

On the closures of monotone algebraic classes and variants of the Determinant

Prasad Chaugule¹ and Nutan Limaye²

¹ Indian Institute of Technology, Bombay, India

² IT University of Copenhagen, Denmark

Abstract. In this paper we prove the following two results.

- We show that for any $C \in \{\text{mVF}, \text{mVP}, \text{mVNP}\}$, $C = \overline{C}$. Here, mVF, mVP, and mVNP are monotone variants of VF, VP, and VNP, respectively. For an algebraic complexity class C , \overline{C} denotes the closure of C . For mVBP a similar result was shown in [4]. Here we extend their result by adapting their proof.
- We define polynomial families $\{\mathcal{P}(k)_n\}_{n \geq 0}$, such that $\{\mathcal{P}(0)_n\}_{n \geq 0}$ equals the Determinant polynomial. We show that $\{\mathcal{P}(k)_n\}_{n \geq 0}$ is VBP complete for $k = 1$ and it becomes VNP complete when $k \geq 2$. In particular, $\mathcal{P}(k)_n$ is $\text{Det}_n^{\neq k}(\mathbf{x})$, a polynomial obtained by summing over all signed cycle covers that avoid length k cycles. We show that $\text{Det}_n^{\neq 1}(\mathbf{x})$ is complete for VBP and $\text{Det}_n^{\neq k}(\mathbf{x})$ is complete for VNP for all $k \geq 2$ over any field \mathbb{F} .

1 Introduction

Valiant [17] initiated the study of the complexity of the algebraic computation. Given a polynomial, how efficiently can we compute it? In order to formalise the notion of efficiency, Valiant defined many natural models of computation. These include algebraic circuits, algebraic formulas, and algebraic branching programs.

An *algebraic circuit* is a directed acyclic graph. The nodes with in-degree zero are leaf nodes, which are labelled with variables $X = \{x_1, x_2, \dots, x_n\}$ or field constants. The other nodes are either $+$ or \times operators, which compute polynomials of their inputs. The $+$ node adds its inputs and \times node multiplies its inputs. There is a designated output node which has out-degree 0. The output of the circuit is the polynomial computed by this node. An *algebraic formula* is a circuit in which the underlying DAG is a tree. The size of the circuit/formula is the number of nodes in it.

An *algebraic branching program* (ABP) A is a layered DAG with two special nodes s and t called the source node and the sink node, respectively. The edges are labelled with linear forms $\sum_{i=1}^n c_i x_i + c_0$, where $c_i \in \mathbb{F}$. For every directed path ρ from s to t , we associate a polynomial P_ρ which is formed by multiplying all the labels along the edges in path ρ . The polynomial computed by the ABP A is equal to $\sum_\rho P_\rho$ where the sum is over all $s - t$ paths in A . The size of the algebraic branching program A is the number of nodes in it. We assume that the length of every path from s to t is the same.

A polynomial family f_n is called a p -bounded family if both the number of variables and the degree of f_n are polynomially bounded in n . The class of p -bounded families computed by polynomial sized circuits is called VP. Similarly, the class of p -bounded polynomial families computed by formulas of polynomial size are called VF. Finally, the class of p -bounded polynomial families computed by ABPs of polynomial size is called VBP.

Closures of monotone complexity classes. Instead of studying the complexity of the exact computation of any polynomial, we could also study the algebraic approximation of a polynomial. A polynomial Q over $\mathbb{F}(\varepsilon)[X]$ is said to be an algebraic approximation of a polynomial P over $\mathbb{F}[X]$ if there exist polynomials Q_1, Q_2, \dots in $\mathbb{F}[X]$ such that $Q \equiv P + \sum_{i \geq 1} \varepsilon^i Q_i$. It is possible that approximating a polynomial is computationally less expensive than computing it exactly. More formally, let \mathcal{C} be any class of polynomials and $\overline{\mathcal{C}}$, the *closure of \mathcal{C}* , be the polynomials approximated by the polynomials in \mathcal{C} . One can always ask the *strict containment* question: Is $\mathcal{C} \subsetneq \overline{\mathcal{C}}$? One of the important and well-known questions is the VP vs. $\overline{\text{VP}}$ question. Here $\overline{\text{VP}}$ stands for a class of p -bounded polynomials that are approximated by polynomial sized circuits³. From the definition, it is clear that $\text{VP} \subseteq \overline{\text{VP}}$, but whether the containment is strict or not is an open question, which has a rich and long history ([3], [2], [13], [8]).

In some cases, strict containment holds. Let $\sum^{[k]} \prod \Sigma$ denotes the class of polynomials which can be represented as the sum of product of affine forms, where k denotes the fan-in of the top-most addition gate in the circuit. Kumar [10] showed that the class of polynomials $\sum^{[3]} \prod \Sigma$ is strictly contained in $\overline{\sum^{[3]} \prod \Sigma}$. Let VP_k denotes the class of algebraic branching programs of width k . Bringmann, Ikenmeyer and Zuiddam [5] looked at the class of polynomials computed by *width-2 algebraic branching programs* and showed that for any constant k , $\text{VP}_k \subseteq \overline{\text{VP}_2}$. This along with a result of Allender and Wang [1] shows that $\text{VP}_2 \subsetneq \overline{\text{VP}_2}$.

In this work, however, we consider monotone algebraic complexity classes and for each class \mathcal{C} that we consider, we show that $\mathcal{C} = \overline{\mathcal{C}}$. That is, we answer the strict containment question negatively for monotone algebraic complexity classes. Recently, Bläser, Ikenmeyer, Mahajan, Pandey and Saurabh [4] showed that mVBP equals its closure, where mVBP is a class of polynomials computable by *monotone algebraic branching program* (formal definition in Section 2). We extend their result to other monotone algebraic complexity classes. We show that $\text{mVP} = \overline{\text{mVP}}$, that is, anything that can be approximated by monotone circuits of polynomial size (computing polynomials of polynomial degree) can also be computed by them. We also show that mVF , a class of polynomials computed by monotone algebraic *formulas*, is equal to its closure and that mVNP , the monotone analogue of VNP ⁴ (formally defined in Section 3) equals its closure.

³ Formally, $\overline{\text{VP}}$ is defined using *topological approximations*. However, for reasonably well-behaved fields \mathbb{F} , the two notions of approximation are equivalent. We will focus on algebraic approximation in this note.

⁴ Let $f_n(x_1, x_2, \dots, x_{k(n)})$ be a p -bounded polynomial family. f_n is said to be in VNP if there exists a family $g_n \in \text{VP}$ such that $f_n =$

Overall, our results imply that approximation does not add extra power to the monotone algebraic models of computation. Our proof is elementary and a simple generalisation of the proof of $\text{mVBP} = \overline{\text{mVBP}}$ from [4].

Variants of the Determinant. We recall the combinatorial definition of the determinant polynomial. (See for instance [11]).

Definition 1. Let G_n be a directed complete graph on n vertices, where the edge label of edge (i, j) is $x_{i,j}$ for $i, j \in [n]$. Let \mathcal{C} denotes the set of cycle covers⁵ of G_n . We define $\text{Det}_n(\mathbf{X}) = \sum_{C \in \mathcal{C}} s(C) \cdot m_C$, where $s(C) = (-1)^{n+t}$, where t is the number of cycles in the cycle cover C and m_C is the monomial formed by multiplying the labels on all the edges in C .

We define a variant $\text{Det}_n^{\neq k}(\mathbf{X})$ of the determinant, which sums over all signed cycle covers that avoid length k cycles. Formally,

Definition 2. Let k be a fixed constant. Let G_n be a directed complete graph on n vertices, where the edge label of edge (i, j) is $x_{i,j}$ for $i, j \in [n]$. Let $\mathcal{C}^{\neq k}$ denote all those cycle covers of G_n in which all the cycles have length $\neq k$. We define

$$\text{Det}_n^{\neq k}(\mathbf{X}) = \sum_{C \in \mathcal{C}^{\neq k}} s(C) \cdot m_C$$

where $s(C) = (-1)^{n+t}$, where t is the number of cycles in the cycle cover C and m_C is the monomial formed by multiplying the labels on all the edges in C .

We prove the following Lemma.

Lemma 1. $\text{Det}_n^{\neq 1}(\mathbf{X})$ is complete⁶ for VBP and $\text{Det}_n^{\neq k}(\mathbf{X})$ is complete for VNP for all $k \geq 2$ over any field \mathbb{F} .

Using Valiants criteria ([17]), it is easy to observe that for $k \geq 0$, $\text{Det}_n^{\neq k}(\mathbf{X})$ is in VNP. We prove VNP hardness in Section 4.

The family $\text{Det}_n^{\neq k}(\mathbf{X})$ can be viewed as a parametrized family with k as the parameter such that $\text{Det}_n^{\neq 1}(\mathbf{X})$ is VBP-complete and $\text{Det}_n^{\neq k}(\mathbf{X})$ for all $k \geq 2$ is VNP-complete over all fields. In previous works by Durand et.al. [7] and Chaugule et. al. [6], many polynomials were proposed which characterised VP. These were defined using homomorphism polynomials. It would be interesting to come up with a parametrized polynomial family which characterizes not just VBP and VNP, but all natural complexity classes, namely VF, VBP, VP, and VNP.

$\sum_{y_1, y_2, \dots, y_{k'(n)} \in \{0,1\}} g_n(x_1, x_2, \dots, x_{k(n)}, y_1, y_2, \dots, y_{k'(n)})$, where $k'(n)$ is polynomially bounded in n .

⁵ A cycle cover of a directed graph H is a set of vertex disjoint cycles which are subgraphs of graph H and contains all the vertices of H .

⁶ A family $f_n(X)$ is said to be a p -projection of $g_n(Y)$ (denoted as $f_n(X) \leq_p g_n(Y)$) if there exists a polynomially bounded function $t : \mathbb{N} \rightarrow \mathbb{N}$ such that $f_n(X)$ can be computed from $g_{t(n)}(Y)$ by setting its variables to one of the variables of $f_n(X)$ or to field constants. A p -bounded family f_n is said to be hard for a complexity class \mathcal{C} if for any $g_n \in \mathcal{C}$, $g_n \leq_p f_n$. A p -bounded family f_n is complete for a class \mathcal{C} if it is in \mathcal{C} and is hard for \mathcal{C} .

2 Preliminaries

2.1 Border complexity

Analogous to algebraic complexity classes VP, VF, VBP, we define the border complexity variants of these classes. We denote the corresponding complexity classes as $\overline{\text{VP}}$, $\overline{\text{VF}}$, and $\overline{\text{VBP}}$ respectively.

Definition 3. A polynomial family $h_n \in \mathbb{F}[X]$ is said to be in class $\overline{\text{VP}}$ ($\overline{\text{VF}}$, or $\overline{\text{VBP}}$ respectively) if and only if there exist polynomials $h_{n,i} \in \mathbb{F}[X]$ and a function $t : \mathbb{N} \rightarrow \mathbb{N}$ such that the polynomial $g_n(x) = h_n(x) + \varepsilon h_{n,1}(x) + \varepsilon^2 h_{n,2}(x) + \dots + \varepsilon^{t(n)} h_{n,t(n)}(x)$ can be computed by an algebraic circuit $C \in \mathbb{F}(\varepsilon)[X]$ (algebraic formula $F \in \mathbb{F}(\varepsilon)[X]$ or an algebraic branching program $A \in \mathbb{F}(\varepsilon)[X]$, respectively) of size polynomial in n , where ε is the new indeterminate.

2.2 Monotone complexity

Since, we are in the monotone setting, hereafter, we will work only with the field of real numbers or the field of rational numbers, for the sake of the presentation, we set, $\mathbb{F} = \mathbb{R}$.

An algebraic circuit (algebraic formula and algebraic branching programs, respectively) is said to be a monotone algebraic circuit (monotone algebraic formula and monotone algebraic branching programs, respectively) if all the input constants (from \mathbb{R}) are non-negative.

Definition 4. The class of p -bounded families computed by polynomial sized monotone circuits (monotone formula/ABPs) is called $m\text{VP}$ ($m\text{VF}$, $m\text{VBP}$, respectively).

2.3 Monotone border complexity

Analogous to our complexity classes $m\text{VP}$, $m\text{VF}$, $m\text{VBP}$, we now define the border complexity variants of these classes. We denote the corresponding complexity classes as $\overline{m\text{VP}}$, $\overline{m\text{VF}}$, $\overline{m\text{VBP}}$ respectively.

Let $\mathbb{R}_+[\varepsilon, \varepsilon^{-1}]$ denote the ring of Laurent polynomials that are non-negative for all sufficiently small $\varepsilon > 0$. In other words, the monotonicity condition means that for each $\alpha \in \mathbb{R}_+[\varepsilon, \varepsilon^{-1}]$, there is a $\beta > 0$ such that for all $\varepsilon \in (0, \beta)$, $\alpha(\varepsilon) \geq 0$.

Definition 5. A polynomial family $h_n \in \mathbb{F}[X]$ is said to be in class $\overline{m\text{VP}}$ ($\overline{m\text{VF}}$, $\overline{m\text{VBP}}$) if and only if there exist polynomials $h_{n,i} \in \mathbb{F}[X]$ such that the polynomial $g_n(x) = h_n(x) + \varepsilon h_{n,1}(x) + \varepsilon^2 h_{n,2}(x) + \dots + \varepsilon^t h_{n,t}(x)$ can be computed by an algebraic circuit C (algebraic formula, branching program, resp.) of size polynomial in n where the input are either labelled by variables or polynomials from $\mathbb{R}_+[\varepsilon, \varepsilon^{-1}]$.

The definition of $\overline{m\text{VNP}}$ is slightly involved. We define it in Section 3.3.

3 Closures of monotone algebraic classes

The first result we prove in this section is about algebraic closure of mVP.

Theorem 1. $mVP = \overline{mVP}$

In [4], Bläser, Ikenmeyer, Mahajan, Pandey, and Saurabh proved that “ $mVBP = \overline{mVBP}$ ”. Our proof has an analogous proof outline. It consists of the following three steps.

1. Let \mathcal{C}_n be a monotone algebraic circuit of size $s = \text{poly}(n)$ computing the polynomial $Q = \varepsilon^0 f_0 + \sum_{i=1}^k \varepsilon^i f_i$. We convert the circuit \mathcal{C}_n into another monotone circuit \mathcal{C}'_n of size $\text{poly}(s)$, such that the circuit \mathcal{C}'_n computes the polynomial $\varepsilon^t Q$ for some positive integer t , and no input gate in \mathcal{C}'_n has a label with negative exponent of ε as its input (Section 3.1).
2. We give a general procedure to convert a monotone circuit computing the polynomial $\varepsilon^t Q$ to $\varepsilon^{t-1} Q$ (with edge labels). Moreover, no input gate or edge in the converted circuit has a negative exponent of ε as its label. (Section 3.2)
3. We repeatedly apply step 2 and construct a monotone circuit which computes the polynomial $\varepsilon^0 Q$. By finally substituting ε to 0, the result follows.

Although, the overall proof idea is very similar to the proof idea from [4], the details in Step 1 and Step 2 are slightly different. Informally, we use the structural properties of the universal circuit to get Step 1, whereas we exploit the monotonicity property of the given circuit to achieve Step 2.

3.1 Step 1 of Theorem 1

Before going into the details of the proof this step of Theorem 1, we recall the following definitions.

Definition 6 ([14], [16]). *A family of circuits $\{\mathcal{U}_n\}_{n \in \mathbb{N}}$ is called a universal circuit family if for every polynomial $f_n(x_1, x_2, \dots, x_n)$ of degree $d(n)$ which is computed by a circuit of size $s(n)$, there exists another circuit Ψ which computes f_n such that the underlying Directed Acyclic Graph (DAG) of Ψ is the same as that of \mathcal{U}_m for some $m \in \text{poly}(n, s, d)$. We assume that both $s(n)$ and $d(n)$ are polynomially bounded in n .*

A universal circuit \mathcal{C}_n is said to be in a normal form if it satisfies the following properties.

- *The circuit \mathcal{C}_n is a semi-bounded circuit, i.e., the indegree of \times gate is 2, whereas the indegree of $+$ gates is unbounded.*
- *The circuit \mathcal{C}_n is a multiplicatively disjoint circuit.*
- *The circuit \mathcal{C}_n is a layered circuit where layers are alternately labelled with $+$ and \times gates. We assume that the root node is a \times gate. Also, the distance between the root node to every leaf node of circuit \mathcal{C}_n is the same.*
- *The degree of the circuit \mathcal{C}_n is n . The depth of the circuit $\mathcal{C}_n = 2c \lceil \log n \rceil$, for some constant c .*

- The number of variables $v(n)$ and the size $s(n)$ of the circuit C_n are both polynomially bounded in n .

It is known that a universal circuit can be assumed to be in a normal form.

We recall the definition of a parse tree of an algebraic circuit.

Definition 7 ([12]). *The set of parse trees of a circuit C , $\mathcal{T}(C)$ is defined inductively on the size of circuit C*

- If the size of C is 1, then the circuit C is its own parse tree.
- If the circuit size is greater than 1, then the output gate is either a multiplication gate or an addition gate
 1. Let the output gate t be an addition gate. Then the parse trees of circuit C are formed by taking a parse tree of any one of its children, say t' along with the edge (t', t) .
 2. Let the output gate t be a multiplication gate. Let t_1 and t_2 be the children gates of t . Let C_{t_1} and C_{t_2} be the subcircuits rooted at gates t_1 and t_2 in circuit C . The parse trees of C are formed by taking a node disjoint copy of a parse tree of subcircuit C_{t_1} and a parse tree of subcircuit C_{t_2} along with the edges (t_1, t) and (t_2, t) .

Remark 1. Given an algebraic circuit C computing a polynomial f . Let $\mathcal{T}(C)$ denote the set of all parse trees of circuit C . Let $\text{mon}(T)$ be the monomial associated with $T \in \mathcal{T}(C)$, where $\text{mon}(T)$ is equal to the product of the labels of the leaves of T . It is known that f is equal to $\sum_{T \in \mathcal{T}(C)} \text{mon}(T)$ (see [12]).

Remark 2. Note that the shape of every parse tree T of the universal circuit (in normal form) C_n is the same. Moreover, the number of leaf nodes in any parse tree T of C_n is equal to $2^{c \lceil \log n \rceil} = n^c$.

The following Lemma gives the details for the first step.

Lemma 2. *Consider a monotone algebraic circuit C_n of size $s = \text{poly}(n)$ computing a polynomial $Q = \varepsilon^0 f_0 + \sum_{i=1}^k \varepsilon^i f_i$ (that is, circuit C_n computes the polynomial f_0 in the border sense), then there exists another monotone circuit C'_n (in normal form) of size $\text{poly}(s)$ such that the circuit C'_n computes the polynomial $\varepsilon^t Q$, for some positive integer t . Moreover, no input gate in circuit C'_n has a label with a negative exponent of ε as its input. And the underlying DAG structure of C'_n is the same as that of C_n .*

Proof. Consider a circuit C_n which computes the polynomial Q and say it has size s . We know that there exists a universal circuit in normal form \mathcal{U}_m , where $m = \text{poly}(s)$ such that the circuit \mathcal{U}_m also computes Q . Moreover, it is known that the monotonicity of circuit C_n can be preserved in circuit \mathcal{U}_m . Let j be the largest negative exponent in any of the input gates of C_n . We multiply the label of every input gate of circuit \mathcal{U}_m by ε^j . We call this new circuit C'_n . Since, the circuit C_n is in the normal form, the shape of every parse tree of circuit C_n is the same and the number of leaf nodes in any parse tree is equal to n^c (see

Remark 1 and Observation 2). Therefore, the polynomial computed by circuit \mathcal{C}'_n is equal to a scaled version of polynomial computed by circuit \mathcal{C}_n , that is, the polynomial computed by \mathcal{C}'_n is equal to $\varepsilon^t Q$, where $t = j \times n^c$. We did not change the underlying graph structure of \mathcal{C}_n to obtain \mathcal{C}'_n . \square

3.2 Step 2 of Theorem 1

We now present the details regarding the second step of our main proof.

Here we will use algebraic circuits with edge labels. A circuit C with edge labels over a field \mathbb{F} and variable set $X = \{x_1, x_2, \dots, x_n\}$ is exactly similar to the algebraic circuit we have been considering, except for the edge function $w : E \rightarrow X \cup \mathbb{F}$. Here E denotes the edge set of C . The polynomial computed at any input gate u is equal to the label of u . Let P_u denote the polynomial computed at the node u in the circuit C . The polynomial computed at any computation gate u with operation op , with u_ℓ and u_r as its children nodes is equal to the $(w((u, u_\ell)) \times P_{u_\ell}) \text{op} (w((u, u_r)) \times P_{u_r})$. The polynomial computed by the circuit C is the polynomial computed by the output gate of circuit C . The size of the algebraic circuit C is the number of nodes in it.

Definition 8. *A node u in any monotone circuit over $\mathbb{R}_+[\varepsilon^{-1}, \varepsilon]$ is called a good node, if the polynomial f_u computed at node u is divisible by ε .*

Lemma 3. *Consider any monotone circuit \mathcal{D}_n of size s with edge label function $w : E \rightarrow \{\varepsilon^i | i \in \mathbb{Z}_{\geq 0}\}$, where E is the set of edges of \mathcal{D}_n . Suppose it computes a polynomial $\varepsilon^b Q$ for some $b \geq 1$ where $Q = f_0 + \sum_{i=1}^b \varepsilon^i f_i$ and the circuit \mathcal{D}_n satisfies the following properties:*

- No input gate or any edge in \mathcal{D}_n has a label with a negative exponent of ε .
- The underlying graph structure of \mathcal{D}_n is the same as the graph structure of a universal circuit as in Definition 6.

Then there exists a circuit \mathcal{D}'_n of size s such that \mathcal{D}'_n computes $\varepsilon^{b-1} Q$. Moreover, no input gate (or an edge label) in \mathcal{D}'_n is labelled with a negative exponent of ε . Also, the underlying graph structure of the circuit \mathcal{D}'_n is same as that of \mathcal{D}_n , upto the relabelling of the input gates and edge labels of the circuit \mathcal{D}_n .

Let \mathcal{G} denote the set of all good nodes of circuit \mathcal{D}_n . In order to prove Lemma 3, we need to describe the circuit \mathcal{D}'_n . The circuit \mathcal{D}'_n is exactly similar to the circuit \mathcal{D}_n , with labels of some input nodes $\mathcal{L}' \subseteq \mathcal{G}$ scaled by $1/\varepsilon$, and an updated edge function w' . Moreover, the new function w' also satisfies the property that no edge has a label with a negative exponent of ε .

Instead of \mathcal{L}' , we construct a larger set \mathcal{S} such that $\mathcal{L}' \subseteq \mathcal{S} \subseteq \mathcal{G}$ and the root node $r \in \mathcal{S}$. Let f_u denote the polynomial computed at node u in circuit \mathcal{D}_n . Let \widehat{f}_u denote the polynomial computed at node u in circuit \mathcal{D}'_n . It suffices to prove the following lemma.

Lemma 4. *In the circuit \mathcal{D}'_n , for every node $u \in \mathcal{S}$, $\widehat{f}_u = \frac{1}{\varepsilon} f_u$.*

Since, the root node $r \in \mathcal{S}$, the proof of Lemma 3 immediately follows (see Section 3.2 for proof details).

Identifying the set \mathcal{S} and the edge label function \hat{w} The main idea is to mark some of the good nodes of circuit \mathcal{D}_n which will form the set \mathcal{S} . First of all, we will mark the root node of circuit \mathcal{D}_n . The main goal is to scale the polynomial computed at the marked root node by $1/\epsilon$. We will propagate this effect from the root node layer to the layer immediately after it and so on till we reach to the layer of the leaf nodes. In this process, we may also change the labels of the edges thereby changing the old w function to the new updated w' function. Upon reaching the last layer, we scale all the marked leaf nodes of \mathcal{D}_n by $1/\epsilon$ (by relabelling). We now discuss the procedure in detail.

Let us number the layers of \mathcal{D}_n from 1 to m such that the layer containing the root node is numbered as 1 and so on till the layer containing the leaf nodes is numbered m .

Marking the root node: We mark the root node of circuit \mathcal{D}_n . Note that by our assumption of circuit \mathcal{D}_n , the root node is always a good node.

Marking the nodes at layer $i + 1$: Given the marking of nodes upto layer numbered i , we give a procedure to mark the nodes in layer numbered $i + 1$. We break this case into two parts depending on whether $i + 1$ is even or odd.

Case 1: $i + 1$ is odd and $i + 1 \leq m$: We know that the layer numbered i consists of the $+$ gates.. Let u_1, u_2, \dots, u_t be the marked nodes of layer i . Inductively, we know that u_1, u_2, \dots, u_t are all good nodes. For each $j \in [t]$, let $u_{j,1}, u_{j,2}, \dots, u_{j,f(j)}$ be children of node u_j . Note that, for all $j \in [t]$, the nodes $u_{j,1}, u_{j,2}, \dots, u_{j,f(j)}$ are in layer numbered $i + 1$. For each $k \in [f(j)]$, the condition of the monotonicity of \mathcal{D}_n (along with the property that u_j is a good node and is an addition gate) guarantees that

- 1a. either $u_{j,k}$ is a good node,
- 1b. or the edge $(u_{j,k}, u_j)$ is labelled by ϵ^β for some $\beta \geq 1$.

We now describe a procedure which essentially scales the polynomials computed via $u_{j,1}, u_{j,2}, \dots, u_{j,f(j)}$ by $1/\epsilon$, which in turn implies that the the polynomial computed at u_j is also scaled by $1/\epsilon$. This scaling is done either immediately by reducing the exponent of ϵ along the edge between the child node and the $+$ gate or is postponed to the next layer by marking the child node. We now state it in Algorithm 1 below.

Case 2: $i + 1$ is even and $i + 1 \leq m$: We know that the layer numbered i consists of \times gates. Let u_1, u_2, \dots, u_t be the marked nodes of layer i . Inductively, we know that u_1, u_2, \dots, u_t are all good nodes. For each $j \in [t]$, let $u_{j,\ell}$ and $u_{j,r}$ be the left child and the right child of node u_j , respectively. Note that, for all $j \in [t]$, the nodes $u_{j,\ell}$ and $u_{j,r}$ are in layer numbered $i + 1$. Since the circuit \mathcal{D}_n is multiplicatively disjoint, for each $j \in [t]$, nodes $u_{j,\ell}$ and $u_{j,r}$ are always distinct. The property of monotonicity of \mathcal{D}_n (along with the property that u_j is a good node and is a multiplication gate) guarantees that one of the following two cases holds.

- 2a. If both $u_{j,\ell}$ and $u_{j,r}$ are not good nodes then there exist at least one edge from $\{(u_{j,\ell}, u_j), (u_{j,r}, u_j)\}$ which is labelled by ϵ^γ for some $\gamma \geq 1$. Let us call that edge e' .


```

for  $j = 1$  to  $t$  do
  for  $k = 1$  upto  $f(j)$  do
    if the node  $u_{j,k}$  is already marked then
      | we do nothing;
    end
    else if Case 1a holds then
      | we mark the node  $u_{j,k}$ ;
    end
    else if Case 1b holds then
      | we relabel the edge  $(u_{j,k}, u_j)$  initially labelled by  $\varepsilon^\beta$  to  $\varepsilon^{\beta-1}$ ;
    end
  end
end

```

Algorithm 1: Procedure to mark nodes on layer $i + 1$ when $i + 1$ is odd.

- 2b. Either $u_{j,\ell}$ or $u_{j,r}$ is a good node. (If both are good we arbitrarily fix one and call it as z)

The main idea of the following procedure is to scale either the polynomial fed via $u_{j,\ell}$ or the polynomial fed via $u_{j,r}$ by $1/\varepsilon$, which in turn scales the polynomial computed at u_j by $1/\varepsilon$. This scaling is done either immediately by reducing/increasing the exponent of ε appropriately along the edge between the child node and the \times gate and/or is postponed to the next layer by marking the child node. We now state the procedure in Algorithm 2 below.

We know that \mathcal{S} consists of all the marked nodes of \mathcal{D}_n . Let $\ell_1, \ell_2, \dots, \ell_{t'}$ be the marked leaf nodes of \mathcal{D}_n . By the construction, we know that these nodes are good nodes. Therefore, their labels have the exponent of ε at least 1. We reduce the exponent of ε in each of these marked leaves by 1. We call the new circuit \mathcal{D}'_n where \widehat{w} denotes the edge label function of \mathcal{D}'_n .

Proof of Lemma 4 We prove by using induction on the layer numbers of the nodes in \mathcal{S} in the circuit \mathcal{D}'_n .

Base Case: Consider the layer m of \mathcal{D}'_n . This layer consists of the leaf nodes of circuit \mathcal{D}'_n . By our construction, it is easy to note that the base case holds.

Inductive case: We assume that the inductive hypothesis holds for layer numbered $i + 1$ and show that it holds for layer i . We break this case into two parts depending on whether i is even or odd.

i is even: We know that the layer numbered i consists of the addition gates. Let u_1, u_2, \dots, u_t be the marked nodes of layer i . For each $j \in [t]$, let $u_{j,1}, u_{j,2}, \dots, u_{j,f(j)}$ be children of node u_j . For each $j \in [t]$ we know that $f_{u_j} = \sum_{k=1}^{f(j)} w(u_{j,k}, u_j) \times f_{u_{j,k}}$ and therefore, $\widehat{f}_{u_j} = \sum_{k=1}^{f(j)} \widehat{w}(u_{j,k}, u_j) \times \widehat{f}_{u_{j,k}}$. By our construction, for all $k \in [f(j)]$, either $u_{j,k}$ is a marked node or the edge label function \widehat{w} updates the weight on edge $(u_{j,k}, u_j)$ as follows : $\widehat{w}(u_{j,k}, u_j) = \frac{1}{\varepsilon} w(u_{j,k}, u_j)$. Let $\mathcal{M} \subseteq [f(j)]$ is such that for each $a \in \mathcal{M}$, $u_{j,a}$ is

```

Set  $\mathcal{E} = \phi$ 
for  $j = 1$  to  $t$  do
  if exactly one of the nodes from  $u_{j,\ell}$  and  $u_{j,r}$  is marked then
    | we do nothing
  end
  else if both the nodes  $u_{j,\ell}$  and  $u_{j,r}$  are marked then
    | we arbitrarily pick one of the edges from  $\{(u_{j,\ell}, u_j), (u_{j,r}, u_j)\}$ . Let us
    | call that edge  $e$ . we increase the exponent of  $\varepsilon$  on edge  $e$  by 1.
  end
  else if Case 2a holds then
    | we reduce the exponent of  $\varepsilon$  on edge  $e'$  by 1, that is, we relabel the
    | edge  $e'$  by  $\varepsilon^{\gamma-1}$ .
  end
  else if Case 2b holds then
    | we mark the node  $z$ . Let  $\mathcal{K} \subseteq \mathcal{E}$  such that each node  $a \in \mathcal{K}$  is a parent
    | of  $z$ .
    for each  $a \in \mathcal{K}$  do
      | the exponent of the  $\varepsilon$  on edge  $(z, a)$  is increased by 1.
    end
  end
  Update  $\mathcal{E} = \mathcal{E} \cup \{u_j\}$ .
end

```

Algorithm 2: Procedure to mark nodes on layer $i + 1$ when $i + 1$ is even.

a marked node. Inductively, we know that $\widehat{f}_{u_{j,a}} = \frac{1}{\varepsilon} f_{u_{j,a}}$. Therefore,

$$\widehat{f}_{u_j} = \sum_{k=1}^{f(j)} \widehat{w}(u_{j,k}, u_j) \times \widehat{f}_{u_{j,k}} \quad (1)$$

$$= \sum_{a \in \mathcal{M}} \widehat{w}(u_{j,a}, u_j) \times \widehat{f}_{u_{j,a}} + \sum_{b \in \overline{\mathcal{M}}} \widehat{w}(u_{j,b}, u_j) \times \widehat{f}_{u_{j,b}} \quad (2)$$

$$= \sum_{a \in \mathcal{M}} w(u_{j,a}, u_a) \times \frac{1}{\varepsilon} f_{u_{j,a}} + \sum_{b \in \overline{\mathcal{M}}} \frac{1}{\varepsilon} w(u_{j,b}, u_j) \times f_{u_{j,b}} = \frac{1}{\varepsilon} f_{u_j} \quad (3)$$

i is odd : This case is similar to the previous one. We skip the other details of this case. \square

By repeatedly applying the step 2, we obtain a circuit \mathcal{C}'_n which computes the polynomial $\varepsilon^0 Q$. This finishes the proof of Theorem 1.

3.3 mVNP and mVF

We state the definition of the monotone variant of VNP.

Definition 9. Let $f_n(x_1, x_2, \dots, x_{k(n)})$ be a p -bounded polynomial family. We say that f_n is in mVNP if there exists a family $g_n \in \text{mVP}$ such that $f_n = \sum_{y_1, y_2, \dots, y_{k'(n)} \in \{0,1\}} g_n(x_1, x_2, \dots, x_k, y_1, y_2, \dots, y_{k'(n)})$, where $k'(n)$ is polynomially bounded in n .

We now recall the definition of $\overline{\text{VNP}}$ from [9].

Definition 10 ([9]). Let $f_n(x_1, x_2, \dots, x_{k(n)})$ be a p -bounded polynomial family. We say that f_n is in $\overline{\text{VNP}}$ if there exists a p -bounded family $\hat{f}_n \in \text{VNP}$ over the field $\mathbb{F}(\varepsilon)$ and a function $t : \mathbb{N} \rightarrow \mathbb{N}$ such that $\hat{f}_n(x) = f_n(x) + \varepsilon f_{n,1}(x) + \varepsilon f_{n,2}(x) + \dots + \varepsilon^{t(n)} f_{n,t(n)}(x)$ for some $f_{n,1}, f_{n,2}, \dots, f_{n,t(n)}$ defined over \mathbb{F} .

We can now define the class $\overline{\text{mVNP}}$.

Definition 11. Let $f_n(x_1, x_2, \dots, x_{k(n)})$ be a p -bounded polynomial family. We say that f_n is in $\overline{\text{mVNP}}$ if there exist a p -bounded family $\hat{f}_n \in \text{mVNP}$ over $\mathbb{R}_+[\varepsilon, \varepsilon^{-1}]$ and a function $t : \mathbb{N} \rightarrow \mathbb{N}$ such that $\hat{f}_n(x) = f_n(x) + \varepsilon f_{n,1}(x) + \varepsilon f_{n,2}(x) + \dots + \varepsilon^{t(n)} f_{n,t(n)}(x)$ for some $f_{n,1}, f_{n,2}, \dots, f_{n,t(n)}$ defined over \mathbb{R} .

Lemma 5. $\text{mVNP} = \overline{\text{mVNP}}$

Proof. Let $f_n(x_1, x_2, \dots, x_{k(n)}) \in \overline{\text{mVNP}}$. By the definition of $\overline{\text{mVNP}}$, we know that there exists a p -family $\hat{f}_n \in \text{mVNP}$ (over the field $\mathbb{F}(\varepsilon)$) such that $\hat{f}_n(x) = f_n(x) + \varepsilon f_{n,1}(x) + \varepsilon f_{n,2}(x) + \dots + \varepsilon^{t(n)} f_{n,t(n)}(x)$ for some $f_{n,1}, f_{n,2}, \dots, f_{n,t(n)} \in \mathbb{R}[X]$. We know that $\hat{f}_n = \sum_{y_1, y_2, \dots, y_{k'(n)} \in \{0,1\}} g_n(X, Y)$, where $k'(n)$ is polynomially bounded in n and $g_n(X, Y) \in \text{mVP}$ (over the field $\mathbb{F}(\varepsilon)$). Let $g_n(X, a_1, a_2, \dots, a_{k'(n)})$ denote the evaluation of the polynomial $g_n(X, Y)$ at $y_1 = a_1, y_2 = a_2, \dots, y_k = a_{k'(n)}$. Since the polynomial \hat{f}_n is monotone and that it converges, each of the summand in $\hat{f}_n = \sum_{y_1, y_2, \dots, y_{k'(n)} \in \{0,1\}} g_n(X, Y)$ must also necessarily converge. That is, for any boolean setting of variables $y_1 = a_2, y_2 = a_2, \dots, y_{k'(n)} = a_{k'(n)}$, the polynomial $g_n(X, a_1, a_2, \dots, a_{k'(n)})$ must converge. Therefore, $g_n(X, Y)$ must also converge. By Theorem 1, there exists a circuit C_{g_n} which computes $g_n(X, Y)$ such that there are no negative exponents of ε in any of its labels. Let $g_n(X, Y)_{\varepsilon=0}$ denote the polynomial obtained after substituting $\varepsilon = 0$ in $g_n(X, Y)$. We know that $f_n(x) = \sum_{y_1, y_2, \dots, y_{k'(n)} \in \{0,1\}} g_n(X, Y)_{\varepsilon=0}$. By the definition of mVNP , the result follows. \square

We now state the final lemma of this section.

Lemma 6. $\text{mVF} = \overline{\text{mVF}}$

We use the proof of $\text{mVBP} = \overline{\text{mVBP}}$ from [4] to prove Lemma 6. We skip the details of this proof.

4 Complexity of $\text{Det}_n^{\neq k}(X)$

It is not very hard to see that $\text{Det}_n^{\neq k}(X)$ is VBP complete for $k = 1$. The proof idea is similar to the proof which shows that $\text{Det}_n(X)$ is VBP complete. In the rest of this section, we prove that $\text{Det}_n^{\neq k}(X)$ is VNP complete for all $k \geq 2$.

Proof idea. Before going into the details of the complexity of $\text{Det}_n^{\neq k}(\mathbf{X})$ ($k \geq 2$), we will look at the various ingredients required to show that $\text{Det}_n^{\neq k}(\mathbf{X})$ ($k \geq 2$) is VNP-complete. In Section 4.1, we discuss the gadget construction \mathcal{H}_k and its properties. In Section 4.2, we discuss the rosette construction and its properties. For any $f_n(X) \in \text{VNP}$, we use the gadgets \mathcal{H}_k and rosettes to construct the graph T_m ($m = \text{poly}(n)$) such that $\text{Det}_m^{\neq k}(\mathbf{X})$ defined over T_m computes $f_n(X)$.

4.1 Gadget Construction \mathcal{H}_k and its properties

Gadget \mathcal{H}_k . For every $k > 1$, we construct a *partial iff gadget* \mathcal{H}_k .

- Let $V(\mathcal{H}_k) = \{a_i | 1 \leq i \leq 2k - 1\}$.
- Let $E(\mathcal{H}_k) = \{(a_t, a_{t+1}) | 1 \leq t \leq 2k - 2\} \cup \{(a_1, a_{k+1}), (a_{k+1}, a_2), (a_{2k-1}, a_1)\} \cup \{(a_m, a_m) | 3 \leq m \leq 2k - 1\}$. We call the edges in $E(\mathcal{H}_k)$ as the gadget edges.
- The weights of (a_1, a_2) and the self-loop on a_{k+1} are -1 . All the other edge weights are 1 .

Consider a directed graph G with two distinct edges (u, v) and (u', v') . We say that the gadget \mathcal{H}_k is placed between the edges (u, v) and (u', v') , if we delete both the edges (u, v) and (u', v') in G and we add the following directed edges, $\{(u, a_1), (a_1, v), (u', a_2), (a_2, v')\}$. We set $w(u, a_1) = w(u, v)$ and $w(u', a_2) = w(u', v')$. We set $w(a_1, v) = w(a_2, v') = 1$.

Before getting into the details of the properties of the gadget \mathcal{H}_k , we first define *perceived sign* and *perceived monomial* of a cycle cover.

Let G be a directed graph with edge labelling function $\Phi : E(G) \rightarrow X \cup \mathbb{F}$, where $X = \{x_i | 1 \leq i \leq n\}$. Let $\mathcal{C} = \{C_1, \dots, C_k\}$ be a cycle cover of G . Let $\{e_1, \dots, e_t\}$ be the edges in \mathcal{C} . The sign of the cycle cover, which we denote as $\text{sign}(\mathcal{C})$ is defined as $(-1)^k$. The monomial of \mathcal{C} , denoted as $\text{mon}(\mathcal{C})$, is defined as the product of the labels of edges in \mathcal{C} , i.e. $\prod_{i=1}^t \Phi(e_i)$.

Note that the coefficient of this monomial can be either positive or negative based on the number of negatively signed edge labels in \mathcal{C} . We denote this sign by $s(m_{\mathcal{C}})$. Based on this, we define *perceived sign* $\hat{s}(\mathcal{C})$ and *perceived monomial* $\hat{m}_{\mathcal{C}}$ as follows.

Let $\hat{s}(\mathcal{C})$ be $(-1)^k \cdot s(m_{\mathcal{C}})$. And let $\hat{m}_{\mathcal{C}} = m_{\mathcal{C}} \cdot s(m_{\mathcal{C}})$. Note that

$$s(\mathcal{C}) \cdot m_{\mathcal{C}} = \hat{s}(\mathcal{C}) \cdot \hat{m}_{\mathcal{C}}. \quad (4)$$

Properties of \mathcal{H}_k . We now state some important properties of \mathcal{H}_k . Consider a directed graph $G = (V, E)$ with two distinct edges (u, v) and (u', v') . Let G' be the graph obtained by placing the gadget \mathcal{H}_k between the edges (u, v) and (u', v') in G . Let $\mathcal{C}^{\neq k}$ be a cycle cover (without any cycle of length k) with *perceived sign* \hat{s} which uses either both or none of the edges (u, v) and (u', v') in G , then there exists another cycle cover $\mathcal{C}'^{\neq k}$ with *perceived sign* \hat{s}' in graph G' such that the *perceived monomial* associated with both $\mathcal{C}^{\neq k}$ and $\mathcal{C}'^{\neq k}$ are same and $\hat{s} = \hat{s}'$. We also show that if there exists a cycle cover $\mathcal{C}^{\neq k}$ in G such that it uses exactly one of the edges from (u, v) and (u', v') , then such a cycle cover does not survive in G' .

1. If $\mathcal{C}^{\neq k}$ does not use any of the edges (u, v) and (u', v') then the $\mathcal{C}'^{\neq k}$ consists of all the cycles in $\mathcal{C}^{\neq k}$ and the gadget vertices are covered within themselves by a single cycle $(a_1, a_2, \dots, a_{2k-1}, a_1)$. The total number of cycles in cycle cover $\mathcal{C}'^{\neq k}$ is one more than the number of cycles in cycle cover $\mathcal{C}^{\neq k}$, but since the weight of edge (a_1, a_2) is -1 , the *perceived sign* of cycle cover $\mathcal{C}'^{\neq k}$ is same as the *perceived sign* of cycle cover $\mathcal{C}^{\neq k}$.
2. If $\mathcal{C}^{\neq k}$ uses both the edges (u, v) and (u', v') then $\mathcal{C}'^{\neq k}$ has all the cycles in $\mathcal{C}^{\neq k}$ except that the edges (u, v) and (u', v') are now replaced by directed path (u, a_1, v) and (u', a_2, v') respectively. The vertices $a_3, a_4, \dots, a_{2k-1}$ are covered by self-loops (of weight 1) in $\mathcal{C}'^{\neq k}$. Since the total number of cycles covered by self-loops in the gadget is always odd and the self-loop on the vertex a_{k+1} has weight -1 , the *perceived sign* is preserved.
3. If $\mathcal{C}^{\neq k}$ uses the edge (u, v) but not the edge (u', v') then there is only one way to cover the vertices a_2, a_3, \dots, a_{k+1} with a cycle $(a_2, a_3, \dots, a_{k+1}, a_2)$ of length k , which is not a valid cycle cover.
4. If $\mathcal{C}^{\neq k}$ uses the edge (u', v') but not the edge (u, v) then there is only one way to cover the vertices $a_{k+1}, a_{k+2}, \dots, a_{2k-1}$ with a cycle $(a_{k+1}, a_{k+2}, \dots, a_{2k-1}, a_{k+1})$ of length k , which is not a valid cycle cover.
5. Note that after placing the gadget \mathcal{H}_k , there could be new cycle covers that arise in the graph G' which were not present in graph G . There are only two possible cases. Out of these two cases, in one of the cases, the contribution of all such cycle covers in the overall sum is 0 whereas in the other case, the contribution is not 0. We explain both the cases in detail below.
 - (a) Let $\mathcal{C}^{\neq k}$ be a cycle cover consisting of a cycle, say \mathcal{C}_1 starting with vertex u followed by the edge (u, a_1) and then the edge path $\mathcal{P}_1 = (a_1, a_2)$ followed by an edge (a_2, v') . Since there are two paths from a_1 and a_2 (within the gadget vertices), there exists another cycle cover $\mathcal{C}'^{\neq k}$ consisting of a cycle \mathcal{C}'_1 starting with vertex u followed by the edge (u, a_1) and then the path $\mathcal{P}'_1 = (a_1 - a_{k+1} - a_2)$ followed by an edge (a_2, v') such that the perceived monomials of both $\mathcal{C}^{\neq k}$ and $\mathcal{C}'^{\neq k}$ are same. Moreover, since the number of cycles in cycle cover $\mathcal{C}'^{\neq k}$ is one less than the number of cycles in cycle cover $\mathcal{C}^{\neq k}$, their perceived signs are different. In other words, there exists a bijection Ψ between the set of cycle covers using \mathcal{P}_1 in one of its cycles and the set of cycle covers using \mathcal{P}'_1 in one of its cycles such that for any \mathcal{C} , $\Psi(\mathcal{C})$ and \mathcal{C} have same perceived monomials but with opposite perceived signs. Therefore, the contribution of such cycle covers to the overall sum is 0.
 - (b) Let $\mathcal{C}^{\neq k}$ be a cycle cover consisting of a cycle, say \mathcal{C}_1 with a path \mathcal{P}_1 in it starting with vertex u' followed by the edge (u', a_2) and then a path from vertex a_2 to vertex a_1 (within the gadget \mathcal{H}_k) followed by an edge (a_1, v) . There are no cancellations possible in this case and therefore, such cycle covers survives.

Remark 3. Unlike the Valiant's iff gadget, in this gadget, we do not guarantee that the contribution of cycle covers (which may arise due to the placing of this gadget) is 0 (see Point 5(b) above). Therefore, we call \mathcal{H}_k *partial iff gadget*. For

the proof to work, we exploit the property of the graph on which these gadgets are placed such that the contribution of such new cycle covers is 0.

4.2 Rosette Construction $R(\ell, \mathcal{I})$

In this section, we describe the rosette construction $R(\ell, \mathcal{I})$ for every $\ell > k$ and $\mathcal{I} \subseteq [\ell]$. The construction of $R(\ell, \mathcal{I})$ is very similar to the construction of $R(\ell)$ as stated in [15], except for some modifications to incorporate the restriction about the length of the cycle in the cycle cover. Formally, we consider a directed cycle \mathcal{C} of length ℓ with vertex set $|V(\mathcal{C})| = \{u_1, u_2, \dots, u_\ell\}$ and $E(\mathcal{C}) = \{e_i = (u_i, u_{i+1}) \mid 1 \leq i \leq \ell - 1\} \cup \{e_\ell = (u_\ell, u_1)\}$. We call the edges in $|E(\mathcal{C})|$ as *connector edges* and the vertices in $|V(\mathcal{C})|$ as *connector vertices*. Consider a set $\mathcal{S}(\mathcal{I}) = \{e_i \mid i \in \mathcal{I}\}$. For every edge (u_i, u_j) in $\mathcal{S}(\mathcal{I})$, we add a new vertex $t_{i,j}$ and add edges $(u_i, t_{i,j})$ and $(t_{i,j}, u_j)$. We call the edges in set $\mathcal{S}(\mathcal{I})$ as *participating edges*. We add self-loops on all the vertices of our graph. We arbitrarily pick one of the connector vertices and set the weight of the self-loop on it to 1, whereas the weights of all the other self-loops are set to -1 . The weights of all the edges in rosette $R(\ell, \mathcal{I})$ (except the self-loops) are set to 1. This completes the construction of $R(\ell, \mathcal{I})$. It is easy to observe that every $R(\ell, \mathcal{I})$ contains a unique longest cycle. We denote such a cycle by \mathcal{Z} . The rosette $R(\ell, \mathcal{I})$ satisfies the following four properties.

1. There is no cycle cover in $R(\ell, \mathcal{I})$ that contains a cycle of length k .
2. For any subset $\phi \neq X \subseteq \mathcal{S}(\mathcal{I})$, there exists exactly one cycle cover of $R(\ell, \mathcal{I})$ which, among the participating edges, contains exactly the edges in X . Such a cycle cover always contains a single cycle which is not a loop and all other remaining vertices are covered with self loops.
3. There are only two cycle covers of $R(\ell, \mathcal{I})$ which contain no participating edges. The first cycle cover consist of only self-loops on each of the vertices in $R(\ell, \mathcal{I})$. The other cycle cover consists of the unique longest cycle \mathcal{Z} .
4. There are no other cycle covers in $R(\ell, \mathcal{I})$.

Note that for any cycle cover C of rosette $R(\ell, \mathcal{I})$, the *perceived sign* is -1 and the *perceived monomial* is 1.

4.3 Construction of graph T_m from $f_n(X) \in \text{VNP}$

Let $f_n(X) \in \text{VNP}$. From the definition of VNP we know that $f_n(X) = \sum_{y_1, y_2, \dots, y_{p(n)} \in \{0,1\}} g'_n(X, Y)$, where $p : \mathbb{N} \rightarrow \mathbb{N}$ is polynomially bounded function in n and $g'_n(X, Y)$ is in VP. Moreover, for any $f_n(X) \in \text{VNP}$, $f_n(X) = \sum_{y_1, y_2, \dots, y_{p(n)} \in \{0,1\}} g_n(X, Y)$, where $g_n(X, Y)$ is in VF [12].

1. Since, $g_n(X, Y) \in \text{VF}$ and $\text{VF} \subseteq \text{VBP}$, there exists an algebraic branching program of size $s = \text{poly}(n)$ to compute $g_n(X, Y)$, say \mathcal{B}_n . Let s_0 and t_0 be the source and sink of \mathcal{B}_n , respectively. Without loss of generality assume that the length of the longest path from s_0 to t_0 in \mathcal{B}_n is at least k . We add

- a special vertex α and add directed edges from α to s_0 and from t to α . We set the weights of both the edges (α, s_0) and (t, α) to 1. We add self-loops on all the vertices in our constructed graph except α . We set the weight of all self-loops to be 1. We call this graph $\hat{\mathcal{B}}_n$.
2. We know that $Y = \{y_i | 1 \leq i \leq p(n)\}$. Let $occ(i)$ denote the total number of edges in $\hat{\mathcal{B}}_n$ which are labelled with variable y_i . Let $Y_1 = \{y_i | y_i \in Y, occ(i) > k\}$ and $Y_2 = \{y_i | y_i \in Y, occ(i) \leq k\}$. Clearly, $Y = Y_1 \cup Y_2$. For every $y_i \in Y_1$, we consider a rosette $R(occ(i), \mathcal{I} = [occ(i)])$. Similarly, for every $y_j \in Y_2$, we consider a rosette $R(occ(j) + k, \mathcal{I} \subset [occ(j)])$, where the set $|\mathcal{S}(\mathcal{I})| = occ(j)$. We call the graph constructed so far as the *partial graph* denoted by \tilde{T}_m .
 3. For any $y_i \in Y_1$, let $c_{y_i,1}, c_{y_i,2}, \dots, c_{y_i,occ(i)}$ be the connector edges in $R(occ(i), \mathcal{I} = [occ(i)])$. Let $e_{y_i,1}, e_{y_i,2}, \dots, e_{y_i,occ(i)}$ be the distinct edges in $\hat{\mathcal{B}}_n$ which are labelled with variable y_i . We place \mathcal{H}_k between the edges $c_{y_i,t}$ and $e_{y_i,t}$ for all $1 \leq t \leq occ(i)$.
 4. For any $y_j \in Y_2$, let $c_{y_j,1}, c_{y_j,2}, \dots, c_{y_j,occ(j)}, \dots, c_{y_j,occ(j)+k}$ be the connector edges in $R(occ(j) + k, \mathcal{I} \subset [occ(j) + k])$. Without loss of generality, let us assume that $\mathcal{S}(\mathcal{I}) = \{c_{y_j,1}, c_{y_j,2}, \dots, c_{y_j,occ(j)}\}$. Let $e_{y_j,1}, e_{y_j,2}, \dots, e_{y_j,occ(j)}$ be the distinct edges in $\hat{\mathcal{B}}_n$ which are labelled with variable y_j . We place “partial iff gadget” \mathcal{H}_k between the edges $c_{y_j,t}$ and $e_{y_j,t}$ for all $1 \leq t \leq occ(j)$.

This completes the construction of graph T_m . It is easy to note that $m = \text{poly}(n)$.

4.4 Proof of VNP hardness of $\text{Det}_n^{\neq k}(X)$ for $k \geq 2$

Let $f_n(X) \in \text{VNP}$. We have $f_n(X) = \sum_{y_1, y_2, \dots, y_{p(n)} \in \{0,1\}} g_n(X, Y)$, where $p(\cdot)$ is polynomially bounded function in n and $g_n(X, Y)$ is in VF. We know that \mathcal{B}_n computes $g_n(X, Y)$. Let $\mathcal{P} = \{\mathcal{P}_i | 1 \leq i \leq \mu\}$ be the set of all $s_0 - t_0$ paths in \mathcal{B}_n . Let m_i be the monomial associated with \mathcal{P}_i formed by multiplying all the edges of \mathcal{B}_n participating in path \mathcal{P}_i . Let $m'_j := m_j(a_1, \dots, a_{p(n)})$ be an evaluation of m_j for a specific Boolean setting of Y variables.

Recall that T_m is the graph constructed from $f_n(X)$ as stated in Section 4.3. We will prove that for every non-zero m'_j there is a unique cycle cover \mathcal{C} in T_m such that the product of the perceived sign of \mathcal{C} and the perceived monomial of \mathcal{C} is exactly equal to m'_j . Moreover, there are no other cycle covers in T_m . This will prove the VNP hardness. We skip the other details of the proof in the conference version.

Acknowledgement. We would like to thank Shourya Pandey, Radu Curticapean, and the anonymous reviewers who gave useful comments on the earlier draft of the paper.

References

1. Eric Allender and Fengming Wang. On the power of algebraic branching programs of width two. *computational complexity*, 25(1):217–253, 2016.
2. Dario Bini. Relations between exact and approximate bilinear algorithms. applications. *Calcolo*, 17(1):87–97, 1980.
3. Dario Andrea Bini, M Capovani, Francesco Romani, and G Lotti. $O(n^{2.7799})$ complexity for $n \times n$ approximate matrix multiplication. 1979.
4. Markus Bläser, Christian Ikenmeyer, Meena Mahajan, Anurag Pandey, and Nitin Saurabh. Algebraic branching programs, border complexity, and tangent spaces. In *35th Computational Complexity Conference, CCC 2020*, volume 169 of *LIPICs*, pages 21:1–21:24, 2020.
5. Karl Bringmann, Christian Ikenmeyer, and Jeroen Zuiddam. On algebraic branching programs of small width. *Journal of the ACM (JACM)*, 65(5):1–29, 2018.
6. Prasad Chaugule, Nutan Limaye, and Aditya Varre. Variants of homomorphism polynomials complete for algebraic complexity classes. *ACM Trans. Comput. Theory*, 13(4):21:1–21:26, 2021.
7. Arnaud Durand, Meena Mahajan, Guillaume Malod, Nicolas de Rugy-Altherre, and Nitin Saurabh. Homomorphism polynomials complete for VP. *Chicago Journal of Theoretical Computer Science*, 2016, 2016.
8. Joshua A. Grochow, Ketan D. Mulmuley, and Youming Qiao. Boundaries of VP and VNP. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP*, volume 55 of *LIPICs*, pages 34:1–34:14, 2016.
9. Christian Ikenmeyer and Abhiroop Sanyal. A note on VNP-completeness and border complexity. *Inf. Process. Lett.*, 176:106243, 2022.
10. Mrinal Kumar. On the power of border of depth-3 arithmetic circuits. *ACM Transactions on Computation Theory (TOCT)*, 12(1):1–8, 2020.
11. Meena Mahajan and V. Vinay. Determinant: Combinatorics, algorithms, and complexity. *Chicago Journal of Theoretical Computer Science*, 1997.
12. Guillaume Malod and Natacha Portier. Characterizing Valiant’s algebraic complexity classes. In *MFCS*, pages 704–716. Springer, 2006.
13. Ketan D Mulmuley and Milind Sohoni. Geometric complexity theory I: An approach to the P vs. NP and related problems. *SIAM Journal on Computing*, 31(2):496–526, 2001.
14. Ran Raz. Elusive functions and lower bounds for arithmetic circuits. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 711–720. ACM, 2008.
15. Nitin Saurabh. Algebraic models of computation. *MS Thesis*, 2012.
16. Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends® in Theoretical Computer Science*, 5(3–4):207–388, 2010.
17. L. G. Valiant. Completeness classes in algebra. In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing, STOC ’79*, pages 249–261, 1979.