2}} (x_{u,i}g_{v,0}^{\overrightarrow{j}}g_{w,1}^{\overrightarrow{k}}+x_{u,i}g_{v,1}^{\overrightarrow{j}}g_{w,0}^{\overrightarrow{k}}).$$

Now from the circuits C_v and C_w we appropriately reuse the subcircuits for $\{g_{v,0}^{\overrightarrow{j}}, g_{v,1}^{\overrightarrow{j}}, g_{w,0}^{\overrightarrow{k}}, g_{w,1}^{\overrightarrow{k}}\}$. The other case, that of computing $g_{u,1}^{\overrightarrow{j}}$, is completely analogous.

Hence, the final circuit size, denoted by $S(d)$, satisfies the following recurrence:

$$S(d) \leq 2S(d-1) + O(2^{3m}).$$

Solving the recursion we get $S(d)$ is at most $O\left(2^d 2^{3m}\right)$. The upper bound follows since $k = 2^{d+1} - 1$.

5.2 Lower Bound

Next we show the lower bound result. Consider the Boolean function IP_T described in Subsection 3.1, where T is a full binary tree on k vertices.

Claim 5.1. *There exists a number $t \approx \frac{2}{3}k$ such that for any partition $P = (V_1, V_2)$ of the vertex set $V(T)$ with $|V_1| = t$, under the uniform distribution $\mathcal{U}$ over $\{0,1\}^{km}$,*

$$\text{Disc}_{\mathcal{U},P}(\text{IP}_T) \leq k^{-\Omega(m)}.$$

Proof. Using Lemma 3.3 we know there exists a number $t \approx \frac{2}{3}k$ such that for any partition $P = (V_1, V_2)$ with $|V_1| = t$, $\tau(T, P) = \Omega(\log k)$. By Lemma 3.4, under the uniform distribution $\mathcal{U}$ over $\{0,1\}^{km}$, we know that $\text{Disc}_{\mathcal{U},P}(\text{IP}_T) \leq 2^{-\Omega(\tau(T,P)\cdot m)} = k^{-\Omega(m)}$. □

Now we apply the second part of Theorem 3.1 to prove the lower bound. Here the polynomial $f = f_{T,m}$ and C^f is the Boolean function IP_T. Using Claim 5.1,

$$\gamma = \max_P \text{Disc}_{\mathcal{U},P}(\text{IP}_T) \leq k^{-\Omega(m)}.$$

Let $\gamma = k^{-cm}$ for some constant $c > 0$ and Δ be the uniform distribution $\mathcal{U}$ over $\{0,1\}^{km}$. One can easily verify that the condition $\epsilon \geq \frac{6\gamma}{1-3\gamma}$ is satisfied by choosing $\epsilon \geq k^{\frac{-cm}{10}}$. Using Theorem 3.1, the monotone ABP complexity of $g = F_{k,n} - \epsilon \cdot f_{T,m}$ is at least $\frac{\epsilon}{3\gamma}$ which is $k^{\Omega(m)}$. The proof for $F_{k,n} + \epsilon \cdot f_{T,m}$ is analogous.

6 Monotone Separations via Path-IP Polynomial

In this section, we prove Theorem 1.3 and Theorem 1.4.

6.1 Separation between Monotone ABPs and Monotone Formulas

For convenience Theorem 1.3 is restated below.

Theorem 1.3. *Let Γ be a simple path on k vertices. Then, the polynomial $f_{\Gamma,m}$ can be computed by a monotone ABP of size $O(kn)$. On the other hand, there exists a constant $c > 0$ such that any monotone formula computing either of the polynomial $F_{k,n} \pm \epsilon \cdot f_{\Gamma,m}$ has size $k^{\Omega(m)}$ as long as $\epsilon \geq k^{-cm}$.*

The proof is divided in two parts.

Upper Bound

We first give the monotone ABP construction for the polynomial $f_{\Gamma,m}$. The ABP has $k+2$ layers $0, 1, \ldots, k, k+1$. Layer 0 and $k+1$ are the source and sink vertex respectively. Let the path Γ be $u_1 \to u_2 \to \cdots \to u_k$.

Layer 1 and k contains 2^m vertices labelled with $\{(u_1, i) | i \in [2^m]\}$ and $\{(u_k, j) | j \in [2^m]\}$ respectively. For every other layer $\ell \in [2, 3, \ldots, k-1]$, we have 2^{m+1} vertices labelled with $\{(u_\ell, j)_b \mid j \in [2^m],\ b \in \{0,1\}\}$. Next we describe the edge relations between consecutive layers.

- **Layer 0 to layer 1 :** The source node s in layer 0 is connected to every node in layer 1. The edge label of $s \to (u_1, i)$ is labeled by variable $x_{u_1,i}$.
- **Layer 1 to 2 :** A node (u_1, i) is connected to $(u_2, j)_b$ in layer 2 if and only if $\langle \overrightarrow{i}, \overrightarrow{j} \rangle = b \pmod 2$. The edge gets the label $x_{u_2,j}$. Here $\overrightarrow{i}, \overrightarrow{j}$ are the binary representation of i and j respectively.
- **Layer ℓ to $\ell + 1$ for $\ell \in [2, k-2]$:** A node $(u_\ell, j)_b$ in layer ℓ is connected to the node $(u_{\ell+1}, j')_{b'}$ if and only if $b + \langle \overrightarrow{j}, \overrightarrow{j'} \rangle = b' \pmod 2$. This edge label is $x_{u_{\ell+1},j'}$.
- **Layer $k-1$ to k :** A node $(u_{k-1}, i)_b$ is connected to the node (u_k, j) for if and only if $b + \langle \overrightarrow{i}, \overrightarrow{j} \rangle = 1 \pmod 2$.
- **Layer k to $k + 1$:** Every node in layer k is connected to sink vertex with edge label 1.

The size of the monotone ABP is $O(k2^m)$. Note that in the ABP construction each layer incrementally maintains the partial parity information. More precisely, a monomial $x_{u_1,i_i} x_{u_2,i_2} \cdots x_{u_k,i_k}$ is generated exactly once between the source and sink if and only if $\text{IP}_\Gamma(\overrightarrow{i_1}, \ldots, \overrightarrow{i_k}) = -1$. Hence the polynomial computed between the source and the sink is simply $f_{\Gamma,m}$.

Lower Bound

Consider the Boolean function IP_Γ described in Subsection 3.1, where Γ is a path on k vertices. We use the third part of Theorem 3.1 to prove the lower bound. Since there $\Omega(\log k)$-wise partitions are considered, we set $t = \Omega(\log k)$. By Lemma 3.6, under the uniform distribution $\mathcal{U}$ over $\{0,1\}^{km}$ for every t-wise partition P we know that $\gamma = \text{Disc}^t_{\mathcal{U},P}(\text{IP}_\Gamma) = 2^{-\Omega(tm)} = k^{-\Omega(m)}$.
Analogous to the case in Section 5, we now use the third part of Theorem 3.1 to show our lower bound against the size of monotone formulas computing $F_{k,n} - \epsilon \cdot f_{\Gamma,m}$ using the aboved discrepancy upper bound. As in Section 5, the lower bound follows by choosing $\epsilon \geq k^{-\Omega(m)}$ appropriately. The proof for $F_{k,n} + \epsilon \cdot f_{\Gamma,m}$ is analogous.

6.2 Separation between Monotone Constant Width ABPs and Monotone Constant Depth Formulas

Now we prove Theorem 1.4 which is restated below.

Theorem 1.4. *Let Γ be a simple path on k vertices. Then, the polynomial $f_{\Gamma,1}$ can be computed by a monotone width-4 ABP of size $O(k)$. On the other hand, there exists a constant $c > 0$ such that any monotone formula of product-depth d computing either of the polynomial $F_{k,2} \pm \epsilon \cdot f_{\Gamma,1}$ has size $2^{k^{\Omega(1/d)}}$ as long as $\epsilon \geq 2^{-ck^{1/d}}$.*

Upper Bound

Using the ABP construction given in Section 6.1 we get a width-4 ABP of size $O(k)$ for the polynomial $f_{\Gamma,1}$.

Lower Bound

To show the lower bound, consider the Boolean function IP_Γ described in Subsection 3.1 where Γ is a k vertex path and each vertex gets a 1 bit assignment (i.e, $m = 1$). Using the fourth part of Theorem

3.1, we set $t = \Omega(k^{\frac{1}{d}})$. By Lemma 3.6 under the uniform distribution $\mathcal{U}$ over $\{0,1\}^k$ for every t-wise partition P we know $\text{Disc}^t_{\mathcal{U},P}(\text{IP}_\Gamma) = 2^{-\Omega(k^{\frac{1}{d}})}$.
Now we use the fourth part of Theorem 3.1 to show monotone constant depth formula lower bound for polynomial $F_{k,n} - \epsilon \cdot f_{\Gamma,1}$ using the discrepancy upper bound. Similar to the earlier cases, the lower bound follows by choosing $\epsilon \geq 2^{-\Omega(k^{\frac{1}{d}})}$ appropriately.

References

[AJS09] Vikraman Arvind, Pushkar S. Joglekar, and Srikanth Srinivasan, *On lower bounds for constant-width arithmetic circuits*, 20th International Symposium on Algorithms and Computation (ISAAC), Springer-LNCS, 2009, pp. 637–646.

[CDGM22] Arkadev Chattopadhyay, Rajit Datta, Utsab Ghosal, and Partha Mukhopadhyay, *Monotone complexity of spanning tree polynomial re-visited*, 13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA (Mark Braverman, ed.), LIPIcs, vol. 215, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, pp. 39:1–39:21.

[CDM21] Arkadev Chattopadhyay, Rajit Datta, and Partha Mukhopadhyay, *Lower bounds for monotone arithmetic circuits via communication complexity*, STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021 (Samir Khuller and Virginia Vassilevska Williams, eds.), ACM, 2021, pp. 786–799.

[CELS18] Suryajith Chillara, Christian Engels, Nutan Limaye, and Srikanth Srinivasan, *A near-optimal depth-hierarchy theorem for small-depth multilinear circuits*, 59th IEEE Symposium on Foundations of Computer Science (FOCS), IEEE, 2018, pp. 934–945.

[CG88] Benny Chor and Oded Goldreich, *Unbiased bits from sources of weak randomness and probabilistic communication complexity*, SIAM J. Comput. **17** (1988), no. 2, 230–261.

[CKR20] Bruno Pasqualotto Cavalar, Mrinal Kumar, and Benjamin Rossman, *Monotone circuit lower bounds from robust sunflowers*, LATIN 2020: Theoretical Informatics - 14th Latin American Symposium, São Paulo, Brazil, January 5-8, 2021, Proceedings (Yoshiharu Kohayakawa and Flávio Keidi Miyazawa, eds.), Lecture Notes in Computer Science, vol. 12118, Springer, 2020, pp. 311–322.

[DMPY12] Zeev Dvir, Guillaume Malod, Sylvain Perifel, and Amir Yehudayoff, *Separating multilinear branching programs and formulas*, 44th ACM Symposium on Theory of Computing (STOC), ACM, 2012, pp. 615–624.

[GS12] S. B. Gashkov and I. S. Sergeev, *A method for deriving lower bounds for the complexity of monotone arithmetic circuits computing real polynomials*, Sbornik. Mathematics **203(10)** (2012).

[Hay11] Thomas P. Hayes, *Separating the k-party communication complexity hierarchy: An application of the zarankiewicz problem*, Discrete Mathematics & Theoretical Computer Science **13** (2011), no. 4, 15–22.

[HLW06] Shlomo Hoory, Nathan Linial, and Avi Wigderson, *Expander graphs and their applications*, Bull. Amer. Math. Soc. **43** (2006), 439–561.

[Hru20] Pavel Hrubeš, *On ϵ-sensitive monotone computations*, Computational Complexity **29** (2020), no. 2, 6.

[HY11] Pavel Hrubes and Amir Yehudayoff, *Homogeneous formulas and symmetric polynomials*, Comput. Complex. **20** (2011), no. 3, 559–578.

[HY16] ______, *On isoperimetric profiles and computational complexity*, 43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy (Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, eds.), LIPIcs, vol. 55, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016, pp. 89:1–89:12.

[HY21] Pavel Hrubeš and Amir Yehudayoff, *Shadows of newton polytopes*, 36th Computational Complexity Conference, CCC 2021, July 20-23, 2021, Toronto, Ontario, Canada (Virtual Conference) (Valentine Kabanets, ed.), LIPIcs, vol. 200, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, pp. 9:1–9:23.

[KN06] Eyal Kushilevitz and Noam Nisan, *Communication complexity*, Cambridge University Press, USA, 2006.

[KPR22] Balagopal Komarath, Anurag Pandey, and C.S. Rahul, *Graph homomorphism polynomials: Algorithms and complexity*, to appear in the 49th International Colloquium on Automata, Languages and Programming (ICALP), LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[Pit09] Toniann Pitassi, *Best-partition multiparty communication complexity*, Manuscript online at http://www.cs.toronto.edu/ toni/Courses/CommComplexity/Papers/bestpartition.ps, 2009, Course notes for Foundations of Communication Complexity, Fall 2009.

[RY09] Ran Raz and Amir Yehudayoff, *Lower bounds and separations for constant depth multilinear circuits*, Computational Complexity **18** (2009), no. 2, 171–207.

[RY11] ______, *Multilinear formulas, maximal-partition discrepancy and mixed-sources extractors*, J. Comput. Syst. Sci. **77** (2011), no. 1, 167–190.

[Sap21] Ramprasad Saptharishi, *A survey of lower bounds in arithmetic circuit complexity*, Manuscript online at https://github.com/dasarpmar/lowerbounds-survey/releases/download/v9.0.3/fancymain.pdf, 2021, A selection of lower bounds in arithmatic circuit complexity.

[Sri20] Srikanth Srinivasan, *Strongly exponential separation between monotone VP and monotone VNP*, ACM Trans. Comput. Theory **12** (2020), no. 4, 23:1–23:12.

[Val79] Leslie G. Valiant, *Completeness classes in algebra*, Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1979, Atlanta, Georgia, USA (Michael J. Fischer, Richard A. DeMillo, Nancy A. Lynch, Walter A. Burkhard, and Alfred V. Aho, eds.), ACM, 1979, pp. 249–261.

[Val86] ______, *Negation is powerless for boolean slice functions*, SIAM J. Comput. **15** (1986), no. 2, 531–535.

[Yeh19] Amir Yehudayoff, *Separating monotone VP and VNP*, Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019 (Moses Charikar and Edith Cohen, eds.), ACM, 2019, pp. 425–429.

A Appendix

The following theorems are re-statements of the corresponding theorems in [HY11]. There, they were stated for multilinear formulas. Here, we port the statements and their proofs from [HY11] to the set-multilinear setting needed for our work.

A.1 Structure Theorem for Monotone Set-Multilinear Formulas

Theorem 2.3. *[HY11, Lemma 4] Let f be a degree k set-multilinear polynomial computed by a (monotone) formula of size s. Then there exists (monotone) set-multilinear-*log*-product polynomials $g_1, g_2, \dots, g_{s'}$ such that $s' \leq s$,*

$$f = g_1 + g_2 + \cdots + g_{s'}.$$

Proof. Let Φ be the (monotone) set-multilinear formula computing polynomial f. W.l.o.g Φ is a syntactically set-multilinear formula. For any node w in the formula let Φ_w be the sub-formula rooted at node w. Further we denote by $\Phi(w \leftarrow \beta)$, the formula obtained after removing the sub-formula rooted at node w and relabelling it by β. For any node w, let the polynomial computed at node w be f_w and $I(f_w)$ be the set of rows of matrix M on which the polynomial f_w is defined. First note the following claim.

Claim A.1. *There exists a node w in the formula Φ such that $\frac{k}{3} \leq |I(f_w)| \leq \frac{2k}{3}$.*

Proof. W.l.o.g every node in Φ has in-degree 2. The polynomial computed at the root node r has $|I(f_r)| = k$. Now we traverse on the path towards the leaves by picking the child w among the children w, v when $|I(f_w)| \geq |I(f_v)|$. The traversing continues whenever the heavier child satisfies the condition $|I(f_w)| > \frac{2k}{3}$. We stop this process when we first encounter a node w such that $|I(f_w)| > \frac{2k}{3}$ but for its' children u, v $|I(f_u)|, |I(f_v)| \leq \frac{2k}{3}$. We will choose the heavier child here. I.e, we choose u if $|I(f_u)| \geq |I(f_v)|$ and the node u satisfies the property in the claim. □

We prove the theorem 2.3 by doing induction on s. For the base case when $s = 1$, the polynomial f is only one variable or constant. So, it trivially satisfies the conditions in the definition 2.2.
Using the claim A.1, let w be a node in the formula satisfying $\frac{k}{3} \leq |I(f_w)| \leq \frac{2k}{3}$ and size of $\Phi_w < s$. We can write the polynomial f in the following way,

$$f = g \cdot f_w + f'$$

where the polynomial f' is the set-multilinear polynomial computed by $\Phi(w \leftarrow 0)$. Clearly the formula size of Φ_w and $\Phi(w \leftarrow 0)$ is $< s$. Let they are s_w and $s(w \leftarrow 0)$. In particular $s_w + s(w \leftarrow 0) \leq s$. So we can use induction hypothesis on f_w and f'. That is, we can write $f_w = h_1 + \cdots + h_{s'_w}$ and $f' = h'_1 + \cdots + h'_{s'(w \leftarrow 0)}$ where for every i, h_i and h'_i are set-multilinear-log-product polynomials and $s'_w \leq s_w$, $s'(w \leftarrow 0) \leq s(w \leftarrow 0)$. Clearly $g \cdot h'_i$ is set-multilinear polynomial and the sets $(I(g), I(h'_i))$ is a partition of $[k]$ where $\frac{k}{3} \leq |I(g)| \leq \frac{2k}{3}$. Hence the set-multilinear-log-product decomposition of f is

$$f = gh_1 + gh_2 + \cdots + gh_{s'_w} + h'_1 + \cdots + h_{s'(w \leftarrow 0)}.$$

□

A.2 Structure Theorem for Monotone Set-Multilinear Constant Depth Formulas

Theorem 2.4. *[HY11, Lemma 9] Let f be a degree k set-multilinear polynomial computed by a (monotone) formula of size s and product depth d. Let $q > 1$ be a natural number such that $k > (2q)^d$. Then there exists (monotone) set-multilinear-$(q, k(2q)^{-d})$-form polynomials $g_1, g_2, \dots, g_{s'}$ such that $s' \leq s$,*

$$f = g_1 + g_2 + \cdots + g_{s'}.$$

Proof. Let Ψ be the size s product depth d (monotone) set-multilinear formula computing f. For any node v in the formula we denote the sub-formula rooted at node v by Ψ_v. Further we define the polynomial computed at node v by f_v and $I(f_v)$ be the set of rows in the matrix $M_{k\times n}$ on which f_v is defined. First note the following claim.

Claim A.2. *Let $t > 1$ be any positive real number such that $k > t^d$. Then there exists a product node v in Ψ such that $|I(f_v)| \geq k \cdot t^{-d+1}$ and for every children u of v, $|I(f_u)| < \frac{|I(f_v)|}{t}$. Moreover if $t = 2q$ for $q \in \mathbb{N}$ then f_v is in $(q, k(2q)^{-d})$-form.*

Proof. The proof is by induction on product depth d.
For the base case $d = 1$. Let v be any product gate with children $u_1, u_2, \dots u_p$. Clearly $|I(f_v)| = k$ and for every child, $|I(u_i)| \leq 1 < \frac{k}{t}$. So v is the required node in the claim.
Inductively assume the claim is true for every node at product depth $d' < d$. Let v be a product node at depth d with children $u_1, \dots, u_p$ and $|I(f_v)| = k$. If for every children u_i, $|I(f_{u_i})| < \frac{|I(f_u)|}{t} = \frac{k}{t}$, then v is our required node. Otherwise, let u_i be a child such that $|I(f_{u_i})| \geq \frac{k}{t}$. Product depth of $u_i < d$. So by induction hypothesis there is a node w in Ψ_{u_i} at product depth $d' < d$, such that $|I(f_w)| \geq |I(f_{u_i})| \cdot t^{-d'+1} \geq k \cdot t^{-d+1}$. Also for every children w_i of w, $|I(f_{w_i})| < \frac{|I(f_w)|}{t}$. So, w is the required node for the claim.
Let v be the node in the claim with children $u_1, \dots, u_p$ such that $|I(f_v)| = m \geq kt^{-d+1}$ and $|I(f_{u_i})| < \frac{m}{t}$. Then by appropriately grouping the polynomials $f_{u_1}, \dots, f_{u_p}$, it can be ensured that we get a new set of polynomials $\{g_1, g_2, \dots, g_{p'}\}$ such that $\frac{m}{t} \leq |I(g_j)| \leq \frac{2m}{t}$ for every $j \in [p']$. Hence f has $(\lfloor \frac{t}{2} \rfloor, \frac{m}{t})$-form. Putting $t = 2q$ and $m = kt^{-d+1}$ we get our desired form. □

We prove the theorem 2.4 by doing induction on the size, s. The base case is easy to verify. By Claim A.2 there is a node v in the formula with polynomial f_v is in $(q, k(2q)^{-d})$-form. Write

$$f = g \cdot f_v + f'$$

where f' is the polynomial computed by the formula Ψ after removing the sub-formula rooted at node v and relabelling it by the element 0. Clearly the sets $I(g)$ and $I(f_v)$ forms a partition of the rows of matrix. From Claim A.2 The polynomial f_v is in $(q, k(2q)^{-d})$ form. That is $f_v = f_1 \cdot f_2 \cdots f_q$ where each $|I(f_i)| \geq k(2q)^{-d}$. Clearly the polynomial $(gf_1) \cdots f_q$ is also in $(q, k(2q)^{-d})$-form. Hence the proof follows by doing induction on the sub-formula of size $< s$ computing f'.

□

B Good Matching in Constant Degree Expander Graphs

Lemma 3.2. *[Pit09, Hay11] Let d be a constant and G be a d regular expander graph on k vertices with the second largest eigen value of the normalized adjacency matrix $\lesssim \frac{1}{\sqrt{d}}$ and $P = (V_1, V_2)$ be a nearly balanced partition of the vertex set V. Then,*

$$\tau(G, P) = \Omega_d(k).$$

Proof. Take the partition $P = (V_1, V_2)$ such that $\frac{k}{3} \leq |V_1|, |V_2| \leq \frac{2k}{3}$ and construct the induced bipartite graph G_P. using Expander Mixing Lemma [HLW06, Lemma 2.5] we know $|E(G_P)| \geq \frac{d|V_1||V_2|}{k} - \lambda d\sqrt{|V_1||V_2|}$. Substituting the values of λ, $|V_1|$ and $|V_2|$ we get $|E(G_P)| = \Omega_d(k)$. Since it is a constant degree regular graph, using Lemma 3.1 we get $\tau(G, P) = \Omega_d(k)$.

□