Electronic Colloquium on Computational Complexity, Revision 2 of Report No. 38 (2016)

Some Complete and Intermediate Polynomials in Algebraic Complexity Theory

Meena Mahajan and Nitin Saurabh

The Institute of Mathematical Sciences, HBNI, Chennai, India
meena@imsc.res.in, nitin@imsc.res.in

Abstract. We provide a list of new natural VNP-intermediate polynomial families, based on basic (combinatorial) NP-complete problems that are complete under *parsimonious* reductions. Over finite fields, these families are in VNP, and under the plausible hypothesis $\mathsf{Mod}_p\mathsf{P} \not\subseteq \mathsf{P/poly}$, are neither VNP-hard (even under oracle-circuit reductions) nor in VP. Prior to this, only the Cut Enumerator polynomial was known to be VNP-intermediate, as shown by Bürgisser in 2000.
We next show that over rationals and reals, the clique polynomial cannot be obtained as a monotone p-projection of the permanent polynomial, thus ruling out the possibility of transferring monotone clique lower bounds to the permanent. We also show that two of our intermediate polynomials, based on satisfiability and Hamiltonian cycle, are not monotone affine polynomial-size projections of the permanent. These results augment recent results along this line due to Grochow.
Finally, we describe a (somewhat natural) polynomial defined independent of a computation model, and show that it is VP-complete under polynomial-size projections. This complements a recent result of Durand et al. (2014) which established VP-completeness of a related polynomial but under constant-depth oracle circuit reductions. Both polynomials are based on graph homomorphisms. Variants yield families similarly complete for VBP and VNP.

1 Introduction

The algebraic analogue of the P versus NP problem, famously referred to as the VP versus VNP question, is one of the most significant problem in algebraic complexity theory. Valiant [29] showed that the Permanent polynomial is VNP-complete (over fields of char $\neq 2$). A striking aspect of this polynomial is that the underlying decision problem, in fact even the search problem, is in P. Given a graph, we can decide in polynomial time whether it has a perfect matching, and if so find a maximum matching in polynomial time [12]. Since the underlying problem is an easier problem, it helped in establishing VNP-completeness of a host of other polynomials by a reduction from the Permanent polynomial (cf. [4]). Inspired from classical results in structural complexity theory, in particular [21], Bürgisser [5] proved that if Valiant's hypothesis (i.e. VP $\neq$ VNP) is true, then, over any field there is a p-family in VNP which is neither in VP nor VNP-complete with respect to c-reductions. Let us call such polynomial families VNP-intermediate (i.e. in VNP, not VNP-complete, not in VP). Further, Bürgisser [5] showed that over finite fields, a *specific* family of polynomials is VNP-intermediate, provided the polynomial hierarchy PH does not collapse to the second level. On an intuitive level these polynomials enumerate *cuts* in a graph. This is a remarkable result, when compared with the classical P-NP setting or the BSS-model. Though the existence of problems with intermediate complexity has been established in the latter settings, due to the involved "diagonalization" arguments used to construct them, these problems seem highly unnatural. That is, their definitions are not motivated by an underlying combinatorial

ISSN 1433-8092

problem but guided by the needs of the proof and, hence, seem artificial. The question of whether there are other naturally-defined VNP-intermediate polynomials was left open by Bürgisser [4]. We remark that to date the *cut enumerator* polynomial from [5] is the only known example of a natural polynomial family that is VNP-intermediate.

The question of whether the classes VP and VNP are distinct is often phrased as whether Perm_n is *not* a quasi-polynomial-size projection of Det_n. The importance of this reformulation stems from the fact that it is a purely algebraic statement, devoid of any dependence on circuits. While we have made very little progress on this question of determinantal complexity of the permanent, the progress in restricted settings has been considerable. One of the success stories in theoretical computer science is unconditional lower bound against monotone computations [25, 26, 1]. In particular, Razborov [26] proved that computing the permanent over the Boolean semiring requires monotone circuits of size at least $n^{\Omega(\log n)}$. Jukna [19] observed that if the Hamilton cycle polynomial is a monotone p-projection of the permanent, then, since the clique polynomial is a monotone projection of the Hamiltonian cycle [29] and the clique requires monotone circuits of exponential size [1], one would get a lower bound of $2^{n^{\Omega(1)}}$ for monotone circuits computing the permanent, thus improving on [26]. The importance of this observation is also highlighted by the fact that such a monotone p-projection, over the reals, would give an alternate proof of the result of Jerrum and Snir [18] that computing the permanent by monotone circuits over $\mathbb{R}$ requires size at least $2^{n^{\Omega(1)}}$. (Jerrum and Snir [18] proved that the permanent requires monotone circuits of size $2^{\Omega(n)}$ over $\mathbb{R}$ and the tropical semiring.) The first progress on this question raised in [19] was made recently by Grochow [15]. He showed that the Hamiltonian cycle polynomial is not a monotone sub-exponential-size projection of the permanent. This answered Jukna's specific question about the Hamiltonian cycle in its entirety, but the underlying motivation still remains unanswered. That is, *Is clique a monotone p-projection of the permanent?* Maybe not via the Hamiltonian cycle polynomial, but, say, via the '*satisfiability*' polynomial [29]. It is known (see Section 5 [1]) that clique is a monotone projection of the satisfiability polynomial over $O(n^4)$ variables. Thus it still left open the possibility of transferring monotone circuit lower bounds for clique to the permanent.

While the Perm vs Det problem has become synonymous with the VP vs VNP question, there is a somewhat unsatisfactory feeling about it. This rises from two facts: one, that the VP-hardness of the determinant is known only under the more powerful quasi-polynomial-size projections, and, second, the lack of natural VP-complete polynomials (with respect to polynomial-size projections) in the literature. (In fact, with respect to p-projections, the determinant is complete for the possibly smaller class VBP of polynomial-sized algebraic branching programs.) To remedy this situation, it seems crucial to understand the computation in VP. Bürgisser [4] showed that a generic polynomial family constructed using a topological sort of a generic VP circuit, while controlling the degree, is complete for VP. Raz [24], using the depth reduction of [30], showed that a family of "universal circuits" is VP-complete. Thus both families directly depend on the circuit definition or characterization of VP. Last year, Durand et al. [11] made significant progress and provided a natural, first of its kind, VP-complete polynomial. However, the natural polynomials studied by Durand et al. lacked a bit of punch because their completeness was established under polynomial-size *constant depth c-reductions* rather than projections.

In this paper, we make progress on all three fronts. First, we provide a list of new natural polynomial families, based on basic (combinatorial) NP-complete problems [14] whose completeness is via *parsimonious* reductions [28], that are VNP-intermediate over finite fields (Theorem 1). Then, we answer the main motivation of Jukna by proving that clique itself is not a monotone affine polynomial-size projection of the permanent. Thus this possibility of transferring monotone circuit lower bounds for clique to permanent cannot work. Moreover, we also show that over reals, some of our intermediate polynomials are not monotone affine polynomial-size projections of the permanent (Theorem 6). As in [15], the lower bound results about monotone affine projections are unconditional. Finally, we improve upon [11] by characterizing VP and establishing a natural VP-complete polynomial under polynomial-size projections (Theorem 9). Further modifications yield families similarly complete for VBP (Theorems 10, 11) and VNP (Theorem 12).

Organization of the paper. We give basic definitions in Section 2. Section 3 contains our discussion on intermediate polynomials. In Section 4 we establish lower bounds under monotone affine projections. The discussion on completeness results appears in Section 5. We end in Section 6 with some interesting questions for further exploration.

2 Preliminaries

Algebraic complexity: We say that a polynomial f is a *projection* of g if f can be obtained from g by setting the variables of g to either constants in the field, or to the variables of f. A sequence (f_n) is a *p-projection* of (g_m), if each f_n is a projection of g_t for some $t = t(n)$ polynomially bounded in n. There are other notions of reductions between families of polynomials, like *c-reductions* (polynomial-size oracle circuit reductions), *constant-depth c-reductions*, and *linear p-projections*. For more on these reductions, see [4].

An arithmetic circuit is a directed acyclic graph with leaves labeled by variables or constants from an underlying field, internal nodes labeled by field operations $+$ and $\times$, and a designated output gate. Each node computes a polynomial in a natural way. The polynomial computed by a circuit is the polynomial computed at its output gate. A *parse tree* of a circuit captures monomial generation within the circuit. Duplicating gates as needed, unwind the circuit into a formula (fan-out one); a parse tree is a minimal sub-tree (of this unwound formula) that contains the output gate, that contains all children of each included $\times$ gate, and that contains exactly one child of each included $+$ gate. For a complete definition see [22]. A circuit is said to be *skew* if at every $\times$ gate, at most one incoming edge is the output of another gate.

A family of polynomials $(f_n(x_1, \ldots, x_{m(n)}))$ is called a p-family if both the degree $d(n)$ of f_n and the number of variables $m(n)$ are bounded by a polynomial in n. A p-family is in VP (resp. VBP) if a circuit family (skew circuit family, resp.) (C_n) of size polynomially bounded in n computes it. A sequence of polynomials (f_n) is in VNP if there exist a sequence (g_n) in VP, and polynomials m and t such that for all n, $f_n(\bar{x}) = \sum_{\bar{y} \in \{0,1\}^{t(\bar{x})}} g_n(x_1, \ldots, x_{m(n)}, y_1, \ldots, y_{t(n)})$.

(VBP denotes the algebraic analogue of branching programs. Since these are equivalent to skew circuits, we directly use a skew circuit definition of VBP.)

Boolean complexity: We need some basics from Boolean complexity theory. Let $\mathsf{P/poly}$ denote the class of languages decidable by polynomial-sized Boolean circuit families. A function $\phi : \{0,1\}^* \to \mathbb{N}$ is in $\#\mathsf{P}$ if there exists a polynomial p and a polynomial time deterministic Turing machine M such that for all $x \in \{0,1\}^*$, $f(x) = |\{y \in \{0,1\}^{p(|x|)} \mid M(x,y) = 1\}|$. For a prime p, define

$$\#_p\mathsf{P} = \{\psi : \{0,1\}^* \to \mathbb{F}_p \mid \psi(x) = \phi(x) \bmod p \text{ for some } \phi \in \#\mathsf{P}\},$$
$$\mathsf{Mod}_p\mathsf{P} = \{L \subseteq \{0,1\}^* \mid \text{ for some } \phi \in \#\mathsf{P},\, x \in L \iff \phi(x) \equiv 1 \bmod p\}$$

It is easy to see that if $\phi : \{0,1\}^* \to \mathbb{N}$ is $\#\mathsf{P}$-complete with respect to parsimonious reductions (that is, for every $\psi \in \#P$, there is a polynomial-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ such that for all $x \in \{0,1\}^*$, $\psi(x) = \phi(f(x))$), then the language $L = \{x \mid \phi(x) \equiv 1 \bmod p\}$ is $\mathsf{Mod}_p\mathsf{P}$-complete with respect to many-one reductions.

Graph Theory: We consider the treewidth and pathwidth parameters for an undirected graph. We will work with a "canonical" form of decompositions which is generally useful in dynamic-programming algorithms.

A *(nice) tree decomposition* of a graph G is a pair $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$, where T is a tree, rooted at X_r, whose every node t is assigned a vertex subset $X_t \subseteq V(G)$, called a bag, such that the following conditions hold:

1. $X_r = \emptyset$, $|X_\ell| = 1$ for every leaf ℓ of T, and $\cup_{t \in V(T)} X_t = V(G)$.
   That is, the root contain the empty bag, the leaves contain singleton sets, and every vertex of G is in at least one bag.
2. For every $(u,v) \in E(G)$, there exists a node t of T such that $\{u,v\} \subseteq X_t$.
3. For every $u \in V(G)$, the set $T_u = \{t \in V(T) \mid u \in X_t\}$ induces a connected subtree of T.
4. Every non-leaf node t of T is of one of the following three types:
   - **Introduce node:** t has exactly once child t', and $X_t = X_{t'} \cup \{v\}$ for some vertex $v \notin X_{t'}$. We say that v is *introduced* at t.
   - **Forget node:** t has exactly one child t', and $X_t = X_{t'} \setminus \{w\}$ for some vertex $w \in X_{t'}$. We say that w is *forgotten* at t.
   - **Join node:** t has two children t_1, t_2, and $X_t = X_{t_1} = X_{t_2}$.

The *width* of a tree decomposition $\mathcal{T}$ is one less than the size of the largest bag; that is, $\max_{t \in V(T)} |X_t| - 1$. The *tree-width* of a graph G is the minimum possible width of a tree decomposition of G.

In a similar way we can also define a *nice path decomposition* of a graph. For a complete definition we refer to [8].

A sequence (G_n) of graphs is called a p-family if the number of vertices in G_n is polynomially bounded in n. It is further said to have *bounded* tree(path)-width if for some absolute constant c independent of n, the tree(path)-width of each graph in the sequence is bounded by c.

A *homomorphism* from G to H is a map from $V(G)$ to $V(H)$ preserving edges. A graph is called *rigid* if it has *no* homomorphism to itself other than the identity map. Two graphs G and H are called *incomparable* if there are *no* homomorphisms from $G \to H$ as well as $H \to G$. It is known that asymptotically almost all graphs are rigid, and almost all pairs of nonisomorphic graphs are also incomparable. For the purposes of this paper, we only need a collection of three rigid and mutually incomparable graphs. For more details, we refer to [16]. For the purposes of this paper, we only need a collection of three rigid and mutually incomparable graphs. We can use, for instance, the three graphs depicted in Figure 1. For the graph G, in Fig. 1, there is an edge between i and j if $1 \leqslant |i-j| \leqslant 4$. Further add an edge between 1 and 16. The G_i's are obtained, as shown in Fig. 1, by adding an extra edge between 1 and $7+i$. For completeness, we include in the appendix a proof, following the arguments from [16], that these graphs are rigid and pairwise incomparable.

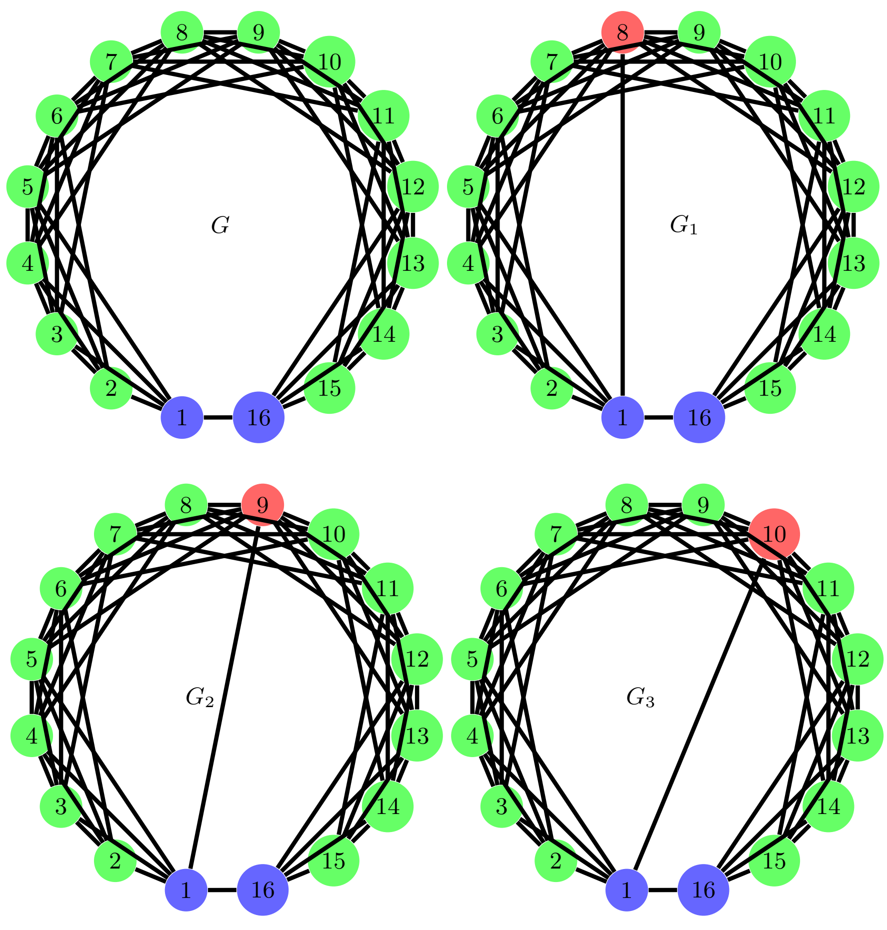

Fig. 1. G_1, G_2, G_3: three rigid pairwise-incomparable graphs

3 VNP-intermediate

In [5], Bürgisser showed that unless PH collapses to the second level, an explicit family of polynomials, called the cut enumerator polynomial, is VNP-intermediate. He raised the question, recently highlighted again in [15], of whether there are other such natural VNP-intermediate polynomials. In this section we show that in fact his proof strategy itself can be adapted to other polynomial families as well. The strategy can be described abstractly as follows: Find an explicit polynomial family $h = (h_n)$ satisfying the following properties.

M: Membership. The family is in VNP.
E: Ease. Over a field $\mathbb{F}_q$ of size q and characteristic p, h can be evaluated in P. Thus if h is VNP-hard, then we can efficiently compute #P-hard functions, modulo p.
H: Hardness. The monomials of h encode solutions to a problem that is #P-hard via parsimonious reductions. Thus if h is in VP, then the number of solutions, modulo p, can be extracted using coefficient computation.

Then, unless $\mathsf{Mod}_p\mathsf{P} \subseteq \mathsf{P/poly}$ (which in turn implies that PH collapses to the second level, [20]), h is VNP-intermediate.

We provide a list of p-families that, under the same condition $\mathsf{Mod}_p\mathsf{P} \not\subseteq \mathsf{P/poly}$, are VNP-intermediate. All these polynomials are based on basic combinatorial NP-complete problems that are complete under parsimonious reduction.
(1) The *satisfiability* polynomial $\mathsf{Sat^q} = (\mathsf{Sat^q}_n)$: For each n, let Cl_n denote the set of all possible clauses of size 3 over $2n$ literals. There are n variables $\tilde{X} = \{X_i\}_{i=1}^n$, and also $8n^3$ clause-variables $\tilde{Y} = \{Y_c\}_{c \in \mathsf{Cl}_n}$, one for each 3-clause c.

$$\mathsf{Sat^q}_n := \sum_{a \in \{0,1\}^n} \left(\prod_{i \in [n]: a_i = 1} X_i^{q-1} \right) \left(\prod_{\substack{c \in \mathsf{Cl}_n \\ a \text{ satisfies } c}} Y_c^{q-1} \right).$$

For the next three polynomials, we consider the complete graph G_n on n nodes, and we have the set of variables $\tilde{X} = \{X_e\}_{e \in E_n}$ and $\tilde{Y} = \{Y_v\}_{v \in V_n}$.
(2) The *vertex cover* polynomial $\mathsf{VC^q} = (\mathsf{VC^q}_n)$:

$$\mathsf{VC^q}_n := \sum_{S \subseteq V_n} \left(\prod_{e \in E_n :\, e \text{ is incident on } S} X_e^{q-1} \right) \left(\prod_{v \in S} Y_v^{q-1} \right).$$

(3) The *clique/independent set* polynomial $\mathsf{CIS^q} = (\mathsf{CIS^q}_n)$:

$$\mathsf{CIS^q}_n := \sum_{T \subseteq E_n} \left(\prod_{e \in T} X_e^{q-1} \right) \left(\prod_{v \text{ incident on } T} Y_v^{q-1} \right).$$

(4) The *clow* polynomial $\mathsf{Clow^q} = (\mathsf{Clow^q}_n)$: A clow in an n-vertex graph is a closed walk of length exactly n, in which the minimum numbered vertex (called the head) appears exactly

once.

$$\mathsf{Clow^q}_n := \sum_{w\text{: clow of length } n} \left(\prod_{e\text{: edges in } w} X_e^{q-1} \right) \left(\prod_{\substack{v\text{: vertices in } w \\ \text{(counted only once)}}} Y_v^{q-1} \right).$$

If an edge e is used k times in a clow, it contributes $X_e^{k(q-1)}$ to the monomial. But a vertex v contributes only Y_v^{q-1} even if it appears more than once. More precisely,

$$\mathsf{Clow^q}_n := \sum_{\substack{w=\langle v_0,v_1,\ldots,v_{n-1}\rangle: \\ \forall j>0, \quad v_0<v_j}} \left(\prod_{i\in[n]} X^{q-1}_{(v_{i-1},v_{i \bmod n})} \right) \left(\prod_{v\in\{v_0,v_1,\ldots,v_{n-1}\}} Y_v^{q-1} \right).$$

(5) The *3D-matching* polynomial $\mathsf{3DM^q} = (\mathsf{3DM^q}_n)$: Consider the complete tripartite hypergraph, where each part in the partition (A_n, B_n, C_n) contain n nodes, and each hyperedge has exactly one node from each part. We have variables X_e for hyperedge e and Y_v for node v.

$$\mathsf{3DM^q}_n := \sum_{M\subseteq A_n\times B_n\times C_n} \left(\prod_{e\in M} X_e^{q-1} \right) \left(\prod_{\substack{v\in M \\ \text{(counted only once)}}} Y_v^{q-1} \right).$$

We show that if $\mathsf{Mod}_p\mathsf{P} \not\subseteq \mathsf{P/poly}$, then all five polynomials defined above are VNP-intermediate.

Theorem 1. *Over a finite field $\mathbb{F}_q$ of characteristic p, the polynomial families $\mathsf{Sat^q}$, $\mathsf{VC^q}$, $\mathsf{CIS^q}$, $\mathsf{Clow^q}$, and $\mathsf{3DM^q}$, are in VNP. Further, if $\mathsf{Mod}_p\mathsf{P} \not\subseteq \mathsf{P/poly}$, then they are all VNP-intermediate; that is, neither in VP nor VNP-hard with respect to c-reductions.*

Proof. (M) An easy way to see membership in VNP is to use Valiant's criterion ([29]; see also Proposition 2.20 in [4]); the coefficient of any monomial can be computed efficiently, hence the polynomial is in VNP. This establishes membership for all families.

We first illustrate the rest of the proof by showing that the polynomial $\mathsf{Sat^q}$ satisfies the properties (H), (E).

(H): Assume $(\mathsf{Sat^q}_n)$ is in VP, via polynomial-sized circuit family $\{C_n\}_{n\geq 1}$. We will use C_n to give a $\mathsf{P/poly}$ upper bound for computing the number of satisfying assignments of a 3-CNF formula, modulo p. Since this question is complete for $\mathsf{Mod}_p\mathsf{P}$, the upper bound implies $\mathsf{Mod}_p\mathsf{P}$ is in $\mathsf{P/poly}$.

Given an instance ϕ of 3SAT, with n variables and m clauses, consider the projection of $\mathsf{Sat^q}_n$ obtained by setting all Y_c for $c \in \phi$ to t, and all other variables to 1. This gives the polynomial $\mathsf{Sat^q}\phi(t) = \sum_{j=1}^{m} d_j t^{j(q-1)}$ where d_j is the number of assignments (modulo p) that satisfy exactly j clauses in ϕ. Our goal is to compute d_m.

We convert the circuit C into a circuit D that compute elements of $\mathbb{F}_q[t]$ by explicitly giving their coefficient vectors, so that we can pull out the desired coefficient. (Note that after the projection described above, C works over the polynomial ring $\mathbb{F}_q[t]$.) Since the polynomial computed by C is of degree $m(q-1)$, we need to compute the coefficients of all intermediate polynomials too only upto degree $m(q-1)$. Replacing $+$ by gates performing coordinate-wise addition, $\times$ by a sub-circuit performing (truncated) convolution, and supplying appropriate coefficient vectors at the leaves gives the desired circuit. Since the number of clauses, m, is polynomial in n, the circuit D is also of polynomial size. Given the description of C as advice, the circuit D can be evaluated in P, giving a $\mathsf{P/poly}$ algorithm for computing #3-SAT(ϕ) mod p. Hence $\mathsf{Mod}_p\mathsf{P} \subseteq \mathsf{P/poly}$.

(E) Consider an assignment to $\tilde{X}$ and $\tilde{Y}$ variables in $\mathbb{F}_q$. Since all exponents are multiples of $(q-1)$, it suffices to consider 0/1 assignments to $\tilde{X}$ and $\tilde{Y}$. Each assignment a contributes 0 or 1 to the final value; call it a contributing assignment if it contributes 1. So we just need to count the number of contributing assignments. An assignment a is contributing exactly when $\forall i \in [n]$, $X_i = 0 \implies a_i = 0$, and $\forall c \in \mathsf{Cl}_n$, $Y_c = 0 \implies a$ does not satisfy c. These two conditions, together with the values of the X and Y variables, constrain many bits of a contributing assignment; an inspection reveals how many (and which) bits are so constrained. If any bit is constrained in conflicting ways (for example, $X_i = 0$, and $Y_c = 0$ for some clause c containing the literal $\bar{x}_i$), then no assignment is contributing (either $a_i = 1$ and the X part becomes zero due to $X_i^{a_i}$, or $a_i = 0$ and the Y part becomes zero due to Y_c). Otherwise, some bits of a potentially contributing assignment are constrained by X and Y, and the remaining bits can be set in any way. Hence the total sum is precisely $2^{(\#\text{ unconstrained bits})} \bmod p$.

Now assume $\mathsf{Sat^q}$ is VNP-hard. Let L be any language in $\mathsf{Mod}_p\mathsf{P}$, witnessed via #P-function f. (That is, $x \in L \iff f(x) \equiv 1 \bmod p$.) By the results of [6, 4], there exists a p-family $r = (r_n) \in \mathsf{VNP}_{\mathbb{F}_p}$ such that $\forall n$, $\forall x \in \{0,1\}^n$, $r_n(x) = f(x) \bmod p$. By assumption, there is a c-reduction from r to $\mathsf{Sat^q}$. We use the oracle circuits from this reduction to decide instances of L. On input x, the advice is the circuit C of appropriate size reducing r to $\mathsf{Sat^q}$. We evaluate this circuit bottom-up. At the leaves, the values are known. At $+$ and $\times$ gates, we perform these operations in $\mathbb{F}_q$. At an oracle gate, the paragraph above tells us how to evaluate the gate. So the circuit can be evaluated in polynomial time, showing that L is in $\mathsf{P/poly}$. Thus $\mathsf{Mod}_p\mathsf{P} \subseteq \mathsf{P/poly}$.

For the other four families, it suffices to show the following, since the rest is identical as for $\mathsf{Sat^q}$.

H'. The monomials of h encode solutions to a problem that is #P-hard via parsimonious reductions.

E'. Over $\mathbb{F}_q$, h can be evaluated in P.

We describe this for the polynomial families one by one.

The *vertex cover* polynomial $\mathsf{VC^q} = (\mathsf{VC^q}_n)$:

$$\mathsf{VC^q}_n := \sum_{S \subseteq V_n} \left(\prod_{e \in E_n :\, e \text{ is incident on } S} X_e^{q-1} \right) \left(\prod_{v \in S} Y_v^{q-1} \right).$$

(H'): Given an instance of vertex cover $A = (V(A), E(A))$ such that $|V(A)| = n$ and $|E(A)| = m$, we show how $\mathsf{VC^q}_n$ encodes the number of solutions of instance A. Consider the following projection of $\mathsf{VC^q}_n$. Set $Y_v = t$, for $v \in V(A)$. For $e \in E(A)$, set $X_e = z$; otherwise $e \notin E(A)$ and set $X_e = 1$. Thus, we have

$$\mathsf{VC^q}_n(z,t) = \sum_{S \subseteq V_n} z^{(\#\text{ edges incident on } S)(q-1)} t^{|S|(q-1)}.$$

Hence, it follows that the number of vertex cover of size k, modulo p, is the coefficient of $z^{m(q-1)}t^{k(q-1)}$ in $\mathsf{VC^q}_n(z,t)$.
(E'): Consider the weighted graph given by the values of $\tilde{X}$ and $\tilde{Y}$ variables. Each subset $S \subseteq V_n$ contributes 0 or 1 to the total. A subset $S \subseteq V_n$ contributes 1 to $\mathsf{VC^q}_n$ if and only if every vertex in S has non-zero weight, and every edge incident on each vertex in S has non-zero weight. That is, S is a subset of full-degree vertices. Therefore, the total sum is $2^{(\#\text{ full-degree vertices})} \bmod p$.

The *clique/independent set* polynomial $\mathsf{CIS^q} = (\mathsf{CIS^q}_n)$:

$$\mathsf{CIS^q}_n := \sum_{T \subseteq E_n} \left(\prod_{e \in T} X_e^{q-1} \right) \left(\prod_{v \text{ incident on } T} Y_v^{q-1} \right).$$

(H'): Given an instance of clique $A = (V(A), E(A))$ such that $|V(A)| = n$ and $|E(A)| = m$, we show how $\mathsf{CIS^q}_n$ encodes the number of solutions of instance A. Consider the following projection of $\mathsf{CIS^q}_n$. Set $Y_v = t$, for $v \in V(A)$. For $e \in E(A)$, set $X_e = z$; otherwise $e \notin E(A)$ and set $X_e = 1$. (This is the same projection as used for vertex cover.) Thus, we have

$$\mathsf{CIS^q}_n(z,t) = \sum_{T \subseteq E_n} z^{|T \cap E(A)|(q-1)} t^{(\#\text{ vertices incident on } T)(q-1)}.$$

Now it follows easily that the number of cliques of size k, modulo p, is the coefficient of $z^{\binom{k}{2}(q-1)}t^{k(q-1)}$ in $\mathsf{CIS^q}_n(z,t)$.
(E'): Consider the weighted graph given by the values of $\tilde{X}$ and $\tilde{Y}$ variables. Each subset $T \subseteq E_n$ contributes 0 or 1 to the sum. A subset $T \subseteq E_n$ contributes 1 to the sum if and only if all edges in T have non-zero weight, and every vertex incident on T must have non-zero weight. Therefore, we consider the graph induced on vertices with non-zero weights. Any subset of edges in this induced graph contributes 1 to the total sum; all other subsets contribute 0. Let ℓ be the number of edges in the induced graph with non-zero weights. Thus, the total sum is $2^\ell \bmod p$.

The *clow* polynomial $\mathsf{Clow^q} = (\mathsf{Clow^q}_n)$: A clow in an n-vertex graph is a closed walk of length exactly n, in which the minimum numbered vertex (called the head) appears exactly once.

$$\mathsf{Clow^q}_n := \sum_{w:\text{ clow of length } n} \left(\prod_{e:\text{ edges in } w} X_e^{q-1} \right) \left(\prod_{\substack{v:\text{ vertices in } w \\ \text{(counted only once)}}} Y_v^{q-1} \right).$$

(If an edge e is used k times in a clow, it contributes $X_e^{k(q-1)}$ to the monomial.)
(H'): Given an instance $A = (V(A), E(A))$ of the Hamiltonian cycle problem with $|V(A)| = n$ and $|E(A)| = m$, we show how $\mathsf{Clow^q}_n$ encodes the number of Hamiltonian cycles in A. Consider the following projection of $\mathsf{Clow^q}_n$. Set $Y_v = t$, for $v \in V(A)$. For $e \in E(A)$, set $X_e = z$; otherwise $e \notin E(A)$ and set $X_e = 1$. (The same projection was used for $\mathsf{VC^q}$ and $\mathsf{CIS^q}$.) Thus, we have

$$\mathsf{Clow^q}_n(z,t) = \sum_{w\text{: clow of length } n} \left(\prod_{e\text{: edges in } w \cap E(A)} z^{q-1} \right) \left(\prod_{\substack{v\text{: vertices in } w \\ \text{(counted only once)}}} t^{q-1} \right).$$

From the definition, it now follows that number of Hamiltonian cycles in A, modulo p, is the coefficient of $z^{n(q-1)}t^{n(q-1)}$.
(E'): To evaluate $\mathsf{Clow^q}_n$ on instantiations of $\tilde{X}$ and $\tilde{Y}$ variables, we consider the weighted graph given by the values to the variables. We modify the edge weights as follows: if an edge is incident on a node with zero weight, we make its weight 0 irrespective of the value of the corresponding X variable. Thus, all zero weight vertices are isolated in the modified graph G. Hence, the total sum is equal to the number of closed walks of length n, modulo p, in this modified graph. This can be computed in polynomial time using matrix powering as follows: Let G_i denote the induced subgraph of G with vertices $\{i, \ldots, n\}$, and let A_i be its adjacency matrix. We represent A_i as an $n \times n$ matrix with the first $i-1$ rows and columns having only zeroes. Now the number of clows with head i is given by the $[i,i]$ entry of $A_i A_{i+1}^{n-2} A_i$.

The *3D-matching* polynomial $\mathsf{3DM^q} = (\mathsf{3DM^q}_n)$: Consider the complete tripartite hyper-graph, where each partition contain n nodes, and each hyperedge has exactly one node from each part. As before, there are variables X_e for hyperedge e and Y_v for node v.

$$\mathsf{3DM^q}_n := \sum_{M \subseteq A_n \times B_n \times C_n} \left(\prod_{e \in M} X_e^{q-1} \right) \left(\prod_{\substack{v \in M \\ \text{(counted only once)}}} Y_v^{q-1} \right).$$

(H'): Given an instance of 3D-Matching $\mathcal{H}$, we consider the usual projection. The variables corresponding to the vertices are all set to t. The edges present in $\mathcal{H}$ are all set to z, and the ones not present are set to 1. Then the number of 3D-matchings in $\mathcal{H}$, modulo p, is equal to the coefficient of $z^{n(q-1)}t^{3n(q-1)}$ in $\mathsf{3DM^q}_n(z,t)$.
(E'): To evaluate $\mathsf{3DM^q}_n$ over $\mathbb{F}_q$, consider the hypergraph obtained after removing the vertices with zero weight, edges with zero weight, and edges that contain a vertex with zero weight (even if the edges themselves have non-zero weight). Every subset of hyperedges in this modified hypergraph contributes 1 to the total sum, and all other subsets contribute 0. Hence, the evaluation equals $2^{(\#\text{ edges in the modified hypergraph})} \bmod p$. □

It is worth noting that the cut enumerator polynomial $\mathsf{Cut^q}$, showed by Bürgisser to be VNP-intermediate over field $\mathbb{F}_q$, is in fact VNP-complete over the rationals when $q = 2$, [9]. Thus the above technique is specific to finite fields.

4 Monotone projection lower bounds

Consider the following polynomial families, defined over an $n \times n$ symbolic matrix,

$$\mathsf{Clique}_n := \sum_{\substack{S \subseteq [n] \\ |S| = \sqrt{n}}} \prod_{\substack{i,j \in S \\ i<j}} x_{i,j}, \ \mathsf{HC}_n := \sum_{\substack{\sigma \in S_n \\ \sigma \text{ is a } n\text{-cycle}}} \prod_{i=1}^{n} x_{i,\sigma(i)}, \ \text{ and } \ \mathsf{Perm}_n := \sum_{\sigma \in S_n} \prod_{i=1}^{n} x_{i,\sigma(i)}.$$

Over the Boolean $\{\wedge, \vee\}$-semi-ring, it is known that $\mathsf{Clique} = (\mathsf{Clique}_n)$ is a monotone p-projection of $\mathsf{HC} = (\mathsf{HC}_n)$ [29]. In fact, Clique_n is a monotone projection of HC_{25n^2} [1]. Jukna [19] observed if it were the case that HC is a monotone p-projection of Perm, then using the $2^{n^{\Omega(1)}}$ lower bound of Alon and Boppana [1] for Clique_n, we would get a lower bound of $2^{n^{\Omega(1)}}$ for Perm_n.

Grochow [15] answered the above question in the negative by showing that the Hamiltonian cycle family HC is not a monotone sub-exponential-size projection of the permanent. Thus, he established that monotone circuit lower bounds for Clique *cannot* be transferred to the permanent *via* the Hamiltonian cycle HC. However, it still leaves open the possibility of transfer via, say, '*satisfiability*' [29]. It is known that Clique, over the Boolean $\{\wedge, \vee\}$-semi-ring, is a monotone polynomial-size projection of satisfiability (see Section 5 [1]).

Here we extend Grochow's arguments to show that Clique itself is not a *monotone p-projection* of Perm. Thus this possibility of transferring monotone circuit lower bounds for clique to permanent cannot work.

Recall that a polynomial $f(x_1, \ldots, x_n)$ is a *projection* of a polynomial $g(y_1, \ldots, y_m)$ if $f(x_1, \ldots, x_n) = g(a_1, \ldots, a_m)$, where a_i's are either constants or x_j for some j. The polynomial f is an *affine* projection of g if f can be obtained from g by replacing each y_i with an affine linear function $\ell_i(\tilde{x})$. Over any subring of $\mathbb{R}$, or more generally any totally ordered semi-ring, a *monotone projection* is a projection in which all constants appearing in the projection are non-negative. We say that the family (f_n) is a (monotone affine) projection of the family (g_n) with *blow-up* $t(n)$ if for all sufficiently large n, f_n is a (monotone affine) projection of $g_{t(n)}$.

Theorem 2. *Over the reals (or any totally ordered semi-ring), the* Clique *family is not a monotone affine p-projection of the* Perm *family. Any monotone affine projection from* Perm *to* Clique *must have a blow-up of at least* $2^{\Omega(\sqrt{n})}$.

Before giving the proof, we set up some notation. For more details, see [2, 27, 15]. For any polynomial p in n variables, let $\mathsf{Newt}(p)$ denote the polytope in $\mathbb{R}^n$ that is the convex hull of the vectors of exponents of monomials of p. The *correlation polytope* $\mathsf{COR}(n)$ is defined as the convex hull of $n \times n$ binary symmetric matrices of rank 1. That is, $\mathsf{COR}(n) := \mathrm{conv}\{vv^{\mathrm{t}} \mid v \in \{0,1\}^n\}$.

For a polytope P, let $\mathsf{c}(P)$ denote the minimal number of linear inequalities needed to define P. A polytope $Q \subseteq \mathbb{R}^m$ is an *extension* of $P \subseteq \mathbb{R}^n$ if there is an affine linear map $\pi \colon \mathbb{R}^m \to \mathbb{R}^n$ such that $\pi(Q) = P$. The *extension complexity* of P, denoted $\mathsf{xc}(P)$, is the minimum size $c(Q)$ of any extension Q (of any dimension) of P.

The following are straightforward, see for instance [15, 13].

Fact 3 *1. [15]*$\mathsf{c}(\mathsf{Newt}(\mathsf{Perm}_n)) \leqslant 2n$.
2. [13] If polytope Q is an extension of polytope P, then $xc(P) \leqslant \mathsf{xc}(Q)$.

We use the following recent results.

Lemma 1 ([15]). *Let $f(x_1, \dots, x_n)$ and $g(y_1, \dots, y_m)$ be polynomials over a totally ordered semi-ring R, with non-negative coefficients. If f is a monotone projection of g, then the intersection of $\mathsf{Newt}(g)$ with some linear subspace is an extension of $\mathsf{Newt}(f)$. In particular, $\mathsf{xc}(\mathsf{Newt}(f)) \leqslant m + \mathsf{c}(\mathsf{Newt}(g))$.*

Theorem 4 ([13]). *There exists a constant $C > 0$ such that for all n, $\mathsf{xc}(\mathsf{COR}(n)) \geqslant 2^{Cn}$.*

We now show that Clique_n is **not** a monotone p-projection of Perm_n. To establish this we will consider a different family $\mathsf{Clique}^* = (\mathsf{Clique}^*{}_n)$ that counts all cliques in a graph. More formally,

$$\mathsf{Clique}^*{}_n := \sum_{S \subseteq [n]} \prod_{i \in S} x_{i,i} \prod_{\substack{i,j \in S \\ i<j}} x_{i,j}.$$

We first claim that proving monotone projection lower bound against Clique^* suffices to establish lower bound against Clique. The proof is basically the VNP-completeness proof of Clique_n (see [17]).

Lemma 2. *The family Clique^* is a monotone p-projection of the family Clique. In particular, $\mathsf{Clique}^*{}_n$ is a monotone projection of $\mathsf{Clique}_{(n+1)^2}$.*

Proof. Let G_n be a complete undirected graph on n vertices $\{v_1, \dots, v_n\}$ with edge weights $x_{i,j}$ on the edge (v_i, v_j). Let G'_n be a complete undirected graph on the vertex set $\{v'_1, \dots, v'_n\}$ with every edge having weight 1. We also add the following set $\{(v_i, v'_j) \mid i \neq j\}$ of cross edges between G_n and G'_n. The edges in this set also have weight 1. To this graph we add a new vertex u such that it is adjacent to every vertex in $G_n \cup G'_n$. The edges adjacent to vertices in G'_n have weight 1. For the vertices in G_n the weight of the edge (u, v_i) is $x_{i,i}$. We call this graph, on $2n+1$ vertices, H_n. We claim that there is a one-to-one correspondence between cliques in G_n (of all sizes) and $(n+1)$-sized cliques in H_n. Let $S \subseteq \{v_1, \dots, v_n\}$ be a subset of vertices such that they form a clique in G_n. Consider the following map which is easily seen to be bijective. Map S to the clique on the following set of vertices in H_n: $S \cup \{v'_j \mid j \notin S\} \cup \{u\}$. Since 0 and 1 are the only constants used in the projection, it is also a monotone projection.

To obtain the lemma we add n^2 isolated vertices to H_n.

Theorem 5. *Over the reals (or any totally ordered semi-ring), the family Clique^* is not monotone affine p-projections of the Perm family. In fact, if $\mathsf{Clique}^*{}_n$ is a monotone affine projection of $\mathsf{Perm}_{t(n)}$, then $t(n) \geqslant 2^{\Omega(n)}$.*

Proof. Let Q be the Newton polytope of $\mathsf{Clique}^*{}_n$. It resides in $N := \binom{n}{2} + n$ dimensions. Furthermore, it is the convex hull of vectors of the form $\langle \tilde{a}, \tilde{b} \rangle$ where $\tilde{a} \in \{0,1\}^{\binom{n}{2}}$ is the characteristic vector of the set of edges of the clique over the set of vertices given by $\tilde{b} \in \{0,1\}^n$, in the complete undirected graph K_n. We will index a vector in N dimensions by pairs (i,j) such that $1 \leqslant i \leqslant j \leqslant n$.

Let us now consider the linear map $\ell\colon \mathbb{R}^N \to \mathbb{R}^{n\times n}$, such that $\ell(A) := B$, and for $1 \leqslant i \leqslant j \leqslant n$, $B_{i,j} = B_{j,i} = A_{(i,j)}$. We now claim that under the map ℓ, Q is mapped to the correlation polytope $\mathsf{COR}(n)$. It suffices to show that vertices of Q under the map ℓ are mapped into $\mathsf{COR}(n)$, and every vertex of $\mathsf{COR}(n)$ has a pre-image in Q under ℓ. Indeed ℓ maps the vertices of Q to the vertices of $\mathsf{COR}(n)$ bijectively. It follows from the map that a vertex $\langle \tilde{a}, \tilde{b}\rangle$ of Q is mapped to the vertex $\tilde{b}\tilde{b}^{\mathrm{t}}$ of $\mathsf{COR}(n)$. Furthermore, the pre-image of a vertex $\tilde{b}\tilde{b}^{\mathrm{t}}$ of $\mathsf{COR}(n)$ is the clique given by the upper-triangular and diagonal entries of $\tilde{b}\tilde{b}^{\mathrm{t}}$. Thus Q is an extension of $\mathsf{COR}(n)$, so by Fact 3 (2), $\mathsf{xc}(\mathsf{COR}(n)) \leqslant \mathsf{xc}(Q)$.

Suppose $\mathsf{Clique^*}_n$ is a monotone projection of $\mathsf{Perm}_{t(n)}$. By Fact 3 (1) and Lemma 1, $\mathsf{xc}(\mathsf{Newt}(\mathsf{Clique^*}_n)) = \mathsf{xc}(Q) \leqslant t(n)^2 + c(\mathsf{Perm}_{t(n)}) \leqslant O(t(n)^2)$. From the preceding discussion and Theorem 4, we get $2^{\Omega(n)} \leqslant \mathsf{xc}(\mathsf{COR}(n)) \leqslant \mathsf{xc}(Q) \leqslant O(t(n)^2)$. Therefore, it follows that $t(n)$ is at least $2^{\Omega(n)}$.

Proof. (of Theorem 2.) Suppose Clique_n is a monotone projection of $\mathsf{Perm}_{t(n)}$. From Lemma 2, it follows that $\mathsf{Clique^*}_n$ is a monotone projection of $\mathsf{Perm}_{t((n+1)^2)}$. Hence, from Theorem 5 we get $t(n^2) \geqslant 2^{\Omega(n)}$. Thus, $t(n) \geqslant 2^{\Omega(\sqrt{n})}$.

Using similar arguments, we now show that Perm also fails to express two of our intermediate families of polynomials, $\mathsf{Sat^q}$ and $\mathsf{Clow^q}$, via monotone affine projections.

Theorem 6. *Over the reals (or any totally ordered semi-ring), for any q, the families $\mathsf{Sat^q}$ and $\mathsf{Clow^q}$ are not monotone affine p-projections of the* Permanent *family. Any monotone affine projection from* Permanent *to $\mathsf{Sat^q}$ must have a blow-up of at least $2^{\Omega(\sqrt{n})}$. Any monotone affine projection from* Permanent *to $\mathsf{Clow^q}$ must have a blow-up of at least $2^{\Omega(n)}$.*

First we set up the required notation and state known results. For any Boolean formula ϕ on n variables, let $\mathsf{p\text{-}SAT}(\phi)$ denote the polytope in $\mathbb{R}^n$ that is the convex hull of all satisfying assignments of ϕ. Let $K_n = (V_n, E_n)$ denote the n-vertex complete graph. The travelling salesperson (TSP) polytope is defined as the convex hull of the characteristic vectors of all subsets of E_n that define a Hamiltonian cycle in K_n.

We use the following recent results.

Proposition 1. *1. For every n there exists a 3SAT formula ϕ with $O(n)$ variables and $O(n)$ clauses such that $\mathsf{xc}(\mathsf{p\text{-}SAT}(\phi)) \geqslant 2^{\Omega(\sqrt{n})}$. [2]*
2. The extension complexity of the TSP polytope is $2^{\Omega(n)}$. [27]

Proof. (of Theorem 6.) Let ϕ be a 3SAT formula with n variables and m clauses as given by Proposition 1 (1). For the polytope $P = \mathsf{p\text{-}SAT}(\phi)$, $\mathsf{xc}(P)$ is high.

Let Q be the Newton polytope of $\mathsf{Sat^q}_n$. It resides in N dimensions, where $N = n + |\mathsf{Cl}_n| = n + 8n^3$, and is the convex hull of vectors of the form $(q-1)\langle \tilde{a}\tilde{b}\rangle$ where $\tilde{a} \in \{0,1\}^n$,

$\tilde{b} \in \{0,1\}^{N-n}$, and for all $c \in \mathsf{Cl}_n$, $\tilde{a}$ satisfies c if and only if $b_c = 1$. For each $\tilde{a} \in \{0,1\}^n$, there is a unique $\tilde{b} \in \{0,1\}^{N-n}$ such that $(q-1)\langle\tilde{a}\tilde{b}\rangle$ is in Q.

Define the polytope R, also in N dimensions, to be the convex hull of vectors that are vertices of Q and also satisfy the constraint $\sum_{c\in\phi} b_c \geq m$. This constraint discards vertices of Q where $\tilde{a}$ does not satisfy ϕ. Thus R is an extension of P (projecting the first n coordinates of points in R gives a $(q-1)$-scaled version of P), so by Fact 3 (2), $\mathsf{xc}(P) \leq \mathsf{xc}(R)$. Further, we can obtain an extension of R from any extension of Q by adding just one inequality; hence $\mathsf{xc}(R) \leq 1 + \mathsf{xc}(Q)$.

Suppose $\mathsf{Sat}^{\mathsf{q}}$ is a monotone affine projection of Perm_n with blow-up $t(n)$. By Fact 3 (1) and Lemma 1, $\mathsf{xc}(\mathsf{Newt}(\mathsf{Sat}^{\mathsf{q}})) = \mathsf{xc}(Q) \leq t(n)^2 + c(\mathsf{Perm}_{t(n)}) \leq O(t(n)^2)$. From the preceding discussion and by Proposition 1 (1), we get $2^{\Omega(\sqrt{n})} \leq \mathsf{xc}(P) \leq \mathsf{xc}(R) \leq 1+\mathsf{xc}(Q) \leq O(t(n)^2)$. It follows that $t(n)$ is at least $2^{\Omega(\sqrt{n})}$.

For the $\mathsf{Clow}^{\mathsf{q}}$ polynomial, let P be the TSP polytope and Q be $\mathsf{Newt}(\mathsf{Clow}^{\mathsf{q}})$. The vertices of Q are of the form $(q-1)\tilde{a}\tilde{b}$ where $\tilde{a} \in \{0,1\}^{\binom{n}{2}}$ picks a subset of edges, $\tilde{b} \in \{0,1\}^n$ picks a subset of vertices, and the picked edges form a length-n clow touching exactly the picked vertices. Define polytope R by discarding vertices of Q where $\sum_{i\in[n]} b_i < n$. Now the same argument as above works, using Proposition 1 (2) instead of (1). $\square$

5 Complete families for algebraic classes

The quest for a natural VP-complete polynomial has generated a significant amount of research [4, 24, 23, 7, 11]. The first success story came from [11], where some naturally defined homomorphism polynomials were studied, and a host of them were shown to be complete for the class VP. But the results came with minor caveats. When the completeness was established under projections, there were non-trivial restrictions on the set of homomorphisms $\mathcal{H}$, and sometimes even on the target graph H. On the other hand, when all homomorphisms were allowed, completeness could only be shown under seemingly more powerful reductions, namely, constant-depth c-reductions. Furthermore, the graphs were either directed or had weights on nodes. It is worth noting that the reductions in [11] actually do not use the full power of generic constant-depth c-reductions; a closer analysis reveals that they are in fact *linear p-projection*. That is, the reductions are linear combinations of polynomially many p-projections (see Chapter 3, [4]). Still, this falls short of p-projections.

In this work, we remove all such restrictions and show that there is a simple explicit homomorphism polynomial family that is complete for VP under p-projections. In this family, the source graphs G are specific bounded-tree-width graphs, and the target graphs H are complete graphs. We also show that a similar family with bounded-path-width source graphs is complete for VBP under p-projections. Thus, homomorphism polynomials are rich enough to characterise computations by circuits as well as algebraic branching programs. In fact, we complement the picture by showing that if we allow tree-width of source graphs to grow linearly in n, we capture VNP as well.

The polynomials we consider are defined formally as follows.

Definition 7 *Let $G = (V(G), E(G))$ and $H = (V(H), E(H))$ be two graphs. Consider the set of variables $\bar{Z} := \{Z_{u,a} \mid u \in V(G) \text{ and } a \in V(H)\}$ and $\bar{Y} := \{Y_{(u,v)} \mid (u,v) \in E(H)\}$. Let $\mathcal{H}$ be a set of homomorphisms from G to H. The homomorphism polynomial $f_{G,H,\mathcal{H}}$ in the variable set $\bar{Y}$, and the generalised homomorphism polynomial $\hat{f}_{G,H,\mathcal{H}}$ in the variable set $\bar{Z} \cup \bar{Y}$, are defined as follows:*

$$f_{G,H,\mathcal{H}} = \sum_{\phi \in \mathcal{H}} \left(\prod_{(u,v) \in E(G)} Y_{(\phi(u),\phi(v))} \right).$$

$$\hat{f}_{G,H,\mathcal{H}} = \sum_{\phi \in \mathcal{H}} \left(\prod_{u \in V(G)} Z_{u,\phi(u)} \right) \left(\prod_{(u,v) \in E(G)} Y_{(\phi(u),\phi(v))} \right).$$

Let **Hom** *denote the set of all homomorphisms from G to H. If $\mathcal{H}$ equals* **Hom***, then we drop it from the subscript and write $f_{G,H}$ or $\hat{f}_{G,H}$.*

Note that for every $G, H, \mathcal{H}$, $f_{G,H,\mathcal{H}}(\bar{Y})$ equals $\hat{f}_{G,H,\mathcal{H}}(\bar{Y}) \mid_{\bar{Z}=\bar{1}}$. Thus upper bounds for $\hat{f}$ give upper bounds for f, while lower bounds for f give lower bounds for $\hat{f}$. The following observation follows straightforwardly from Valiant's Criterion [29] (See also Proposition 2.20 in [4]).

Proposition 2. *Let (G_n) and (H_n) be p-families of graphs. Consider the family of homomorphism polynomials (f_n), where $f_n = \hat{f}_{G_n,H_n}(\bar{Z}, \bar{Y})$. Then, $(f_n) \in \mathsf{VNP}$.*

We show in Theorem 8 that for any p-family (H_m), and any bounded tree-width (path-width, respectively) p-family (G_m), the polynomial family (f_m) where $f_m = \hat{f}_{G_m,H_m}$ is in VP (VBP, respectively). We then show in Theorem 9 that for a specific bounded tree-width family (G_m), and for $H_m = K_{m^6}$, the polynomial family (f_{G_m,H_m}) is hard, and hence complete, for VP with respect to projections. An analogous statement is shown in Theorem 10 for a specific bounded path-width family (G_m) and for $H_m = K_{m^2}$. Over fields of characteristic other than 2, VBP-hardness is obtained for a simpler family of source graphs G_m, as described in Theorem 11. A family of homomorphism polynomials complete for VNP is established in Theorem 12.

5.1 Upper Bound

In [11], it was shown that the homomorphism polynomial $\hat{f}_{T_m,K_n}$ where T_m is a binary tree on m leaves, and K_n is a complete graph on n nodes, is computable by an arithmetic circuit of size $O(m^3n^3)$. Their proof idea is based on recursion: group the homomorphisms based on where they map the root of T_m and its children, and recursively compute the sub-polynomials within each group. The sub-polynomials of a specific group have a special set of variables in their monomials. Hence, the homomorphism polynomial can be computed by suitably combining partial derivatives of the sub-polynomials. The partial derivatives themselves can be computed efficiently using the technique of Baur and Strassen, [3].

Generalizing the above idea to polynomials where the source graph is not a binary tree T_m but a bounded tree-width graph G_m seems hard. The very first obstacle we encounter is to generalize the concept of partial derivative to monomial extension. Combining sub-polynomials to obtain the original polynomial also gets rather complicated.

We sidestep this difficulty by using a dynamic programming approach [10] based on a "nice" tree decomposition of the source graph. This shows that the homomorphism polynomial $\hat{f}_{G,H}$ is computable by an arithmetic circuit of size at most $2|V(G)| \cdot |V(H)|^{tw(G)+1} \cdot (2|V(H)| + 2|E(H)|)$, where $tw(G)$ is the tree-width of G.

Let $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ be a nice tree decomposition of G of width τ. For each $t \in V(T)$, let $M_t = \{\phi \mid \phi\colon X_t \to V(H)\}$ be the set of all mappings from X_t to $V(H)$. Since $|X_t| \leqslant \tau+1$, we have $|M_t| \leqslant |V(H)|^{\tau+1}$. For each node $t \in V(T)$, let T_t be the subtree of T rooted at node t, $V_t := \bigcup_{t' \in V(T_t)} X_{t'}$, and $G_t := G[V_t]$ be the subgraph of G induced on V_t. Note that $G_r = G$.

We will build the circuit inductively. For each $t \in V(T)$ and $\phi \in M_t$, we have a gate $\langle t, \phi \rangle$ in the circuit. Such a gate will compute the homomorphism polynomial from G_t to H such that the mapping of X_t in H is given by ϕ. For each such gate $\langle t, \phi \rangle$ we introduce another gate $\langle t, \phi \rangle'$ which computes the "partial derivative" (or, quotient) of the polynomial computed at $\langle t, \phi \rangle$ with respect to the monomial given by ϕ. As we mentioned before, the construction is inductive, starting at the leaf nodes and proceeding towards the root.

Base case (Leaf nodes): Let $\ell \in V(T)$ be a leaf node. Then, $X_\ell = \{u\}$ such that $u \in V(G)$. Note that any $\phi \in M_\ell$ is just a mapping of u to some node in $V(H)$. Hence, the set M_ℓ can be identified with $V(H)$. Therefore, for all $h \in V(H)$, we label the gate $\langle \ell, h \rangle$ by the variable $Z_{u,h}$. The derivative gate $\langle \ell, h \rangle'$ in this case is set to 1.

Introduce nodes: Let $t \in V(T)$ be an introduce node, and t' be its unique child. Then, $X_t \setminus X_{t'} = \{u\}$ for some $u \in V(G)$. Let $N(u) := \{v | v \in X_{t'} \text{ and } (v, u) \in E(G_t)\}$. Note that there is a one-to-one correspondence between $\phi \in M_t$ and pairs $(\phi', h) \in M_{t'} \times V(H)$. Therefore, for all $\phi (= (\phi', h)) \in M_t$ such that $\forall v \in N(u), (\phi'(v), h) \in E(H)$, we set

$$\langle t, \phi \rangle := Z_{u,h} \cdot \left(\prod_{v \in N(u)} Y_{(\phi'(v),h)} \right) \cdot \langle t', \phi' \rangle \qquad \text{and,}$$
$$\langle t, \phi \rangle' := \langle t', \phi' \rangle',$$

otherwise we set $\langle t, \phi \rangle = \langle t, \phi \rangle' := 0$.

Forget nodes: Let $t \in V(T)$ be a forget node and t' be its unique child. Then, $X_{t'} \setminus X_t = \{u\}$ for some $u \in V(G)$. Again note that there is a one-to-one correspondence between pairs $(\phi, h) \in M_t \times V(H)$ and $\phi' \in M_{t'}$. Let $N(u) := \{v | v \in X_t \text{ and } (v, u) \in E(G_{t'})\}$. Therefore, for all $\phi \in M_t$, we set

$$\langle t, \phi \rangle := \sum_{h \in V(H)} \langle t', (\phi, h) \rangle \qquad \text{and,}$$

$$\langle t, \phi \rangle' := \sum_{\substack{h \in V(H) \text{ such that} \\ \forall v \in N(u), (\phi(v), h) \in E(H)}} Z_{u,h} \cdot \left(\prod_{v \in N(u)} Y_{(\phi(v),h)} \right) \cdot \langle t', (\phi, h) \rangle'.$$

Join nodes: Let $t \in V(T)$ be a join node, and t_1 and t_2 be its two children; we have $X_t = X_{t_1} = X_{t_2}$. Then, for all $\phi \in M_t$, we set

$$\langle t, \phi \rangle := \langle t_1, \phi \rangle \cdot \langle t_2, \phi \rangle' \, (= \langle t_1, \phi \rangle' \cdot \langle t_2, \phi \rangle)$$
$$\langle t, \phi \rangle' := \langle t_1, \phi \rangle' \cdot \langle t_2, \phi \rangle'.$$

The output gate of the circuit is $\langle r, \emptyset \rangle$. The correctness of the algorithm is readily seen via induction in a similar way. The bound on the size also follows easily from the construction.

We observe some properties of our construction. First, the circuit constructed is a constant-free circuit. This was the case with the algorithm from [11] too. Second, if we start with a path decomposition, we obtain *skew* circuits, since the *join* nodes are absent. The algorithm from [11] does not give skew circuits when T_m is a path. (It seems the obstacle there lies in computing partial-derivatives using skew circuits.)

From the above algorithm and its properties, we obtain the following theorem.

Theorem 8. *Consider the family of homomorphism polynomials* (f_m), *where* $f_m = f_{G_m,H_m}(\bar{Z}, \bar{Y})$, *and* (H_m) *is a p-family of complete graphs.*

- *If* (G_m) *is a p-family of graphs of bounded tree-width, then* $(f_m) \in \mathsf{VP}$.
- *If* (G_m) *is a p-family of graphs of bounded path-width, then* $(f_m) \in \mathsf{VBP}$.

5.2 VP-completeness

We now turn our attention towards establishing VP-*hardness* of the homomorphism polynomials. We need to show that there exists a p-family (G_m) of bounded tree-width graphs such that $(f_{G_m,H_m}(\bar{Y}))$ is hard for VP under projections.

We use *rigid* and mutually *incomparable* graphs in the construction of G_m. Let $I := \{I_0, I_1, I_2\}$ be a fixed set of three connected, rigid and mutually incomparable graphs. Note that they are necessarily *non-bipartite*. Let $c_{I_i} = |V(I_i)|$. Choose an integer $c_{\max} > \max\{c_{I_0}, c_{I_1}, c_{I_2}\}$. Identify two distinct vertices $\{v_\ell^0, v_r^0\}$ in I_0, three distinct vertices $\{v_\ell^1, v_r^1, v_p^1\}$ in I_1, and three distinct vertices $\{v_\ell^2, v_r^2, v_p^2\}$ in I_2.

For every m a power of 2, we denote a complete (perfect) binary tree with m leaves by T_m. We construct a sequence of graphs G_m (Fig. 2) from T_m as follows: first replace the root by the graph I_0, then all the nodes on a particular level are replaced by either I_1 or I_2 alternately (cf. Fig. 2). Now we add edges; suppose we are at a 'node' which is labeled I_i and the left child and right child are labeled I_j, we add an edge between v_ℓ^i and v_p^j in the left child, and an edge between v_r^i and v_p^j in the right child. Finally, to obtain G_m we expand each added edge into a simple path with $c_{\max}$ vertices on it (cf. Fig. 2). That is, a left-edge connection between two incomparable graphs in the tree looks like, $I_i(v_\ell^i)$ − (path with $c_{\max}$ vertices) − $(v_p^j)I_j$.

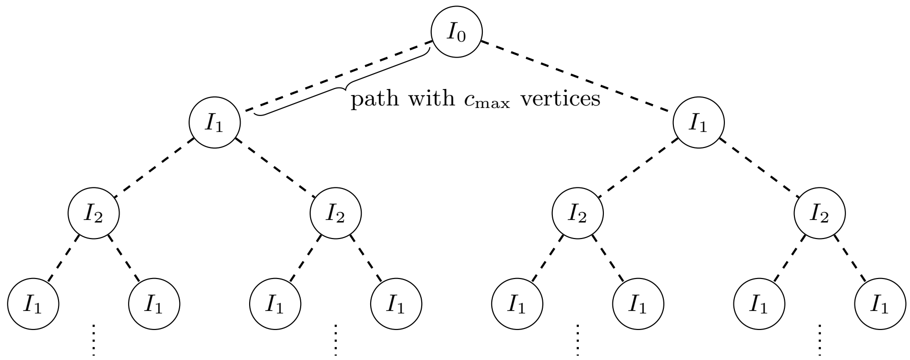

Fig. 2. The graph G_m.

Theorem 9. *Over any field, the family of homomorphism polynomials* (f_m)*, with* $f_m(\bar{Y}) = f_{G_m,H_m}(\bar{Y})$*, where*

- G_m *is defined as above (see Fig. 2), and*
- H_m *is an undirected complete graph on* $\mathsf{poly}(m)$*, say* m^6*, vertices,*

is complete for VP *under p-projections.*

Proof. Membership in VP follows from Theorem 8.

We proceed with the *hardness* proof. The idea is to obtain the VP-complete universal polynomial from [24] as a projection of f_m. This universal polynomial is computed by a normal-form homogeneous circuit with alternating unbounded fanin-in + and bounded fan-in × gates. We would like to put its parse trees in bijection with homomorphisms from G to H. This becomes easier if we use an equivalent universal circuit in a nice normal form as described in [11]. The normal form circuit is *multiplicatively disjoint*; sub-circuits of × gates are disjoint (see [22]). This ensures that even though C_n itself is not a formula, all its parse trees are already subgraphs of C_n even without unwinding it into a formula.

Our starting point is the related graph J_n' in [11]. The parse trees in C_n are complete alternating unary-binary trees. The graph J_n' is constructed in such a way that the parse trees are now in bijection with complete binary trees. To achieve this, we "shortcut" the + gates, while preserving information about whether a subtree came in from the left or the right. For completeness sake we describe the construction of J_n' from [11].

We obtain a sequence of graphs (J_n') from the undirected graphs underlying (C_n) as follows. Retain the multiplication and input gates of C_n. Let us make two copies of each. For each retained gate, g, in C_n; let g_L and g_R be the two copies of g in J_n'. We now define the edge connections in J'_n. Assume g is a $\times$ gate retained in J'_n. Let α and β be two $+$ gates feeding into g in C_n. Let $\{\alpha_1, \ldots, \alpha_i\}$ and $\{\beta_1, \ldots, \beta_j\}$ be the gates feeding into α and β, respectively. Assume without loss of generality that α and β feed into g from left and right, respectively. We add the following set of edges to J'_n: $\{(\alpha_{1L}, g_L), \ldots, (\alpha_{iL}, g_L)\}$, $\{(\beta_{1R}, g_L), \ldots, (\beta_{jR}, g_L)\}$, $\{(\alpha_{1L}, g_R), \ldots, (\alpha_{iL}, g_R)\}$ and $\{(\beta_{1R}, g_R), \ldots, (\beta_{jR}, g_R)\}$. We now would like to keep a single copy of C_n in these set of edges. So we remove the vertex $root_R$ and we remove the remaining spurious edges in following way. If we assume that all edges are directed from root towards leaves, then we keep only edges induced by the vertices reachable from $root_L$ in this directed graph. In [11], it was observed that there is a one-to-one correspondence between parse trees of C_n and subgraphs of J'_n that are rooted at $root_L$ and isomorphic to $\mathsf{T}_{2^{k(n)}}$.

We now transform J'_n using the set $I = \{I_0, I_1, I_2\}$. This is similar to the transformation we did to the balanced binary tree T_m. We replace each vertex by a graph in I; $root_L$ gets I_0 and the rest of the layers get I_1 or I_2 alternately (as in Fig. 2). Edge connections are made so that a left/right child is connected to its parent via the edge $(v_p^j, v_\ell^i)/(v_p^j, v_r^i)$. Finally we replace each edge connection by a path with $c_{\max}$ vertices on it (as in Fig. 2), to obtain the graph J_n. All edges of J_n are labeled 1, with the following exceptions: Every input node contains the same rigid graph I_i. It has a vertex v_p^i. Each path connection to other nodes has this vertex as its end point. Label such path edges that are incident on v_p^i by the label of the input gate.

Let $m := 2^{k(n)}$. The choice of $\mathsf{poly}(m)$ is such that $4s_n \leqslant \mathsf{poly}(m)$, where s_n is the size of J_n. The $\bar{Y}$ variables are set to $\{0, 1, \bar{x}\}$ such that the non-zero variables pick out the graph J_n. From the observations of [11] it follows that for each parse tree p-T of C_n, there exists a homomorphism $\phi\colon G_{2^{k(n)}} \to J_n$ such that $mon(\phi)$ is exactly equal to $mon(p\text{-}\mathsf{T})$. By $mon(\cdot)$ we mean the monomial associated with an object. We claim that these are the only valid homomorphisms from $G_{2^{k(n)}} \to J_n$. We observe the following properties of homomorphisms from $G_{2^{k(n)}} \to J_n$, from which the claim follows. In the following by a rigid-node-subgraph we mean a graph in $\{I_0, I_1, I_2\}$ that replaces a vertex.

- *(i)* Any homomorphic image of a rigid-node-subgraph of $G_{2^{k(n)}}$ in J_n, cannot split across two mutually incomparable rigid-node-subgraphs in J_n. That is, there cannot be two vertices in a rigid subgraph of $G_{2^{k(n)}}$ such that one of them is mapped into a rigid subgraph say n_1, and the other one is mapped into another rigid subgraph say n_2. This follows because homomorphisms do not increase distance.
- *(ii)* Because of *(i)*, with each homomorphic image of a rigid node $g_i \in G_{2^{k(n)}}$, we can associate at most one rigid node of J_n, say n_i, such that the homomorphic image of g_i is a subgraph of n_i and the paths (corresponding to incident edges) emanating from it. But such a subgraph has a homomorphism to n_i itself: fold each hanging path into an edge and then map this edge into an edge within n_i. (For instance, let ρ be a path hanging off n_i and attached to n_i at u, and let v be any neighbour of u within n_i. Mapping vertices of ρ to u and v alternately preserves all edges and hence is a homomorphism.) Therefore, we note that in such a case we have a homomorphism from $g_i \to n_i$. By rigidity and mutual incomparability, g_i must be the same as n_i, and this folded-path homomorphism must be the identity map. The other scenario, where we cannot associate any n_i because

g_i is mapped entirely within connecting paths, is not possible since it contradicts *non-bipartiteness* of mutually-incomparable graphs.

Root must be mapped to the root: The rigidity of I_0 and Property (ii) implies that $I_0 \in G_{2^{k(n)}}$ is mapped identically to I_0 in J_n.
Every level must be mapped within the same level: The children of I_0 in $G_{2^{k(n)}}$ are mapped to the children of the root while respecting left-right behaviour. Firstly, the left child cannot be mapped to the root because of incomparability of the graphs I_1 and I_0. Secondly, the left child cannot be mapped to the right child (or vice versa) even though they are the same graphs, because the minimum distance between the vertex in I_0 where the left path emanates and the right child is $c_{\max}+1$ whereas the distance between the vertex in I_0 where the left path emanates and the left child is $c_{\max}$. So some vertex from the left child must be mapped into the path leading to the right child and hence the rest of the left child must be mapped into a proper subgraph of right child. But this contradicts rigidity of I_1. Continuing like this, we can show that every level must map within the same level and that the mapping within a level is correct. □

5.3 VBP-completeness

In this subsection, we show that homomorphism polynomials are also rich enough to characterize computation by algebraic branching programs. Formally, we establish that there exists a p-family (G_k) of undirected *bounded path-width* graphs such that the family $(f_{G_k,H_k}(\bar{Y}))$ is VBP-complete with respect to p-projections.

We note that for VBP-completeness under projections, the construction in [11] required directed graphs. In the undirected setting they could establish hardness only under *linear p-projection*, that too using 0-1 valued weights.

As before, we use rigid and mutually incomparable graphs in the construction of G_k. Let $I := \{I_1, I_2\}$ be two connected, non-bipartite, rigid and mutually incomparable graphs. Arbitrarily pick vertices $u \in V(I_1)$ and $v \in V(I_2)$. Let $c_{I_i} = |V(I_i)|$, and $c_{max} = \max\{c_{I_1}, c_{I_2}\}$. Consider the sequence of graphs G_k (Fig. 3); for every k, there is a simple path with $(k-1)+2c_{max}$ edges between a copy of I_1 and I_2. The path is between the vertices $u \in V(I_1)$ and $v \in V(I_2)$. The path between vertices a and b in G_k contains $(k-1)$ edges.

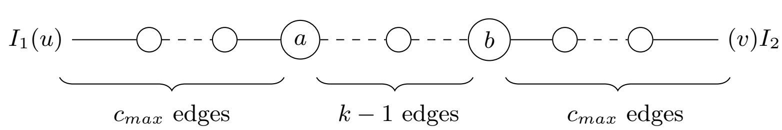

Fig. 3. The graph G_k.

In other words, connect I_1 and I_2 by stringing together a path with c_{max} edges between u and a, a path with $k-1$ edges between a and b, and a path with c_{max} edges between b and v.

Theorem 10. *Over any field, the family of homomorphism polynomials (f_k), where*

– G_k *is defined as above (see Fig. 3),*
– H_k *is the undirected complete graph on* $O(k^2)$ *vertices,*
– $f_k(\bar{Y}) = f_{G_k,H_k}(\bar{Y})$,

is complete for VBP *with respect to p-projections.*

Proof. **Membership:** It follows from Theorem 8.
Hardness: Let $(g_n) \in$ VBP. Without loss of generality, we can assume that g_n is computable by a layered branching program A_n of polynomial size such that the number of layers, ℓ, is more than the width of the algebraic branching program.

Let B'_n be the undirected graph underlying the layered branching program A_n for g_n. Let B_n be the following graph: $I_1(u) - (s)B'_n(t) - (v)I_2$, that is, $u \in I_1$ is connected to $s \in B'_n$ via a path with c_{max} edges and $t \in B'_n$ is connected to $v \in I_2$ via a path with c_{max} edges (cf. Fig. 3). The edges in B'_n inherits the weight from A_n, and the rest of the edges in B_n have weight 1.

Let us now consider f_ℓ when the variables on the edges of H_ℓ are instantiated to values in $\{0, 1\}$ or variables of g_n so that we obtain B_n as a subgraph of H_ℓ. We claim that a valid homomorphism from $\mathrm{G}_\ell \to B_n$ must satisfy the following properties:

(P1) I_1 in G_ℓ must be mapped to I_1 in B_n using the identity homomorphism,
(P2) I_2 in G_ℓ must be mapped to I_2 in B_n using the identity homomorphism.

Assuming the claim, it follows that homomorphisms from $\mathrm{G}_\ell \to B_n$ are in one-to-one correspondence with s-t paths in A_n. In particular, the vertex $a \in \mathrm{G}_\ell$ is mapped to the vertex s in B_n, and the vertex $b \in \mathrm{G}_\ell$ is mapped to the vertex t in B_n. Also, the monomial associated with a homomorphism and its corresponding path are the same. Therefore, we have,

$$f_{\mathrm{G}_\ell,B_n} = g_n.$$

Since ℓ is polynomially bounded, we obtain VBP-completeness of (f_k) over any field.

Let us now prove the claim. We first prove that a valid homomorphism from $\mathrm{G}_\ell \to B_n$ must satisfy the property (P1). There are three cases to consider.

– **Case 1:** *Some vertex of* $V(I_1) \subseteq V(G_\ell)$ *is mapped to* u *in* B_n. Since homomorphisms cannot increase distances between two vertices, we conclude that $V(I_1)$ must be mapped within the subgraph $I_1(u) - (a)$. Suppose further that some vertex on the $(u) - (a)$ path other than u is also in the homomorphic image of $V(I_1)$. Some neighbour of u in $V(I_1) \subseteq V(B_n)$, say u', must also be in the homomorphic image, since otherwise we have a homomorphism from the non-bipartite I_1 to a path, a contradiction. But note that $I_1(u) - (a)$ has a homomorphism to I_1: fold the $(u) - (a)$ path onto the edge $u - u'$ in I_1. Hence, composing the two homomorphisms we obtain a homomorphism from I_1 to I_1 which is not surjective. This contradicts the rigidity of I_1. So in fact the homomorphism must map $V(I_1)$ from G_ℓ entirely within I_1 from B_n, and by rigidity of I_1, this must be the identity map.

- **Case 2:** *Some vertex of $V(I_1) \subseteq V(G_\ell)$ is mapped to v in B_n.* Since homomorphisms cannot increase distances between two vertices, we conclude that $V(I_1)$ must be mapped within the subgraph $(b)-(v)I_2$. But note that $(b)-(v)I_2$ has a homomorphism to I_2 (fold the $(b)-(v)$ path onto any edge incident on v within I_2). Hence, composing the two homomorphisms, we obtain a homomorphism from I_1 to I_2. This is a contradiction, since I_1 and I_2 were incomparable graphs to start with.
- **Case 3:** *No vertex of $V(I_1) \subseteq V(G_\ell)$ is mapped to u or v in B_n.* Then $V(I_1) \subseteq V(\mathrm{G}_\ell)$ must be mapped entirely within one of the following disjoint regions of B_n: (a) $I_1 \setminus \{u\}$, (b) bipartite graph between vertices u and v, and (c) $I_2 \setminus \{v\}$. But then we contradict *rigidity of* I_1 in the first case, *non-bipartiteness of* I_1 in the second case, and *incomparability of* I_1 *and* I_2 in the last.

In a similar way, we could also prove that a valid homomorphism from $\mathrm{G}_\ell \to B_n$ must satisfy the property (P2). ☐

In the above proof, we crucially used incomparability of I_1 and I_2 to rule out flipping an undirected path. It turns out that over fields of characteristic not equal to 2, this is not crucial, since we can divide by 2. We show that if the characteristic of the underlying field is not equal to 2, then the sequence (G_k) in the preceding theorem can be replaced by a sequence of simple undirected cycles of appropriate length. In particular, we establish the following result.

Theorem 11. *Over fields of char $\neq 2$, the family of homomorphism polynomials (f_k), $f_k = f_{G_k,H_k}$, where*

- *G_k is a simple undirected cycle of length $2k+1$ and,*
- *H_k is an undirected complete graph on $(2k+1)^2$ vertices,*

is complete for VBP *under p-projections.*

Proof. **Membership:** As before, it follows from Theorem 8.
Hardness: Let $(g_n) \in$ VBP. Without loss of generality, we can assume that g_n is computable by a layered branching program of polynomial size satisfying the following properties:

- The number of layers, $\ell \geqslant 3$, is odd; say $\ell = 2m+1$. So every path from s to t in the branching program has exactly $2m$ edges.
- The number of layers, is more than the width of the algebraic branching program,

Let us consider f_m when the variables on the edges of H_m have been set to 0, 1, or variables of g_n so that we obtain the undirected graph underlying the layered branching program A_n for g_n as a subgraph of H_m. Now change the weight of the (s,t) edge from 0 to weight y, where y is a new variable distinct from all the other variables of g_n. Call this modified graph B_m. Note that without the new edge, B_m would be bipartite.

Let us understand the homomorphisms from G_m to B_m. Homomorphisms from a simple cycle C to a graph $\mathcal{G}$ are in one-to-one correspondence with closed walks of the same length in $\mathcal{G}$. Moreover, if the cycle C is of odd length, the closed walk must contain a simple odd cycle of at most the same length. Therefore, the only valid homomorphism from G_m to B_m are walks of length $\ell = 2m+1$, and they all contain the edge (s,t) with weight y. But the cycles of length ℓ in B_m are in one-to-one correspondence with s-t paths in A_n. Each cycle contributes 2ℓ walks: we can start the walk at any of the ℓ vertices, and we can follow the directions from A_n or go against those directions. Thus we have,

$$f_{G_m,B_m} = (2(2m+1)) \cdot y \cdot g_n = (2\ell) \cdot y \cdot g_n.$$

Let p be the characteristic of the underlying field. If $p = 0$, we substitute $y = (2\ell)^{-1}$ to obtain g_n. If $p > 2$, then 2ℓ has an inverse if and only if ℓ has an inverse. Since $\ell \geqslant 3$ is an odd number, either p does not divide ℓ or it does not divide $\ell + 2$. Hence, at least one of ℓ, $\ell + 2$ has an inverse. Thus g_n is a projection of f_m or f_{m+1} depending on whether ℓ or $\ell + 2$ has an inverse in characteristic p.

Since $\ell = 2m+1$ is polynomially bounded in n, we therefore show (f_k) is VBP-complete with respect to p-projections over any field of characteristic not equal to 2. □

5.4 VNP-completeness

We end this section by establishing a family of homomorphism polynomial that is complete for VNP with respect to p-projections.

For each $n \in \mathbb{N}$, let $\mathcal{I}_n := \{I_{n1}, I_{n2}, \ldots, I_{nn}\}$ be a set of n rigid and mutually incomparable graphs. If the subscript n is clear from the context, we will drop it and write $\mathcal{I} = \{I_1, I_2, \ldots, I_n\}$. We can further assume that for all $j \in [n]$, I_j is defined on $\Theta(n)$ vertices, in fact $3n+7$ vertices, and its tree-width is also $\Theta(n)$ (see Lemma 3 in the Appendix). For each I_j, we mark two distinct vertices t_j and s_j in its vertex set. Consider the sequence of graphs G_n (see Fig. 4). In words, place I_1 to I_n on an n-cycle, connect the big nodes with the edges from the set $\mathcal{C} := \{(s_j, t_{j+1}) \mid j \in [n-1]\} \cup \{(s_n, t_1)\}$, and finally, to obtain the graph G_n, stretch each edge in $\mathcal{C}$ into a path with $3n+7$ vertices on it.

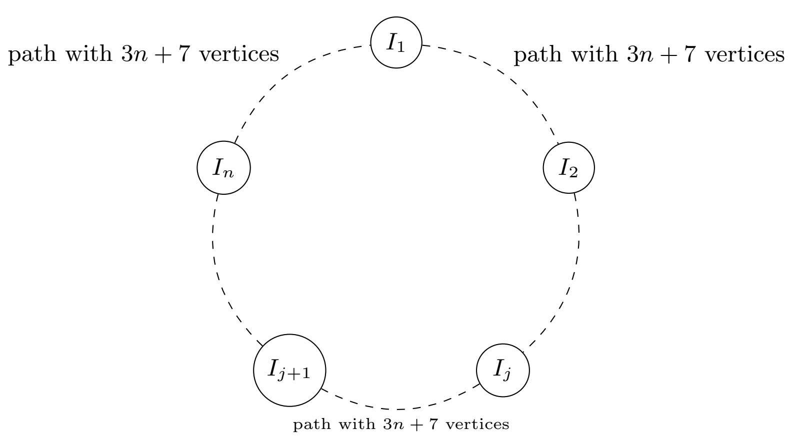

Fig. 4. The Graph G_n.

Theorem 12. *Over any field, the family of homomorphism polynomials* (f_n)*, where*

- G_n *is defined as above (see Fig. 4),*
- H_n *is the undirected complete graph on* $O(n^4)$ *vertices,*
- $f_n(\bar{Y}) = f_{G_n,H_n}(\bar{Y})$*,*

is complete for VNP *with respect to p-projections.*

Proof. *Membership* in VNP follows from Proposition 2. To establish *hardness* we will show that the Hamiltonian cycle family (HC_n) is a p-projection of (f_n). Recall that HC_n is defined as follows:

$$\mathsf{HC}_n := \sum_{\substack{\sigma \in S_n \\ \sigma \text{ is a } n\text{-cycle}}} \prod_{i=1}^{n} x_{i,\sigma(i)}.$$

We now construct a graph UK_n on $O(n^4)$ vertices such that $f_{G_n,UK_n} = \mathsf{HC}_n$. A suitable projection can then restrict H_n to UK_n, showing that HC_n is a projection of f_n.

Consider a copy of I_1, and for each $j \in \{2,\ldots,n\}$, $n-1$ copies of I_j, denoted I_j^i for $i \in \{2,\ldots,n\}$. Let K_n denote a complete directed graph on n vertices $\{v_i \mid i \in [n]\}$.

We will modify K_n to obtain UK_n. We first replace the vertices of K_n as follows: replace v_1 with I_1, and for $i \in \{2,\ldots,n\}$, replace v_i with the set $\{I_j^i \mid 2 \leqslant j \leqslant n\}$. Intuitively by such a replacement we isolate the vertex v_1, and thus make it always the first vertex in a Hamiltonian cycle. This further helps in counting each Hamiltonian cycle exactly once.

Now we add the *connector* edges as follows.

For each edge $\langle v_1, v_i\rangle$ such that $i \neq 1$, we add the edge (s_1, t_2^i) with weight $X_{1,i}$, where s_1 is the marked vertex in I_1, and t_2^i is the marked vertex in I_2^i. Intuitively, using this edge in homomorphisms correspond to using the edge $\langle v_1, v_i\rangle$ in a Hamiltonian cycle.

For each edge $\langle v_i, v_1\rangle$ such that $i \neq 1$, add (s_n^i, t_1) with weight $X_{i,1}$, where s_n^i is the marked vertex in I_n^i, and t_1 is the marked vertex in I_1. As before, using this edge in homomorphisms correspond to using the edge $\langle v_i, v_1\rangle$ in a Hamiltonian cycle.

For each edge $\langle v_i, v_j\rangle$ such that $i \neq j$ and $1 \notin \{i,j\}$, add the following edges $\{(s_k^i, t_{k+1}^j) \mid k \in \{2,\ldots,n-1\}\}$. Moreover, they all have the same weight $X_{i,j}$. As before, s_k^i is the marked vertex in I_k^i and t_{k+1}^j is the marked vertex in I_{k+1}^j. Intuitively, using the edge (s_k^i, t_{k+1}^j) in homomorphisms correspond to using the edge $\langle v_i, v_j\rangle$ in a Hamiltonian cycle such that the vertex v_i is in the k-th position and v_j is in the $(k+1)$-th position.

Now we stretch the connector edges into a path with $3n+7$ vertices on it. Put the label of the connector edge onto the middle edge of this path. Rest of the edges in the path have weight 1. We denote this graph with UK_n. Clearly UK_n is defined on $O(n^4)$ vertices.

We now prove our claim that $\mathsf{HC}_n = f_{G_n,UK_n}$. To prove the claim it suffices to show that homomorphisms from G_n to UK_n are in one-to-one correspondence with the Hamiltonian cycles in K_n. It easily follows that every Hamiltonian cycle gives a homomorphic mapping of G_n into UK_n by following the cycle (based on the intuition described before). For example, if $\langle v_1, v_{k_1}, \ldots v_{k_{n-1}}\rangle$ is a Hamiltonian cycle in K_n, then the homomorphic map of G_n into UK_n is given as follows: I_1 in G_n maps to I_1 in UK_n using identity mapping, then I_2 in

G_n is mapped to $I_2^{k_1}$ in UK_n using identity mapping, and, in general, I_i in G_n is mapped to $I_i^{k_{i-1}}$ in UK_n using identity mapping. For the reverse direction, we use (i) the rigidity and incomparability of the set $\mathcal{I}$, and (ii) the fact that homomorphisms cannot increase distance. Using these two facts we first argue that each rigid node in G_n (from the set $\mathcal{I}$) must map identically to one of its copy in UK_n. We can further argue that no two rigid nodes in G_n can be mapped into the set associated with a single vertex in UK_n. That is, distinct I_i and I_j in G_n can not be mapped simultaneously to I_i^k and I_j^k for any $k \in \{2, \ldots, n\}$. Thus we have shown that a homomorphism from G_n to UK_n necessarily picks out a n-cycle in K_n. Now by the fact there is only one copy of I_1 in UK_n it follows that I_1 in G_n must be mapped to I_1 in UK_n using the identity mapping. This uniquely defines the direction of the n-cycle, and hence each cycle is counted exactly once. $\square$

6 Conclusion

In this paper, we have shown that over finite fields, five families of polynomials are intermediate in complexity between VP and VNP, assuming the PH does not collapse. Over rationals and reals, we have established that two of these families are provably not monotone p-projections of the permanent polynomials. Finally, we have obtained a natural family of polynomials, defined via graph homomorphisms, that is complete for VP with respect to projections; this is the first family defined independent of circuits and with such hardness. Analogous families are shown to be complete for VBP and VNP.

Several interesting questions remain.

The definitions of our intermediate polynomials use the size q of the field $\mathbb{F}_q$, not just the characteristic p. Can we find families of polynomials with integer coefficients, that are VNP-intermediate (under some natural complexity assumption of course) over all fields of characteristic p? Even more ambitiously, can we find families of polynomials with integer coefficients, that are VNP-intermediate over all fields with non-zero characteristic? at least over all finite fields? over fields $\mathbb{F}_p$ for all (or even for infinitely many) primes p?

Our study of homomorphism polynomials raises an intriguing question in this regard. What is the complexity of homomorphism polynomials that are defined on a family G_n such that G_n has tree-width $o(n)$? Equally interestingly, can we find an explicit family of polynomials that is VNP-intermediate in characteristic zero?

A related question is whether there are any polynomials defined over the integers, that are VNP-intermediate over $\mathbb{F}_q$ (for some fixed q) but that are monotone p-projections of the permanent.

Can we show that the remaining intermediate polynomials are also not polynomial-sized monotone projections of the permanent? Do such results have any interesting consequences, say, improved circuit lower bounds?

References

[1] Noga Alon and Ravi B. Boppana. The monotone circuit complexity of Boolean functions. *Combinatorica*, 7(1):1–22, 1987.

[2] David Avis and Hans Raj Tiwary. On the extension complexity of combinatorial polytopes. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP Part I*, pages 57–68, 2013.

[3] Walter Baur and Volker Strassen. The complexity of partial derivatives. *Theoretical Computer Science*, 22(3):317 – 330, 1983.

[4] P. Bürgisser. *Completeness and Reduction in Algebraic Complexity Theory*, volume 7 of *Algorithms and Computation in Mathematics*. Springer, 2000.

[5] Peter Bürgisser. On the structure of Valiant's complexity classes. *Discrete Mathematics & Theoretical Computer Science*, 3(3):73–94, 1999.

[6] Peter Bürgisser. Cook's versus Valiant's hypothesis. *Theoretical Computer Science*, 235(1):71–88, 2000.

[7] Florent Capelli, Arnaud Durand, and Stefan Mengel. The arithmetic complexity of tensor contractions. In *Symposium on Theoretical Aspects of Computer Science STACS*, volume 20 of *LIPIcs*, pages 365–376, 2013.

[8] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[9] Nicolas de Rugy-Altherre. A dichotomy theorem for homomorphism polynomials. In *Mathematical Foundations of Computer Science 2012*, volume 7464 of *LNCS*, pages 308–322. Springer Berlin Heidelberg, 2012.

[10] Josep Díaz, Maria J. Serna, and Dimitrios M. Thilikos. Counting h-colorings of partial k-trees. *Theoretical Computer Science*, 281(1-2):291–309, 2002.

[11] Arnaud Durand, Meena Mahajan, Guillaume Malod, Nicolas de Rugy-Altherre, and Nitin Saurabh. Homomorphism polynomials complete for VP. In *34th Foundation of Software Technology and Theoretical Computer Science Conference, FSTTCS*, pages 493–504, 2014.

[12] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[13] Samuel Fiorini, Serge Massar, Sebastian Pokutta, Hans Raj Tiwary, and Ronald de Wolf. Exponential lower bounds for polytopes in combinatorial optimization. *Journal of the ACM*, 62(2):17, 2015.

[14] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[15] Joshua A. Grochow. Monotone projection lower bounds from extended formulation lower bounds. arXiv:1510.08417 [cs.CC], 2015.

[16] Pavol Hell and Jaroslav Nešetřil. *Graphs and homomorphisms*. Oxford lecture series in mathematics and its applications. Oxford University Press, 2004.

[17] Pavel Hrubes. On hardness of multilinearization, and VNP completeness in characteristics two. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:67, 2015.

[18] Mark Jerrum and Marc Snir. Some exact complexity results for straight-line computations over semirings. *J. ACM*, 29(3):874–897, 1982.

[19] Stasys Jukna. Why is Hamilton Cycle so different from Permanent? http://cstheory.stackexchange.com/questions/27496/why-is-hamiltonian-cycle-so-different-from-permanent, 2014.

[20] Richard M Karp and Richard Lipton. Turing machines that take advice. *L'enseignement mathématique*, 28(2):191–209, 1982.

[21] Richard E. Ladner. On the structure of polynomial time reducibility. *J. ACM*, 22(1):155–171, 1975.

[22] Guillaume Malod and Natacha Portier. Characterizing Valiant's algebraic complexity classes. *Journal of Complexity*, 24(1):16–38, 2008.

[23] Stefan Mengel. Characterizing arithmetic circuit classes by constraint satisfaction problems. In *Automata, Languages and Programming*, volume 6755 of *LNCS*, pages 700–711. Springer Berlin Heidelberg, 2011.

[24] Ran Raz. Elusive functions and lower bounds for arithmetic circuits. *Theory of Computing*, 6:135–177, 2010.

[25] A. A. Razborov. Lower bounds on the monotone complexity of some Boolean functions. *Dokl. Akad. Nauk SSSR*, 281(4):798–801, 1985.

[26] A.A. Razborov. Lower bounds on monotone complexity of the logical permanent. *Mathematical notes of the Academy of Sciences of the USSR*, 37(6):485–493, 1985.

[27] Thomas Rothvoß. The matching polytope has exponential extension complexity. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 263–272, 2014.

[28] Janos Simon. On the difference between one and many (preliminary version). In *Automata, Languages and Programming, Fourth Colloquium, University of Turku, Finland, July 18-22, 1977, Proceedings*, pages 480–491, 1977.

[29] Leslie G. Valiant. Completeness classes in algebra. In *Symposium on Theory of Computing STOC*, pages 249–261, 1979.

[30] Leslie G. Valiant, Sven Skyum, S. Berkowitz, and Charles Rackoff. Fast parallel computation of polynomials using few processors. *SIAM Journal on Computing*, 12(4):641–644, 1983.

Appendix

In this appendix we prove that the graphs G_i, $i \in \{1, 2, 3\}$, from Figure 1, are rigid and pairwise incomparable. For the purpose of proof we will partition the vertices into three classes, namely Red, Blue, and Green (cf. Fig. 1). The vertices of G_i, for $1 \leqslant i \leqslant 3$, are partitioned as follows: the vertex $(7 + i)$ is in the Red set, the vertices 1 and 16 are in the Blue set, and the rest of the vertices are in the Green set.

A graph H is *asymmetric* if the only *automorphism* (isomorphism from H to itself) is the identity.

A graph H is a *core* if every *endomorphism* (homomorphism from H to itself) is an isomorphism (and hence an automorphism).

A graph H is rigid if the only endomorphism is the identity.

H is rigid if and only if it is an asymmetric core.

Let χ_H denote the chromatic number of H, that is, the least k such that some map from $V(H)$ to the set of colours $[k]$ gives all adjacent vertices distinct colours. If there is a homomorphism from G to H, then the definition of homomorphism implies that $\chi(G) \leq \chi(H)$. Hence, if we define *vertex-criticality* saying that that H is vertex-critical if for every $u \in V(H)$, $\chi_{H\backslash\{v\}} < \chi_H$, then it follows that every vertex-critical graph is a core.

Claim. 1: Each graph in $\{G, G_1, G_2, G_3\}$ is a core.

Claim. 2: Each graph in $\{G_1, G_2, G_3\}$ is asymmetric.

Hence, each G_i is rigid.

Claim. 3: The graphs in $\{G_1, G_2, G_3\}$ are pairwise incomparable; for $i \neq j$, there is no homomorphism from G_i to G_j.

Proof. (of Claim 1). We show that G (and hence also each G_i) is not 5-colourable, while for every $u \in [16]$, each $G_i\backslash\{u\}$ is 5-colourable. Hence all 4 graphs are 6-chromatic vertex-critical.

Non-5-colourability: The vertices 1 to 5 form a clique and must get distinct colours, say 1 to 5. Now there is a unique way of extending the colouring sequentially to 6,7,8, ... But this assigns colour 1 to 16, and 1 and 16 are neighbours. So no 5-colouring is possible.

5-colourability: Consider $G_i\backslash\{u\}$. Colour node j with colour j mod 5 if $j < u$, with colour $j-1$ mod 5 if $j > u$. This satisfies all edge constraints: For a black edge (j, k), $1 \leq |j-k| \leq 4$, so if both j and k are present, then their colours are distinct even if $j < u < k$. If the blue-red edge is present, note that the red vertex gets colour 2,3,4,or 5, while vertex 1 always gets colour 1. □

Proof. (of Claim 2). Since isomorphisms must preserve degrees vertex-wise, consider the degrees of vertices in the graphs. First, group the vertices of G by degree.
degree 5: $\{1, 2, 15, 16\}$
degree 6: $\{3, 14\}$

degree 7: $\{4, 13\}$
degree 8: $\{5, 6, 7, 8, 9, 10, 11, 12\}$.

Similarly, group the vertices of G_i by degree.
degree 5: $\{2, 15, 16\}$
degree 6: $\{1, 3, 14\}$
degree 7: $\{4, 13\}$
degree 8: $\{5, 6, 7, 8, 9, 10, 11, 12\} \setminus$ {the red node 7+i}
degree 9: the red node $7 + i$

Consider an automorphism f on G_1. Since only vertex 8 has degree 9, f must map 8 to 8. Vertex 1 is the only neighbour of 8 with degree 6, so f must map 1 to 1. Vertex 1 has two degree-5 neighbours, 2 and 16, but 16 has another degree-5 neighbour 15 while 2 does not have any degree-5 neighbour, so f cannot swap these degree-5 neighbours of 1. So f maps 2 to 2 and 16 to 16. Proceeding this way based on degree, we see that f must in fact fix every vertex.

An identical argument works for G_2. For G_3, one additional twist: the red vertex 10 gets mapped to 10. Now 10 has two degree-6 neighbours, 1 and 14. Can f map 1 to 14? But 1 has a degree-6 neighbour 3, while 14 has no degree-6 neighbour. So f cannot swap 1 and 14. □

Proof. (of Claim 3). Suppose to the contrary that $f : V_1 \to V_2$ is a homomorphism from G_1 to G_2 (the argument is similar for other pairs). If f is not surjective, then by vertex-criticality, G_1 has a homomorphism to a 5-colourable graph, but $\chi(G_1) = 6$, a contradiction. So f must be surjective.

Furthermore, f must induce a bijection between the edges of G_1 and G_2. If it didn't, then two edges of G_1 are mapped to the same edge of G_2. This implies that two vertices of G_1 are mapped to the same vertex of G_2, violating surjectivity.

Thus the vertex degrees must be preserved exactly: for each $u \in V_1$, the degree of u in G_1 is the same as the degree of $f(u)$ in G_2.

Since the red vertices are the only vertices with degree 9, f must map the red vertex of G_1, vertex 8, to the red vertex of G_2, vertex 9. Now use the argument as used in Claim 2 to extend this mapping. f must map 1 to 1, 2 to 2, and so on. We thus reach the conclusion that f must map 8 to 8, contradicting $f(8) = 9$. Hence no such map f is possible. □

In fact, the construction of $G_i, i \in [3]$, can be generalized to obtain a sequence of sets of rigid and mutually incomparable graphs. These constructions were given by Hell and Nešetřil (Exercise 6, Chapter 4, [16]).

Let $1 \leqslant \ell \leqslant n$. Consider the following graph $H(n, \ell)$: the vertex set is $\{1, 2, \ldots, 3n + 7\}$, and the edges are $(1, 3n + 7), (1, n + 4 + \ell)$ and all (i, j) with $1 \leqslant |i - j| \leqslant n + 1$.

Lemma 3. *The graph $H(n, \ell)$ as defined above satisfy the following properties:*

- *Each $H(n, \ell)$ is rigid.*
- *There is no homomorphism $H(n, \ell) \to H(n, \ell')$ for $\ell \neq \ell'$.*

ECCC ISSN 1433-8092
http://eccc.hpi-web.de