The Algebraic Cost of a Boolean Sum

Ian Orzel
Department of Computer Science, The University of Copenhagen, Denmark

Srikanth Srinivasan
Department of Computer Science, The University of Copenhagen, Denmark

Sébastien Tavenas
Chargé de Recherche CNRS au Laboratoire LAMA de l'Université Savoie Mont Blanc, France

Amir Yehudayoff
Department of Computer Science, The University of Copenhagen, Denmark
Department of Mathematics, Technion-IIT, Haifa, Israel

Abstract

It is a well-known fact that the permanent polynomial is complete for the complexity class VNP, and it is largely suspected that the determinant does not share this property, despite its similar expression. We study the question of why the VNP-completeness proof of the permanent fails for the determinant. We isolate three fundamental properties that are sufficient to prove a polynomial sequence is VNP-hard, of which two are shared by both the permanent and the determinant. We proceed to show that the permanent satisfies the third property, which we refer to as the "cost of a boolean sum", while the determinant does not, showcasing the fundamental difference between the polynomial families. We further note that this differentiation also applies in the border complexity setting and that our results apply for counting complexity.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory

Keywords and phrases Algebraic Complexity, Computational Complexity, Permanent, Determinant

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2025.47

Funding *Ian Orzel*: funded by the European Research Council (ERC) under grant agreement no. 101125652 (ALBA).
Srikanth Srinivasan: funded by the European Research Council (ERC) under grant agreement no. 101125652 (ALBA).
Sébastien Tavenas: supported by ANR-22-CE48-0007.
Amir Yehudayoff: supported by a DNRF chair grant, and partially supported by BARC.

1 Introduction

In an attempt to introduce an algebraic approach to solving the classical P versus NP problem, [15] introduces algebraic circuits and the complexity classes VP and VNP. Central to this approach were two fundamental objects: the determinant and the permanent. The determinant belongs to the class VP (it is, in fact, complete for a restriction of this class, VBP, or even VP if we consider quasi-polynomial projections), and the permanent is complete for the class VNP, as proven in [15]. We suspect that the determinant is not VNP-complete, meaning that Valiant's proof of the VNP-hardness of the permanent should fail when the determinant is used in its place. It has been unclear exactly why this is the case, although one would expect it to be related to the construction of a particular graph-theoretical gadget, known as the *iff-coupling* gadget, being impossible in the model of the determinant. However, this is a somewhat narrow and technical reason.

This paper seeks to develop a more general understanding of the reasons this VNP-hardness proof fails when applied to the determinant. In exploring this question, we simplify the proof for the VNP-completeness of the permanent and isolate the exact property that the determinant lacks which causes this VNP-hardness proof to fail (assuming all other properties are the same).

Recall that the VNP-completeness proof of the permanent begins by taking a formula and summing it over all 0-1 substitutions of some of its variables (representing an element of a VNP polynomial sequence). This formula is then transformed into a similarly-sized permanent. We then iteratively remove each 0-1 substitution of a particular variable by modifying the matrix we take the permanent of, only increasing its size by a small amount. This iterative removal is the key piece of the proof that makes it work in the case of the permanent, but, to the authors' knowledge, there does not exist a proof that this step breaks down for the determinant. Studying this step in its full generality is difficult, as the given matrix could have a complex structure. In this paper, we simplify the VNP-completeness proof by showing that we can, without loss of generality, assume that the matrix has exactly two instances of the variable that we seek to remove. Using a small (standard) gadget construction, a boolean sum over two variables can be simulated by a permanent of a matrix that is bigger *only by an additive constant*, which shows that the permanent is VNP-complete. We can then show that the proof fails for the determinant by proving that the last step of this process must incur a super-constant (in fact, a constant *multiplicative factor*) increase in the size of the matrix.

We generalize this concept of "iterated substitutions" to arbitrary polynomials through the boolean sum. Given a polynomial $f \in \mathbb{F}[\mathcal{X}]$ and a subset of its variables $S \subseteq \mathcal{X}$, we can represent the boolean sum, which we denote by $\Sigma_S f$, as the sum of taking f with each variable in S set to 0 and the same with respect to 1. We can now rephrase this step of the proof as converting a boolean sum over an arbitrary-sized set of variables to multiple boolean sums over two variables.

In this paper, we have isolated the following properties as sufficient for proving the VNP-hardness of a polynomial:

1. The polynomial sequence is VF-hard, i.e., a polynomial computed by a formula can be represented as a projection of a polynomial in the sequence (we can replace arguments by variables or constants), indexed by a polynomial in the size of the formula,
2. A boolean sum over many variables can be rewritten as a sequence of boolean sums of the polynomial, each over two variables, and
3. A boolean sum of a polynomial over two variables can be rewritten as a simple projection of itself, whose index is only increased by a fixed constant.

With these properties in mind, the main results of our paper can be summarized as follows:

- A polynomial sequence satisfying properties (1), (2), and (3) is VNP-hard.
- Both the determinant and the permanent satisfy properties (1) and (2).
- The permanent satisfies property (3), but the determinant does not.

In short, we have isolated the simple property that differentiates the determinant from the permanent in our ability to prove VNP-completeness.

1.1 Related Work

The focus of this paper is on studying the noteworthy differences in properties between the permanent and the determinant polynomials, which is an important research direction, as it seems that the separation between VP and VNP is encoded in the differences between these polynomials. It is therefore that the differences in these polynomials have been well-studied.

One significant line of study was that seeking to find lower bounds on the size of a determinant that simulates the permanent (determinantal complexity). In a result of von zur Gathen [17], a lower bound was determined by studying the algebraic-geometric properties of the determinant and the permanent. Specifically, there was a difference between the dimension of the spaces of singular points of the determinant and permanent (points where a polynomial and all of its first-order partial derivatives vanish). Then, in a result of Mignon and Ressayre [11] (see also [3]), the bound was strengthened by studying the rank of the Hessian of these polynomials at vanishing points. It was shown that all zeros of the determinant have a very small Hessian rank, whereas the permanent has zero points with very large Hessian rank.

The work of Landsberg and Ressayre [8] gives an exponential lower bound on the determinantal complexity of the permanent in a restricted setting. They showed that any reductions from the permanent to the determinant that preserve the underlying symmetries must have exponential cost.

Recent work of Hrubeš and Joglekar [6] points to a distinction in reductions to *read-once* determinants and permanents, which forces each variable to appear in the corresponding matrix at most once. Here, they show that every multilinear polynomial can be represented as a read-once permanent, whereas representations using the determinant could require a variable to appear at least $\omega(\sqrt{n}/\lg n)$ times.

1.2 Our Results

We have isolated three properties that are fundamental to understanding our VNP-completeness results for the permanent. We will begin by formally stating these properties.

▶ **Definition 1.** *We say that a polynomial sequence $f = (f_i)_{i\in\mathbb{N}} \in \mathbb{F}[\mathcal{X}]$* simulates formulas *if, for every polynomial $g \in \mathbb{F}[\mathcal{X}]$ that can be computed by a formula of size s, g is a projection of some f_i, where i is polynomial in s. In other words, f is VF-hard.*

We further say that $k \in \mathbb{N}$ variable instances are sufficient for boolean sums of *f if, for every boolean sum $\Sigma_{S_1} \dots \Sigma_{S_\ell} f_i$ [see (2.1)], there is an ℓ' and m that are polynomial in i and ℓ such that this sum can be rewritten as $\Sigma_{T_1} \dots \Sigma_{T_{\ell'}} f_m$, where the size of each T_i is k.*

Finally, we define the algebraic additive cost of boolean summation over f via the map $\alpha_f : \mathbb{N} \to \mathbb{N} \cup \{\infty\}$, defined as follows: for each $s \in \mathbb{N}$, let $\alpha_f(s)$ be the minimum integer[1] *k so that, for every n and $S \subseteq \mathcal{X}_n$ of size $|S| \leq s$, the polynomial $\Sigma_S f_n$ is a simple projection of f_{n+k}. In other words, $\alpha_f(s)$ is the additive f-cost of performing a boolean sum over s variables.*

We can further naturally move the definition of α_f into the border complexity setting, which we denote by $\underline{\alpha_f}$. Formally, we let $\underline{\alpha_f}(s)$ be the minimum integer k so that there is a border sequence $F = \overline{(F_i)}_{i\in\mathbb{N}}$ over $\mathbb{F}(\epsilon)[\mathcal{X}]$ such that, for every $i \in \mathbb{N}$, there exist polynomials $g_i \in \mathbb{F}[\mathcal{X} \cup \{\epsilon\}]$ and $Q_i \in \mathbb{F}[\epsilon]$ and $M \in \mathbb{Z}$ such that

$$F_i = \frac{\epsilon^M \cdot f_i + \epsilon^{M+1} \cdot g_i}{1 + \epsilon Q_i(\epsilon)},$$

and furthermore, $\alpha_F(s) = k$. The algebraic expression captures our intuition that $\lim_{\epsilon\to 0} F_i = f_i$.

[1] If there is no such integer, then it is infinite.

▶ Remark 2. Notice that $\alpha_f(s)$ has no dependence on n (the number of variables) and instead is a bound that applies for all n. One could modify the definition to allow for dependence on n, let's call this new function $\beta_f(s,n)$. Then, instead of the hypothesis we include in Theorem 3, namely $\alpha_f(k) < \infty$, we could rephrase it in terms of $\beta_f(s,n)$. Clearly, $\alpha_f(k) < \infty$ if and only if $\beta_f(s,n) = O(1)$, but, more generally, it is sufficient that the sequence $n, n+\beta_f(s,n), n+\beta_f(s,n+\beta_f(s,n)), \ldots$ iterated n times is polynomially bounded. Observe that a lower bound of $\beta_f(s,n) \geq (1+\epsilon)n$ is enough to make the sequence grow exponentially. To avoid the technicalities involved in characterizing such functions β_f, we have decided to use the $\alpha_f(s)$ notation.

We can now introduce our main results that use these properties. We first have that these properties are sufficient for VNP-hardness.

▶ **Theorem 3.** *If, for a fixed $k \in \mathbb{N}$, $f = (f_i)_{i\in\mathbb{N}} \in \mathbb{F}[\mathcal{X}]$ is such that*

1. *f simulates formulas,*
2. *k variables instances are sufficient for boolean sums over f, and*
3. *$\alpha_f(k) < \infty$,*

then f is VNP-hard.

Further, if the projections and reductions associated with these three properties can be completed in polynomial time, then f is #P-hard.

We use perm and det to denote the permanent and determinant polynomials respectively. It is a well-known fact that perm and det can simulate formulas.

▶ **Lemma 4** ([15])**.** *The* perm *and* det *polynomial sequences can simulate formulas.*

Our main deviation from the well-known VNP-hardness proof is through the decrease of the number of instances of the variables we sum over to two. This idea is stated in the following lemma.

▶ **Lemma 5.** *Two variables instances are sufficient for boolean sums over the* perm *and* det *polynomials.*

Finally, the main result of this paper is through differentiating the determinant from the permanent, which is stated in the following theorem.

▶ **Theorem 6.** *We have that $\alpha_{\mathsf{perm}}(2) \leq 3$, but $\alpha_{\mathsf{det}}(2) = \infty$. We further observe that $\underline{\alpha_{\mathsf{det}}}(2) = \infty$.*

1.3 Structure of Paper

First, in Section 2, we will introduce our notation and definitions. Then, in Section 3, we will prove Theorem 6, which will differentiate the permanent from the determinant. We will do this using a well-known result of Mignon and Ressayre[11] and through a careful selection of matrices. In Section 4, we will prove Lemma 5, showing that we can transform arbitrarily-sized boolean sums to those of size two. In Section 5, we provide the proof of Theorem 3, which is (in our opinion) a slightly simpler and more modular proof of the VNP-completeness of the permanent.

2 Preliminaries

Let $\mathbb{F}$ be a field of characteristic not two. Let $\mathcal{X}$ be an infinite set of indeterminants and $\mathbb{F}[\mathcal{X}]$ be the polynomial ring containing polynomials that use a finite number of indeterminants in $\mathcal{X}$. We will write $(f_i)_{i\in\mathbb{N}} \in \mathbb{F}[\mathcal{X}]$ to denote a sequence a polynomials (we will somtimes simply write it as (f_i)), and it is assumed that $f_i \in \mathbb{F}[\mathcal{X}_i]$, where $\mathcal{X}_i \subseteq \mathcal{X}$ has size polynomial in i. Given a set S, we may use $S^{\mathcal{X}_i}$ to denote an assignment of the variables in $\mathcal{X}_i$ to elements of S, so that we can then see $f_i(a)$ to represent evaluating f_i at the point $a \in S^{\mathcal{X}_i}$.

We will use the typical definitions for circuits, formulas, the complexity classes VF, VP, and VNP, and hardness within these classes. For more information, see [15].

A fundamental tool in computational complexity theory is reduction between problems [7]. Reductions allow us to compare the complexities of different problems. In the algebraic setting, reductions are projections. The polynomial $f \in \mathbb{F}[\mathcal{X}]$ is a projection of $g \in \mathbb{F}[\mathcal{Y}]$ if there is a map $\phi : \mathcal{Y} \to \mathcal{X} \cup \mathbb{F}$ so that $f = \lambda(g \circ \phi)$, where $\lambda \in \mathbb{F} \setminus \{0\}$. The projection is called *simple* if for each $x \in \mathcal{X}$ the cardinality of $\phi^{-1}(x)$ is at most one.

For boolean summation, we use the following notation. For $S \subseteq \mathcal{X}$, denote by f_S the polynomial in the variables $\mathcal{X} \cup \{y\}$, where y is a new variable, that is obtained from f by the substitution $x = y$ for all $x \in S$. For example, $(x_1 + x_2^2 + x_3)_{\{x_1,x_2\}} = y + y^2 + x_3$. Denote by $\Sigma_S f$ the polynomial

$$\Sigma_S f = f_S|_{y=0} + f_S|_{y=1}. \tag{2.1}$$

For example, $\Sigma_{\{x_1,x_2\}}(x_1 + x_2^2 + x_3) = 2 + 2x_3$. The variables in S do not appear in f_S and $\Sigma_S f$.

For two polynomial sequences $g = (g_i)_{i\in\mathbb{N}}$ and $f = (f_i)_{i\in\mathbb{N}}$, we say that g is an iterated boolean sum of f if, for every $i \in \mathbb{N}$, there are disjoint sets $S_1, \ldots, S_\ell \subseteq \mathcal{X}$ such that $g_i = \Sigma_{S_1} \ldots \Sigma_{S_\ell} f_j$, where ℓ and j are polynomial in i. Observe that a polynomial sequence being in VNP is equivalent to there being a sequence in VF each element in the VNP sequence is an iterated boolean sum of a (polynomially-indexed) element in the VF sequence.

The two central polynomial families in algebraic complexity are the permanent and the determinant, defined by,

$$\mathsf{perm}(X) = \mathsf{perm}_n(X) = \sum_{\pi} \prod_{i} x_{i,\pi(i)}$$

and

$$\mathsf{det}(X) = \mathsf{det}_n(X) = \sum_{\pi} (-1)^{\mathsf{sign}(\pi)} \prod_{i} x_{i,\pi(i)},$$

where $X = (x_{i,j})$ is an $n \times n$ matrix of variables, the sum is over permutations π of $[n]$, and the product is over $i \in [n]$. The permanent and determinant can be equivalently defined as the sum of the (signed) weights of cycle covers over a graph, whose adjacency matrix is given; see [9]. The permanent is VNP-complete, so that VP $\neq$ VNP if and only if perm is not in VP [15]. It also appears in many places in computer science because it encapsulates many counting problems (see e.g. [16, 14, 1] and references within); by definition, it counts the number of perfect matchings in a bipartite graph. The VP versus VNP problem is (essentially) about the relationship between the permanent and the determinant (see [15, 12, 10] and references within).

The determinant defines a complexity measure; the determinantal complexity $\mathsf{dc}(f)$ of a polynomial f is the minimum k so that f is a projection of det_k. It is known that $\mathsf{dc}(f) < \infty$ for all f. The VP versus VNP problem is roughly captured by *what is* $\mathsf{dc}(\mathsf{perm}_n)$? Proving that $\mathsf{dc}(\mathsf{perm}_n)$ is super-polynomial in n is one of the major open problems in computer science. The best lower bound we know is $\mathsf{dc}(\mathsf{perm}_n) \geq \frac{n^2}{2}$; see [11, 3].

Given a complexity measure $L : \mathbb{F}[\mathcal{X}] \to \mathbb{N}$, we can define its corresponding "border" measure, denoted $\underline{L}$, to be the polynomial with the smallest L measure that approximates the original polynomial. More formally, given an $f \in \mathbb{F}[\mathcal{X}]$, we define $\underline{L}(f)$ to be the smallest k such that there exist $F \in \mathbb{F}(\epsilon)[\mathcal{X}]$, $g \in \mathbb{F}[\mathcal{X} \cup \{\epsilon\}]$, $Q \in \mathbb{F}[\epsilon]$, and $M \in \mathbb{Z}$ such that

$$F = \frac{\epsilon^M f + \epsilon^{M+1} g}{1 + \epsilon Q(\epsilon)},$$

and furthermore, $L(F) = k$.

In the field of counting complexity, we study the complexity of functions that "count" some quantity, mapping $\varphi : \{0,1\}^* \to \mathbb{N}$. Recall that we define the class NP as the set of decision problems, denoted by $\varphi : \{0,1\}^* \to \{0,1\}$, such that there is a Turing Machine taking in the input and a special "witness" string such that the Turing Machine runs in polynomial time and a string is in the language if and only if there is at least one "witness" string for which the Turing Machine accepts it. We then define the complexity class ♯P similarly, but instead we require the related Turing Machine to have the number of accepting witness strings to a given input to be the same as that defined by φ. The quintessential ♯P-complete problem is ♯3-SAT, which, given a 3-CNF formula, counts the number of satisfying assignments.

3 Cost of Boolean Summation

This section will focus on proving Theorem 6. We will separate the proof of this theorem into two sub-claims. The first is used to show that $\alpha_{\mathsf{det}}(2) = \infty$.

▷ Claim 7. For any $n \in \mathbb{N}$, we have that

$$\mathsf{dc}(\Sigma_{\{x_{1,1},x_{2,2}\}}\mathsf{det}_n) \geq 2(n-2).$$

Next, we will prove that $\alpha_{\mathsf{perm}}(2) \leq 3$ through another claim.

▷ Claim 8. There exists a matrix $E \in (\mathcal{X} \cup \mathbb{F})^{(n+3)\times(n+3)}$ such that

$$\Sigma_{\{x_{1,1},x_{1,2}\}}\mathsf{perm} = \mathsf{perm}(E).$$

Observe that, due to row and column swaps (and the fact that, if the two indeterminants were on the same row or column, the claim would be trivially true), these claims imply what we want.

3.1 Lower Bound on Determinantal Boolean Sums

We now prove that $\alpha_{\mathsf{det}}(2) = \infty$ by proving Claim 7. The proof relies on a mechanism developed by Mignon and Ressayre to lower bound the determinantal complexity [11]. For a polynomial $f \in \mathbb{F}[\mathcal{X}]$, denote by H_f the $|\mathcal{X}| \times |\mathcal{X}|$ Hessian matrix of f defined by

$$(H_f)_{x,x'} = \frac{\partial^2}{\partial x \partial x'} f.$$

▶ **Lemma 9** ([11]; see also Lemma 13.3 in [4]). *Let f be a polynomial in the variables $x = (x_1, \ldots, x_n)$ and let $a \in \mathbb{F}^n$ be so that $f(a) = 0$ then*

$$\mathsf{dc}(f) \geq \frac{\mathsf{rank}(H_f|_{x=a})}{2}.$$

The theorem follows by considering the polynomial

$$f = \Sigma_{\{x_{1,1},x_{2,2}\}}\mathsf{det}_n$$

and the $n \times n$ matrix

$$A = \begin{bmatrix} & \frac{1}{2} & \\ 1 & & \\ & & I \end{bmatrix},$$

where I is the $(n-2) \times (n-2)$ identity matrix. In the two claims below (which complete the proof), let $H = H_f$.

▷ Claim 10. $f(A) = 0$.

Proof of Claim 10.

$$f(A) = \mathsf{det}\left(\begin{bmatrix} & \frac{1}{2} & \\ 1 & & \\ & & I \end{bmatrix}\right) + \mathsf{det}\left(\begin{bmatrix} 1 & \frac{1}{2} & \\ 1 & 1 & \\ & & I \end{bmatrix}\right)$$
$$= -\frac{1}{2} + 1 - \frac{1}{2} = 0.$$ ◁

▷ Claim 11. $\mathsf{rank}(H|_{X=A}) \geq 4(n-2)$.

Proof of Claim 11. Focus on the $4(n-2)$ variables $\mathcal{V}_i = \{x_{1,i}, x_{2,i}, x_{i,1}, x_{i,2}\}$ for $i > 2$. Furthermore, we consider G to be the $4(n-2) \times 4(n-2)$ sub-matrix of H whose rows and columns are labeled by variables in $\bigcup_{i>2} \mathcal{V}_i$. Let $f_0 = \mathsf{det}_n|_{x_{1,1}=x_{2,2}=0}$ and $f_1 = \mathsf{det}_n|_{x_{1,1}=x_{2,2}=1}$, so that $f = f_0 + f_1$.

An entry in the Hessian of det is plus/minus the determinant of the $(n-2)$-minor obtained by deleting the corresponding rows and columns from X (or zero if the two variables share a row or a column).

For convenience, denote by $H_{k\ell mp}$ the $(x_{k,\ell}, x_{m,p})$-entry in H (and in G).

We first claim that after substituting A, the sub-matrix G of H has $n-2$ blocks, each of size 4×4:

$$\begin{bmatrix} G_3 & & & \\ & G_4 & & \\ & & \ddots & \\ & & & G_n \end{bmatrix}$$

where for $i > 2$

$$G_i = \begin{bmatrix} H_{1i1i} & H_{1i2i} & H_{1ii1} & H_{1ii2} \\ H_{2i1i} & H_{2i2i} & H_{2ii1} & H_{2ii2} \\ H_{i11i} & H_{i12i} & H_{i1i1} & H_{i1i2} \\ H_{i21i} & H_{i22i} & H_{i2i1} & H_{i2i2} \end{bmatrix}.$$

Indeed, for $i \neq j$, the matrix obtained from A by deleting row i and column j has a zero row so its determinant is zero. It follows for $k, \ell \in \{1, 2\}$ that

$$(H_f)_{ik\ell j}|_{X=A} = (H_{f_0})_{ik\ell j}|_{X=A} + (H_{f_1})_{ik\ell j}|_{X=A} = 0 + 0 = 0.$$

A similar argument holds for the other entries $H_{ikj\ell}$, $H_{ki\ell j}$, and $H_{kij\ell}$.

Now fix i and focus on G_i. Entries of the form $1i1i$ and $1i2i$ are also zero. So, we are left with

$$G_i = \begin{bmatrix} & & H_{1ii1} & H_{1ii2} \\ & & H_{2ii1} & H_{2ii2} \\ H_{i11i} & H_{i12i} & & \\ H_{i21i} & H_{i22i} & & \end{bmatrix}.$$

The matrix G_i is symmetric so it is enough to understand

$$\begin{bmatrix} H_{1ii1} & H_{1ii2} \\ H_{2ii1} & H_{2ii2} \end{bmatrix}.$$

This matrix is a sum of two matrices (one from f_0 and one from f_1). For f_0, this matrix is

$$\begin{bmatrix} & 1 \\ \frac{1}{2} & \end{bmatrix}.$$

For f_1, this matrix is

$$\begin{bmatrix} -1 & 1 \\ \frac{1}{2} & -1 \end{bmatrix}.$$

The sum of the two matrices is

$$\begin{bmatrix} -1 & 2 \\ 1 & -1 \end{bmatrix},$$

which always has rank two. Rolling back

$$\mathsf{rank}\left(\begin{bmatrix} G_3 & & & \\ & G_4 & & \\ & & \ddots & \\ & & & G_n \end{bmatrix}\right) = 4(n-2). \qquad \lhd$$

▶ **Observation 12.** *We note that this result also applies in the border complexity setting. Specifically, it also follows that* $\underline{\mathsf{dc}}(\Sigma_{\{x_{1,1},x_{2,2}\}}\mathsf{det}_n) \geq 2(n-2)$. *This fact follows from [5], where it is shown that the lower bound from [11] of the determinantal complexity of the permanent also applies in the border complexity setting. Although the proof presented there is specifically for the permanent polynomial, it is easy to see that it also works if you use any arbitrary homogeneous polynomial. We should observe that* $\Sigma_{\{x_{1,1},x_{2,2}\}}\mathsf{det}_n$ *is not homogeneous, but one can easily see that we can apply the result to the natural homogenization of the polynomial and get the desired result.*

3.2 Upper Bound on Permanental Boolean Sum

In this section, we will prove that $\alpha_{\mathsf{perm}}(2) \leq 3$ by proving Claim 8. Specifically, we will select the matrix E to be

$$E = \left[\begin{array}{ccc|cccc} 1 & 1 & 1 & & & & \\ 1 & 0 & -1 & 1 & & & \\ -\frac{1}{2} & \frac{1}{2} & \frac{3}{2} & & 1 & & \\ \hline & 1 & & 0 & x_{1,2} & x_{1,3} & \cdots \\ & & 1 & x_{2,1} & 0 & x_{2,3} & \\ & & & x_{3,1} & x_{3,2} & x_{3,3} & \\ & & & \vdots & & & \ddots \end{array}\right].$$

Proof of Claim 8. To see this, denote by $X[b]$ the matrix X after the substitution $x_{1,1} = x_{2,2} = b$ for $b \in \{0,1\}$. For $S \subseteq \{1,2\}$, denote by $X[b]_{-S}$ the matrix $X[b]$ after deleting the rows and columns in S. We have

$$\mathsf{perm}(X[1]) = \mathsf{perm}\left(\begin{bmatrix} 1 & x_{1,2} & x_{1,3} & \\ x_{2,1} & 1 & x_{2,3} & \\ x_{3,1} & x_{3,2} & x_{3,3} & \\ & & & \ddots \end{bmatrix}\right)$$
$$= \mathsf{perm}(X[0]) + \mathsf{perm}(X[0]_{-\{1\}}) + \mathsf{perm}(X[0]_{-\{2\}}) + \mathsf{perm}(X[0]_{-\{1,2\}}).$$

Now, consider a permutation π such that the associated value of the permutation on E is nonzero. We can then notice that such a permutation falls into one of four categories:

- $\pi(\{1,2,3\}) = \{1,2,3\}$,
- $\pi(2) = 4$ and $\pi(\{1,3\}) = \{1,3\}$,
- $\pi(3) = 5$ and $\pi(\{1,2\}) = \{1,2\}$, or
- $\pi(2) = 4$, $\pi(3) = 5$, and $\pi(1) = 1$.

Let U be the 3×3 upper-left block

$$U = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & -1 \\ -\frac{1}{2} & \frac{1}{2} & \frac{3}{2} \end{bmatrix}.$$

For $S, T \subseteq \{1,2,3\}$ with $|S| = |T|$, we denote by U_S^T the sub-matrix of U obtained by keeping the rows of S and the columns of T. The matrix U is chosen such that

$$\mathsf{perm}(U^{\{1,3\}}_{\{1,2\}}) = \mathsf{perm}(U^{\{1,2\}}_{\{1,3\}}) = 0,$$
$$\mathsf{perm}(U^{\{1\}}_{\{1\}}) = \mathsf{perm}(U^{\{1,2\}}_{\{1,2\}}) = \mathsf{perm}(U^{\{1,3\}}_{\{1,3\}}) = 1,$$
$$\mathsf{perm}(U) = 2.$$

In the first of the above 4 cases, it must be that $\pi(\{4,5,\ldots,m\}) = \{4,5,\ldots,m\}$. Therefore, the sum of the weights of these permutations corresponds to

$$\mathsf{perm}\left(\left[\begin{array}{c|c} U & \\ \hline & X[0] \end{array}\right]\right) = 2\mathsf{perm}(X[0]).$$

In the second case, it must be that $\pi(4) = 2$ and $\pi(\{5,6,\ldots,m\}) = \{5,6,\ldots,m\}$. Thus, this case corresponds to

$$\mathsf{perm}\left(\left[\begin{array}{c|c} U^{\{1,3\}}_{\{1,3\}} & \\ \hline & X[0]_{-\{1\}} \end{array}\right]\right) = \mathsf{perm}(X[0]_{-\{1\}}).$$

In the third case, we have that $\pi(5) = 3$ and $\pi(\{4, 6, 7, \dots, m\}) = \{4, 6, 7, \dots, m\}$. Thus, this case corresponds to

$$\mathsf{perm}\left(\left[\begin{array}{c|c} U^{\{1,2\}}_{\{1,2\}} & \\ \hline & X[0]_{-\{2\}} \end{array}\right]\right) = \mathsf{perm}(X[0]_{-\{2\}}).$$

Finally, in the fourth case, it must be that $\pi(4), \pi(5) \in \{2, 3\}$ and $\pi(\{6, 7, \dots, m\}) = \{6, 7, \dots, m\}$. This case then corresponds to

$$\mathsf{perm}\left(\left[\begin{array}{c|c} U^{\{1\}}_{\{1\}} & \\ \hline & X[0]_{-\{1,2\}} \end{array}\right]\right) = \mathsf{perm}(X[0]_{-\{1,2\}}).$$

Summing it together,

$$\mathsf{perm}(E) = \mathsf{perm}(X[0]) + \mathsf{perm}(X[1]).$$ ◁

3.3 Boolean Sum over a Single Variable

Interestingly, considering boolean sums of two variables is necessary to separate perm from det. This is becaues the boolean cost of removing one variable is the same for both polynomial sequences.

▷ Claim 13. We have that $\alpha_{\mathsf{det}}(1) = \alpha_{\mathsf{perm}}(1) = 0$.

Proof.

$$\begin{aligned}
&\Sigma_{\{x_{1,1}\}}\mathsf{det}(X) \\
&= \mathsf{det}\begin{bmatrix} 0 & x_{1,2} & x_{1,3} & \dots & x_{1,n} \\ x_{2,1} & x_{2,2} & x_{2,3} & \dots & x_{2,n} \\ x_{3,1} & x_{3,2} & x_{3,3} & \dots & x_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & x_{n,3} & \dots & x_{n,n} \end{bmatrix} + \mathsf{det}\begin{bmatrix} 1 & x_{1,2} & x_{1,3} & \dots & x_{1,n} \\ x_{2,1} & x_{2,2} & x_{2,3} & \dots & x_{2,n} \\ x_{3,1} & x_{3,2} & x_{3,3} & \dots & x_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & x_{n,3} & \dots & x_{n,n} \end{bmatrix} \\
&= 2\mathsf{det}\begin{bmatrix} 1/2 & x_{1,2} & x_{1,3} & \dots & x_{1,n} \\ x_{2,1} & x_{2,2} & x_{2,3} & \dots & x_{2,n} \\ x_{3,1} & x_{3,2} & x_{3,3} & \dots & x_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & x_{n,3} & \dots & x_{n,n} \end{bmatrix}.
\end{aligned}$$

A similar calculation holds for the permanent. ◁

4 Two variables suffice for VNP-hardness

The aim of this section is to prove Lemma 5 that shows that summation over two variables suffices for VNP-hardness. We prove it for the determinant (the proof for the permanent is similar). The following construction shows how to replace a boolean summation over m variables by m boolean summations over two variables.

▶ **Observation 14.** *Let $f \in \mathbb{F}[\mathcal{X}]$ and $S = \{x_1, \dots, x_m\} \subseteq \mathcal{X}$. Let $y_1, \dots, y_m$ be m new variables, and define*

$$g(y) = \Big(\prod_{i\in[m]} y_i\Big) + \Big(\prod_{i\in[m]} (1 - y_i)\Big).$$

Then

$$\Sigma_{\{x_1,y_1\}}\Sigma_{\{x_2,y_2\}}\dots\Sigma_{\{x_m,y_m\}}gf = \Sigma_S f.$$

Indeed, on the l.h.s. there is a sum over 2^m terms, but g picks exactly two of them that perfectly agree with the r.h.s.

The observation allows to replace a boolean sum over many variables by several sums, each over just two.

▶ **Observation 15.** *Let g be the polynomial defined above. Then, there is a $2m \times 2m$ matrix A with entries in $\{y_1,\dots,y_m,0,1,-1\}$ so that each y_i appears once in A and*

$$g = \mathsf{det}(A).$$

The matrix A is the m-variate version of the following matrix:

$$A = \left[\begin{array}{cc|cc|cc|cc} -1 & 1 & -1 & & & & & \\ -1 & y_1 & & & & & & \\ \hline & & -1 & 1 & -1 & & & \\ & & -1 & y_2 & & & & \\ \hline & & & & -1 & 1 & -1 & \\ & & & & -1 & y_3 & & \\ \hline 1 & & & & & & -1 & 1 \\ & & & & & & -1 & y_4 \end{array}\right].$$

To see that $\mathsf{det}(A)$ is the polynomial g, it is easier to see the determinant as a signed sum of the weights of the cycles covers of the graph G_A whose A is the adjacency matrix (see Figure 1)

$$\mathsf{det}(A) = \sum_{C \text{ cycles cover}} \left[\prod_{\sigma \text{ cycle of } C} (-1)^{1+\text{length of } C}\text{weight(C)}\right].$$

The reader could find more details about this characterization by cycles covers in [9]. The only cycle which contains a dashed edge is exactly the union of the dashed edges. So the only cover that contains this cycle also contains the m y_i-loops. It corresponds to one cover of signed weight $\left((-1)^{m+1}(-1)^{m-1}\right)\prod_{i=1}^m y_i = \prod_{i=1}^m y_i$. Otherwise, the covers contain no dashed edges and are given by m disjoint graphs. The signed weight is the product of them: $\prod_i((-1)(-1)+(1)(-y_i)) = \prod_i(1-y_i)$. The sum of the covers gives exactly the polynomial g.

▶ Remark 16. For the permanent, every -1 above the diagonal of the matrix A should be changed into a 1, and the 2×2 blocks into $\begin{bmatrix} -1 & 1 \\ 1 & y_1 \end{bmatrix}$.

Proof of Lemma 5. Let $B \in (\mathcal{X}\cup\mathbb{F})^{n\times n}$ and $S_1,\dots,S_\ell \subseteq \mathcal{X}$ be such that $S_1,\dots,S_\ell$ are disjoint. Then, by previous

$$\Sigma_{S_\ell}\mathsf{det}(B) = \Sigma_{\{x_1,y_1\}}\dots\Sigma_{\{x_m,y_m\}}g\mathsf{det}(B) = \Sigma_{\{x_1,y_1\}}\dots\Sigma_{\{x_m,y_m\}}\mathsf{det}\begin{pmatrix} B & 0 \\ 0 & A \end{pmatrix}$$

Notice here that $m \le |S_\ell|$ and the size of the block matrix is at most $n+2m \le n+2|S_\ell|$. We can then use induction on $S_1,\dots,S_{\ell-1}$ to conclude that we can write

$$\Sigma_{S_1}\dots\Sigma_{S_\ell}\mathsf{det}(A) = \Sigma_{T_1}\dots\Sigma_{T_{\ell'}}\mathsf{det}(C),$$

where $\ell' \le \sum_{i=1}^{\ell}|S_\ell| \le n^2$ and the size of C is at most $n+2\sum_{i=1}^{\ell}|S_\ell| \le n+2n^2$. ◀

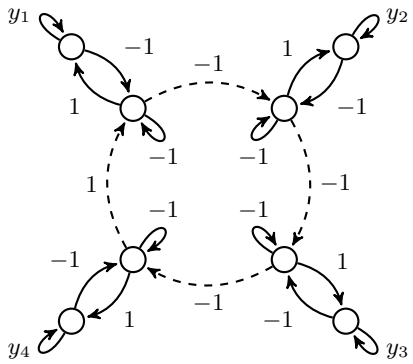

Figure 1 The graph G_A.

5 The permanent is VNP-complete

This section will finally provide the proof of Theorem 3, showing that the three properties we have isolated are sufficient for VNP-hardness. The proof proceeds similarly to Valiant's VNP-completeness proof of the permanent with a few variations. This section thus also provides a (arguably) more modular proof of the VNP-completeness of the permanent.

Proof of Theorem 3. Because f simulates formulas, we can write every n-variate $g \in \mathbb{F}[\mathcal{Y}]$ in VNP (see [2, 13]) as

$$g = (\Sigma_{S_1} \dots \Sigma_{S_\ell} f_d)(a)$$

where $a \in (\mathbb{F} \cup \mathcal{Y})^{\mathcal{X}_d}$ is a variable assignment such that d is polynomial in n, each entry in a is a variable or a field element, and $S_1, \dots, S_\ell$ are pairwise disjoint (implying ℓ is polynomial in d).

Because k variables are sufficient for boolean sums over f, we can assume without loss of generality that $|S_1| = \dots = |S_\ell| = k$. Further, as $\alpha_f(k) < \infty$, we know that we can write

$$\Sigma_{S_\ell} f_d = f_{d+\alpha_f(k)}(b),$$

for some simple variable assignment $b \in (\mathbb{F} \cup \mathcal{Y})^{\mathcal{X}_{i+\alpha_f(k)}}$. Observe that, because this is a simple projection, we can easily (after modifying our sets $S_1, \dots, S_{\ell-1}$) bring out b and write $(\Sigma_{S_1} \dots \Sigma_{S_{\ell-1}} f_{d+\alpha_f(k)})(b') = g$, for some assignment b'. Thus, after repeating this process ℓ times, we can write

$$(\Sigma_{S_1} \dots \Sigma_{S_\ell} f_d)(a) = f_{d+\ell\alpha_f(k)}(c),$$

where $c \in (\mathbb{F} \cup \mathcal{Y})^{\mathcal{X}_{d+\ell\alpha_f(k)}}$ is a variable assignment. Because ℓ and d are polynomial in n, we can conclude that f is VNP-hard. ◀

▶ **Observation 17.** *We will then conclude with the observation that this proof also works for proving ♯P-hardness if each of the steps that we took in this proof can be done in polynomial time. Consider ♯3-SAT, a complete problem for ♯P, where we are given a 3-CNF and must return the number of satisfying assignments it has. Consider some 3-CNF φ in n variables, and observe that the number of possible clauses is polynomial in n, so we can easily encode it using one bit for each possible clause. We will write this number as m_n, and one can easily see that there is a polynomial $f_n \in \mathbb{F}[y_1, \dots, y_{m_n}, x_1, \dots, x_n]$ such that, if we provide*

the y_i variables with the 0/1 encoding of φ and the x_i variables with a variable assignment, it will output 0/1 depending on whether this assignment is a satisfying assignment of φ. One can easily see that f_n can be computed by a formula that is polynomial in n. Finally, we observe that the polynomial $g_n = \Sigma_{\{x_1\}} \dots \Sigma_{\{x_n\}} f_n$ provides the exact answer for ♯3SAT when given the binary encoding of a 3-CNF. Hence, we can carry out our previous proof from the sequence $g = (g_n)_{n \in \mathbb{N}}$ to reduce it to a polynomial with the three properties (ensuring that the steps along the way can be done in polynomial time). At the end, we conclude that we can reduce ♯3-SAT to our polynomial sequence.

References

1 Scott Aaronson. A linear-optical proof that the permanent is #P-hard. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 467(2136):3393–3405, 2011.

2 Peter Bürgisser. *Completeness and reduction in algebraic complexity theory*, volume 7. Springer Science & Business Media, 2000.

3 Jin-Yi Cai, Xi Chen, and Dong Li. Quadratic lower bound for permanent vs. determinant in any characteristic. *computational complexity*, 19(1):37–56, 2010. doi:10.1007/S00037-009-0284-2.

4 Xi Chen, Neeraj Kayal, and Avi Wigderson. Partial derivatives in arithmetic complexity and beyond. *Foundations and Trends in Theoretical Computer Science*, 6(1–2):1–138, 2011. doi:10.1561/0400000043.

5 Joshua A. Grochow. Unifying known lower bounds via geometric complexity theory. *computational complexity*, 24:393–475, 2015. doi:10.1007/S00037-015-0103-X.

6 Pavel Hrubes and Pushkar S. Joglekar. On read-k projections of the determinant. *Electron. Colloquium Comput. Complex.*, TR24-125, 2024. URL: https://eccc.weizmann.ac.il/report/2024/125.

7 Richard M. Karp. On the computational complexity of combinatorial problems. *Networks*, 5(1):45–68, 1975. doi:10.1002/NET.1975.5.1.45.

8 Joseph M. Landsberg and Nicolas Ressayre. Permanent v. determinant: An exponential lower bound assuming symmetry. In Madhu Sudan, editor, *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 29–35. ACM, 2016. doi:10.1145/2840728.2840735.

9 Meena Mahajan and V Vinay. Determinant: Old algorithms, new insights. *SIAM journal on Discrete Mathematics*, 12(4):474–490, 1999. doi:10.1137/S0895480198338827.

10 Guillaume Malod and Natacha Portier. Characterizing Valiant's algebraic complexity classes. *Journal of complexity*, 24(1):16–38, 2008. doi:10.1016/J.JCO.2006.09.006.

11 Thierry Mignon and Nicolas Ressayre. A quadratic bound for the determinant and permanent problem. *International Mathematics Research Notices*, (79):4241–4253, 2004.

12 Ketan D Mulmuley. On P vs. NP and geometric complexity theory. *Journal of the ACM*, 58(2):1–26, 2011.

13 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. doi:10.1561/0400000039.

14 Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991. doi:10.1137/0220053.

15 Leslie G. Valiant. Completeness classes in algebra. In *STOC*, pages 249–261, 1979. doi:10.1145/800135.804419.

16 Leslie G. Valiant. The complexity of computing the permanent. *Theoretical computer science*, 8(2):189–201, 1979. doi:10.1016/0304-3975(79)90044-6.

17 Joachim von zur Gathen. Permanent and determinant. In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 398–401. IEEE Computer Society, 1986. doi:10.1109/SFCS.1986.42.