A Dichotomy Theorem for Polynomial Evaluation

Irénée Briquel, Pascal Koiran

Toward a Dichotomy Theorem for Polynomial Evaluation

Irénée Briquel and Pascal Koiran
LIP*, École Normale Supérieure de Lyon, Université de Lyon
[Irenee.Briquel,Pascal.Koiran]@ens-lyon.fr

February 12, 2009

Abstract

A dichotomy theorem for counting problems due to Creignou and Hermann states that or any finite set S of logical relations, the counting problem #SAT(S) is either in FP, or #P-complete. In the present paper we study polynomial evaluation from this dichotomic point of view. We show that the "hard" cases in the Creignou-Hermann theorem give rise to VNP-complete families of polynomials, and we give partial results for the "easy" case of this dichotomy theorem. We also prove that several problems which were known to be #P-complete under Turing reductions only are in fact #P-complete under many-one reductions.

1 Introduction

.

In a seminal paper, Schaefer [13] proved a dichotomy theorem for boolean constraint satisfaction problems: he showed that for any finite set S of logical relations the satisfiability problem SAT(S) for S-formulas is either in P, or NP-complete. Here, an S-formula over a set of n variables is a conjunction of relations of S where the arguments of each relation are freely chosen among the n veriables. For instance, when S is the set of all 3-clauses SAT(S) is the canonical NP-complete problem 3-SAT; when S is the set of all 2-clauses SAT(S) is the polynomial time problem 2-SAT. Schaefer's result was subsequently extended in a number of directions. In particular, dichotomy theorems were obtained for counting problems, optimization problems and the decision problem of quantified boolean formulas. An account of this line of work can be found in the book by Creignou, Khanna and Sudan [5]. In a different direction, constraint satisfaction problems were also studied

*UMR 5668 ENS Lyon, CNRS, UCBL associée à l'INRIA.

over non-boolean domains. This turned out to be a surprisingly difficult question, and it took a long time before a dichotomy theorem over domains of size 3 could be obtained [3].

In the present paper we study polynomial evaluation from this dichotomic point of view. We work within Valiant's algebraic framework: the role of the complexity class NP in Schaefer's dichomtomy theorem will be played by the class VNP of "easily definable" polynomial families, and the role of P will be played by the class VP of "easily computable" polynomial families [14, 2]. There is a well-known connection between counting problems and polynomial evaluation. For instance, as shown by Valiant the permanent is complete in both settings [15, 14]. In the realm of counting problems, a dichotomy theorem was obtained by Creignou and Hermann [4, 5].

Theorem 1 *For any finite set S of logical relations, the counting problem* #SAT(S) *is either in* FP*, or* #P*-complete.*

In fact, the sets S such that #SAT(S) is in FP are exactly the sets containing only affine constraints (a constraint is called affine if it expressible as a system of linear equations over $\mathbb{Z}/2\mathbb{Z}$).

Main Contributions

To a family of boolean formulas (ϕ_n) we associate the multilinear polynomial family

$$P(\phi_n)(\overline{X}) = \sum_{\overline{\varepsilon}} \phi_n(\overline{\varepsilon})\overline{X}^{\overline{\varepsilon}}, \tag{1}$$

where $\overline{X}^{\overline{\varepsilon}}$ is the monomial $X_1^{\varepsilon_1} \cdots X_{k(n)}^{\varepsilon_{k(n)}}$, and $k(n)$ is the number of variables of ϕ_n. Imagine that the ϕ_n are chosen among the S-formulas of a fixed finite set S of logical relations. One would like to understand how the complexity of the polynomials $P(\phi_n)$ depends on S. Note that when the variables X_i are all set to 1, we obtain the counting problem #SAT(S). It seems therefore reasonable to conjecture that the "easy" and "hard" cases are the same as in Creignou and Hermann's dichotomy theorem. We can partially prove this conjecture.

Definition 1 *A family (ϕ_n) of S-formulas is called a p-family if ϕ_n is a conjunction of at most $p(n)$ relations from S, for some polynomial p (in particular, ϕ_n depends on polynomially many variables when S is finite).*

Theorem 2 (Main Theorem) *Let S be a finite set of logical relations which contains at least one relation that is not affine. Then there exists a p-family (ϕ_n) of S-formulas such that the corresponding polynomial family $P(\phi_n)$ is* VNP*-complete.*

The only case that we could not settle, thereby preventing us from obtaining a complete dichotomy theorem, is the case of sets S containing only affine relations. Ironically, this is the easiest case in Creignou and Hermann's dichotomy theorem. Indeed, in this case a S-formula ϕ over a set of n boolean variables defines an affine subspace of $(\mathbb{Z}/2\mathbb{Z})^n$. The number of satisfying assignements is 2^d, where d is the dimension of this subspace. This dimension can be easily computed by Gaussian elimination. In our algebraic framework we need to solve a "weighted counting" problem: each point $(\varepsilon_1, \ldots, \varepsilon_n)$ of the affine subspace comes with a weight $X_1^{\varepsilon_1} \cdots X_n^{\varepsilon_n}$. We give efficient evaluation algorithms in several cases, but the general case remains open.

The remainder of this paper is mostly devoted to the proof of Theorem 2. The results that we obtain along the way are in our opinion at least as interesting as Theorem 2 by itself. First, we obtain several new VNP-completeness results. The main ones are about:

(i) the vertex cover polynomial $\mathrm{VCP}(G)$ and the independent set polynomial $\mathrm{IP}(G)$; these polynomials are associated to a vertex-weighted graph G. Most VNP-completeness results in the litterature (and certainly all the results in Chapter 3 of [2]) are about edge-weighted graphs rather than vertex-weighted graphs.

(ii) the antichain polynomial $\mathrm{AP}(X)$ and the ideal polynomial $\mathrm{IPP}(X)$; these polynomials are associated to a weighted poset $(X, \leq)$.

Like in most VNP-completeness results, the reduction that we use is Valiant's p-projection. In his work on VNP families that are neither p-computable nor VNP-complete, Bürgisser [1, 2] introduced the more general "c-reductions". They are akin to the Turing (or oracle) reductions from discrete complexity theory. The c-reduction has not been used widely in VNP-completeness proofs. The only examples that we are aware of are:

(i) A remark in [2] on probability generating functions.

(ii) The VNP-completeness of the weighted Tutte polynomial in [11]. Even there, the power of c-reductions is used in a very restricted way since a single oracle call is perfomed in each reduction.

By contrast, the power of Turing reductions has been put to good use in #P-completeness theory (mostly as a tool for performing interpolation). Indeed, as pointed out in [8], "interpolation features prominently in a majority of #P-completeness proofs", and "it is not clear whether the phenomenon of #P-completeness would be as ubiquitous if many-one reducibility were to be used in place of Turing." We argue that the importance of Turing reductions in #P-completeness should be revised downwards since, as a byproduct of our VNP-completeness results, we can replace Turing reductions by many-one reductions in several #P-completeness results from the litterature. In

particular, we obtain a many-one version of Creignou and Hermann's dichotomy theorem[1]. We leave it as an open problem whether the 0/1 partial permanent is #P-complete under many-one reductions (see Section 3 for a definition of the partial permanent, and [7] for a # P-completeness proof under oracle reductions).

Organization of the Paper and Additional Results

Earlier in this section we gave an informal introduction to constraint satisfaction problems. We give more precise definitions at the beginning of Section 2. The remainder of that section is devoted to Valiant's algebraic model of computation. We then begin the proof of Theorem 2. The high-level structure of the proof is similar to Creignou and Hermann's proof of #P-completeness of the corresponding counting problems in [4]. The singletons $S = \{\mathrm{OR}_2\}$, $S = \{\mathrm{OR}_1\}$ and $S = \{\mathrm{OR}_0\}$ play a special role in the proof. Here OR_2 denotes the negative two-clause $(x, y) \mapsto (\overline{x} \vee \overline{y})$; OR_0 denotes the positive two-clause $(x, y) \mapsto (x \vee y)$; and OR_1 denotes the implicative two-clause $(x, y) \to (\overline{x} \vee y)$. The corresponding VNP-completeness results are established in sections 3 and 4. These results are put together in Section 5 to complete the proof of Theorem 2. In Section 6, we build on our VNP-completeness results to prove #P-completeness under many-one reductions for several problems which were only known to be #P-complete under oracle reductions.

Some additional results are provided in the appendix. The first appendix deals with affine constraints. We present some special cases when an efficient evaluation of $P(\phi)$ is possible. Motivated by one of these positive cases, we observe in the second appendix that the permutation function $\mathrm{PERMUTATION}_n$ (the boolean function in n^2 variables that accepts the $n \times n$ permutation matrices) is not affine. This is to be expected since an efficient (nonuniform) algorithm for evaluating the 0/1 permanent would otherwise follow. The point of our (simple) observation about $\mathrm{PERMUTATION}_n$ is that it does not rely on any unproven assumption. Likewise, it is possible to prove unconditionally that treewidth-based methods cannot be used to evaluate the permanent efficiently [9]. It would be interesting to obtain similar impossibility results for other counting or polynomial evaluation methods.

[1] It was already claimed in [4, 5] that Theorem 1 holds true for many-one reductions. This was not fully justified since the proof of Theorem 1 is based on many-reductions from problems which were previously known to be #P-complete under oracle reductions only. The present paper shows that this claim was indeed correct.

2 Preliminaries

2.1 Constraint satisfaction problems

We define a logical relation to be a function from $\{0,1\}^k$ to $\{0,1\}$, for some integer k called the rank of the relation. Let us fix a finite set $S = \{\phi_1, \ldots, \phi_n\}$ of logical relations. An S-formula over n variables $(x_1, \ldots, x_n)$ is a conjunction of boolean formulas, each of the form $g_i(x_{j_i(1)}, \ldots, x_{j_i(k_i)})$ where each g_i belongs to S and k_i is the rank of g_i. In words, each element in the conjunction is obtained by applying a function from S to some variables chosen among the n variables $x_1, \ldots, x_n$.

An instance of the problem SAT(S) studied by Schaefer [13] is an S-formula ϕ, and one must decide whether ϕ is satisfiable. For instance, consider the 3 boolean relations $\mathrm{OR}_0(x,y) = x \vee y$, $\mathrm{OR}_1(x,y) = \overline{x} \vee y$ and $\mathrm{OR}_2(x,y) = \overline{x} \vee \overline{y}$. The classical problem 2-SAT is SAT(S) where $S = \{\mathrm{OR}_0, \mathrm{OR}_1, \mathrm{OR}_2\}$. The counting problem $\#SAT(S)$ was studied by Creignou and Hermann [4]. In this paper we study the complexity of evaluating the polynomials $P(\phi)$ in (1). We would like to understand which sets S give rise to VNP-complete polynomial families, and which one give rise only to easy to compute families. We next define these notions precisely.

2.2 Algebraic complexity theory: Valiant's model

In Valiant's model one studies the computation of multivariate polynomials. This can be done over any field. In the sequel we fix a field K of characteristic $\neq 2$. All considered polynomials are over K.

A p-family is a sequence $f = (f_n)$ of multivariate polynomials such that the number of variables and the degree are polynomially bounded functions of n. A prominent example of a p-family is the permanent family PER = (PER_n), where PER_n is the permanent of an $n \times n$ matrix with independent indeterminate entries.

We define the complexity of a polynomial f to be the minimum number $L(f)$ of nodes of an arithmetic circuit computing f. We recall that the internal nodes of an arithmetic circuit perform additions or multiplications, and each input node is labeled by a constant from K or a variables X_i.

Definition 2 (VP) *A p-family (f_n) is p-computable if $L(f_n)$ is a polynomially bounded function of n. Those families constitute the complexity class* VP.

In Valiant's model, VNP is the analogue of the class NP (or perhaps more accurately, of #P).

Definition 3 (VNP) *A p-family (f_n) is called p-definable if there exists a*

p-computable family $g = (g_n)$ *such that*

$$f_n(X_1, \ldots, X_{p(n)}) = \sum_{\varepsilon \in \{0,1\}^{q(n)}} g_n(X_1, \ldots, X_{p(n)}, \varepsilon_1, \ldots, \varepsilon_{q(n)})$$

The set of p-definable families forms the class VNP*.*

Clearly, VP is included in VNP. To define VNP-completeness we need a notion of reduction:

Definition 4 (*p*-projection) *A polynomial* f_n *with* v *arguments is said to be a projection of a polynomial* g_m *with* u *arguments, and we denote it* $f_n \leq g_m$*, if* $f(X_1, \ldots, X_v) = g_m(a_1, \ldots, a_u)$ *where each* a_i *is a variable of* f_n *or a constant from* K*.*

A p-family $f = (f_n)$ *is a p-projection of* $g = (g_m)$ *if there exists a polynomially bounded function* $t : \mathbb{N} \to \mathbb{N}$ *such that*

$$\exists n_0 \forall n \geq n_0, f_n \leq g_{t(n)}.$$

We write $f \leq_p g$ *if* f *is a p-projection of* g*.*

Definition 5 (VNP-completeness) *A p-family* $g \in$ VNP *is* VNP*-complete if* $f \leq_p g$ *for every p-family* $f \in$ VNP*.*

The VNP-completeness of the permanent under p-projections [14, 2] is a central result in Valiant's theory.

By Valiant's criterion (Proposition 2.20 in [2]), for any finite S of logical relations and any p-family (ϕ_n) of S-formulas the polynomials $(P(\phi_n))$ form a VNP family. All the work in the proof of Theorem 2 therefore goes into the hardness proof.

3 Monotone 2-clauses

In this section we consider the set $\{\mathrm{OR}_2\} = \{(x, y) \mapsto (\overline{x} \vee \overline{y})\}$ and $\{\mathrm{OR}_0\} = \{(x, y) \mapsto (x \vee y)\}$. For $S = \{\mathrm{OR}_2\}$ and $S = \{\mathrm{OR}_0\}$, we show that there exists a VNP-complete family of polynomials $(P(\phi_n))$ associated to a p-family of S-formulas (ϕ_n).

The partial permanent $\mathrm{PER}^*(A)$ of a matrix $A = (A_{i,j})$ is defined by the formula:

$$\mathrm{PER}^*(M) = \sum_{\pi} \prod_{i \in def\, \pi} A_{i\pi(i)}$$

where the sum runs over all injective partial maps from $[1, n]$ to $[1, n]$. It is shown in [2] that the partial permanent is VNP-complete (the proof is attributed to Jerrum). The partial permanent may be written as in (1),

where ϕ_n is the boolean formula that recognizes the matrices of partial maps from $[1,n]$ to $[1,n]$. But ϕ_n is a p-family of $\{\mathrm{OR}_2\}$-formulas since

$$\phi_n(\varepsilon) = \bigwedge_{i,j,k:j\neq k} \overline{\varepsilon_{ij}} \vee \overline{\varepsilon_{ik}} \wedge \bigwedge_{i,j,k:i\neq k} \overline{\varepsilon_{ij}} \vee \overline{\varepsilon_{kj}}.$$

Here the first conjunction ensures that the matrix ε has no more than one 1 on each row; the second one ensures that ε has no more than one 1 on each column. We have obtained the following result.

Theorem 3 *The family (ϕ_n) is a p-family of $\{\mathrm{OR}_2\}$-formulas, and the polynomial family $(P(\phi_n))$ is* VNP*-complete under p-projections.*

The remainder of this section is devoted to the set $S = \{\mathrm{OR}_0\} = \{(x,y) \mapsto x \vee y\}$. The role played by the partial permanent in the previous section will be played by vertex cover polynomials. There is more work to do because the corresponding VNP-completeness result is not available from the litterature.

Consider a vertex-weighted graph $G = (V, E)$: to each vertex $v_i \in V$ is associated a weight X_i. The vertex cover polynomial of G is

$$\mathrm{VCP}(G) = \sum_S \prod_{v_i \in S} X_i \tag{2}$$

where the sum runs over all vertex covers of G (recall that a vertex cover of G is a set $S \subseteq V$ such that for each edge $e \in E$, at least one of the two endpoints of e belongs to S). The univariate vertex cover polynomial defined in [6] is a specialization of ours; it is obtained from $\mathrm{VCP}(G)$ by applying the substitutions $X_i := X$ (for $i = 1, \ldots, n$), where X is a new indeterminate.

Our main result regarding $\{\mathrm{OR}_0\}$-formulas is as follows.

Theorem 4 *There existe a family G_n of polynomial size bipartite graphs such that:*

1. *The family $(\mathrm{VCP}(G_n))$ is* VNP*-complete.*
2. *$\mathrm{VCP}(G_n) = P(\phi_n)$ where ϕ_n is a p-family of $\{\mathrm{OR}_0\}$-formulas.*

Given a vertex-weighted graph G, let us associate to each $v_i \in V$ a boolean variable ε_i. The interpretation is that v_i is chosen in a vertex cover when ε_i is set to 1. We then have

$$\mathrm{VCP}(G) = \sum_{\varepsilon \in \{0,1\}^{|V|}} \Big[\bigwedge_{(v_i,v_j)\in E} \varepsilon_i \vee \varepsilon_j \Big] \overline{X}^{\overline{\varepsilon}}.$$

The second property in Theorem 4 will therefore hold true for any family (G_n) of polynomial size graphs.

To obtain the first property, we first establish a VNP-completeness result for the independent set polynomial $\mathrm{IP}(G)$. This polynomial is defined like the vertex cover polynomial, except that the sum in (2) now runs over all independent sets S (recall that an independent set is a set $S \subseteq V$ such that there are no edges between any two elements of S).

Theorem 5 *There exists a family (G'_n) of polynomial size graphs such that $\mathrm{IP}(G'_n) = \mathrm{PER}^*_n$ where PER^*_n is the $n \times n$ partial permanent. The family $\mathrm{IP}(G'_n)$ is therefore* VNP*-complete.*

Proof. The vertices of G'_n are the n^2 edges ij of the complete bipartite graph $K_{n,n}$, and the associated weight is the indeterminate X_{ij}. Two vertices of G'_n are connected by an edge if they share an endpoint in $K_{n,n}$. An independent set in G'_n is nothing but a partial matching in $K_{n,n}$, and the corresponding weights are the same. □

Next we obtain a reduction from the independent set polynomial to the vertex cover polynomial. The connection between these two problems is not astonishing since vertex covers are exactly the complements of independent sets. In particular, every graph has the same number of independent sets and vertex covers. But we deal here with weighted counting problems, so that there is a little more work to do. The connection between independent sets and vertex covers does imply a relation between the polynomials $\mathrm{IP}(G)$ and $\mathrm{VCP}(G)$. Namely,

$$\mathrm{IP}(G)(X_1, \ldots, X_n) = X_1 \cdots X_n \cdot \mathrm{VCP}(G)(1/X_1, \ldots, 1/X_n). \tag{3}$$

Indeed,

$$\mathrm{IP}(G) \;=\; \sum_{S \text{ independent}} \frac{X_1 \cdots X_n}{\prod_{v_i \notin S} X_i} \;=\; X_1 \cdots X_n \sum_{S' \text{ vertex cover}} \frac{1}{\prod_{v_i \in S'} X_i}.$$

Recall that the incidence graph of a graph $G' = (V', E')$ is a bipartite graph $G = (V, E)$ where $V = V' \cup E'$. In the incidence graph there is an edge between $e' \in E'$ and $u' \in V'$ if u' is one the two endpoints of e' in G. When G' is vertex weighted, we assign to each V'-vertex of G the same weight as in G and we assign to each E'-vertex of G the constant weight -1.

Lemma 1 *Let G' be a vertex weighted graph and G its vertex weighted incidence graph as defined above. We have:*

$$\mathrm{VCP}(G) = (-1)^{e(G')}\mathrm{IP}(G') \tag{4}$$

and

$$\mathrm{IP}(G) = (-1)^{e(G')}\mathrm{VCP}(G') \tag{5}$$

where $e(G')$ is the number of edges of G'.

Proof. We begin with (4). To each independent set I' of G' we can injectively associate the vertex cover $C = I' \cup E'$. The weight of C is equal to $(-1)^{e(G')}$ times the weight of I'. Moreover, the weights of all other vertex covers of G add up to 0. Indeed, any vertex cover C which is not of this form must contain two vertices $u', v' \in V'$ such that $u'v' \in E'$. The symmetric difference $C\Delta\{u'v'\}$ remains a vertex cover of G, and its weight is opposite to the weight of C since it differs from C only by a vertex $u'v'$ of weight -1.

It is possible to obtain (5) by a similar argument. Here, we will deduce this relation from (3) and (4):

$$\begin{aligned} \mathrm{IP}(G)(X_1, \ldots, X_n) &= X_1 \cdots X_n \cdot \mathrm{VCP}(G)(1/X_1, \ldots, 1/X_n) \\ &= (-1)^{e(G')} X_1 \cdots X_n \cdot \mathrm{IP}(G')(1/X_1, \ldots, X_1, \ldots, 1/X_n) \\ &= (-1)^{e(G')} \mathrm{VCP}(G'). \end{aligned}$$

The first and last equalities follow from (3), and the second equality follows from (4). □

To complete the proof of Theorem 4 we apply Lemma 1 to the graph $G' = G'_n$ of Theorem 5. The resulting graph $G = G_n$ satisfies $\mathrm{VCP}(G_n) = \mathrm{IP}(G'_n) = \mathrm{PER}^*_n$ since G'_n has an even number of edges: $e(G'_n) = n^2(n-1)$.

4 Implicative 2-clauses

Here we consider the set $S = \{\mathrm{OR}_1\} = \{(x, y) \to (x \vee \overline{y})\}$. Those logical relations are called implicative, because $x \vee \overline{y}$ is equivalent to $y \Rightarrow x$. The #P-completeness of #SAT(S) was established by a chain of reductions in [12] and [10]. Here we will follow this chain of reductions to find a VNP-complete family associated to S-formulas. These two articles show consecutively that the problems of counting the independent sets, the independent sets in a bipartite graph, the antichains in partial ordered sets (posets), the ideals in posets, and finally satisfaction of implicative 2-clauses are #P-complete. We will start from the family (G'_n) such that $\mathrm{IP}(G'_n) = \mathrm{PER}^*_n$, whose existence is stated in Theorem 5, and follow the reductions for the counting problems.

We first transform the family (G'_n) into a family of bipartite graphs, without changing the independent set polynomials.

Lemma 2 *There exists a family of bipartite graphs (G''_n) such that $\mathrm{IP}(G''_n) = \mathrm{PER}^*_n$, the partial permanent of size $n \times n$.*

Proof. By Lemma 1, we know how to transform a graph G into a bipartite graph G' such that $\mathrm{VCP}(G') = (-1)^{e(G)}\mathrm{IP}(G)$ and $\mathrm{IP}(G') = (-1)^{e(G)}\mathrm{VCP}(G)$ where $e(G)$ is the number of edges of G. By applying this transformation one more time to G', we obtain a graph G'' such that:

$$\mathrm{IP}(G'') = (-1)^{e(G')}\mathrm{VCP}(G') = (-1)^{e(G)}\mathrm{IP}(G)$$

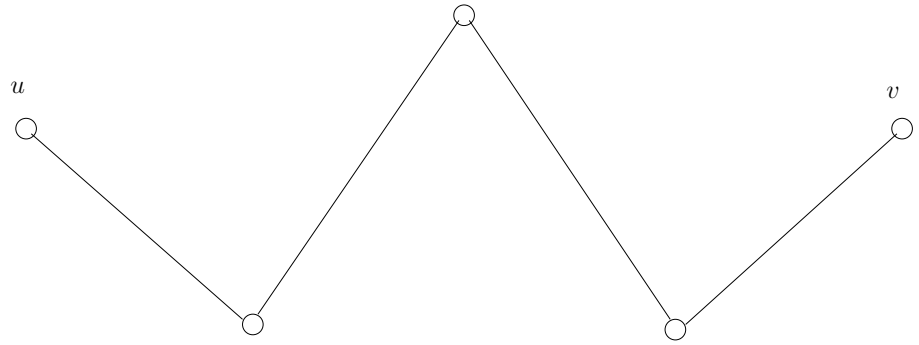

Figure 1: The transformation of Lemma 2

Thus, the transformation of G into G'' consists in the replacement of each edge (u, v) in G by the subgraph represented in Figure 1.

In Theorem 5 we introduced a family (G'_n) such that G'_n has an even number of edges, and $\mathrm{IP}(G'_n) = \mathrm{PER}^*_n$. By applying the transformation above to (G'_n), we obtain a bipartite family (G''_n) such that $\mathrm{IP}(G''_n) = \mathrm{PER}^*_n$. □

In the following we will not only use the statement of this lemma: we will also use the structure of G''_n. More precisely, let us denote by V_1 and V_2 the partite sets of G''_n. We will use for instance the fact that in one of those two sets, say V_1, all vertices have weight -1.

It is pointed out in [12] that, given a bipartite graph, one can construct naturally a partially ordered set. From the bipartite graph $G''_n = (V_1, V_2, E)$, we define the partially ordered set $(X_n, \leq)$ with $X_n = V_1 \cup V_2$, and given x and y in X_n, $x \leq y$ if and only if $x \in V_1$, $y \in V_2$ and $(x, y) \in E$. We see easily that $\leq$ is transitive and antisymmetric.

Next we recall the definition of an antichain.

Definition 6 (Antichain) *An antichain A in a poset $(X, \leq)$ is a subset of X such that for all pair (x, y) of elements of A, x and y are incomparable.*

We define the antichain polynomial of a (weighted) poset $(X, \leq)$ as the polynomial:

$$\mathrm{AP}(X) = \sum_A \prod_{x \in A} w(x)$$

where the sum runs over all antichains A of $(X, \leq)$. Let us consider a bipartite graph G and its corresponding poset $(X, \leq)$. A set $S \subseteq X$ is an antichain in $(X, \leq)$ if and only if it is independent in G. We thus have: $\mathrm{AP}(X) = \mathrm{IP}(G)$. Thus, we can identify the families $(\mathrm{AP}(X_n))$ and $(\mathrm{IP}(G''_n))$. We then define the notion of ideal in a poset.

Definition 7 (Ideal) *An ideal I in a poset $(X, \leq)$ is a subset of X such that for all $x \in I$, all y such that $y \leq x$ belong to I.*

We can also define the ideal polynomial $\mathrm{IPP}(X)$ of the ideals in a poset $(X, \leq)$:

$$\mathrm{IPP}(X) = \sum_{I} \prod_{x \in I} w(x)$$

where the sum runs over all ideals I of $(X, \leq)$.

Given an ideal I in a poset $(X, \leq)$, the maximal elements of I form an antichain A: since they are maximal in I, they cannot be compared. Conversely, given an antichain A, the set of elements x that are less than an element of A form an ideal. One can verify easily that those transformations are bijective and inverse of each other. We thus have a bijection between the ideals and the antichains of a given poset. This fact suffices to the authors of [12], since the bijection shows that a poset has the same number of antichains and ideals; the counting problems are thus equivalent.

But for our weighted counting problems, since the ideals and the antichains have different weights, we cannot identify simply $\mathrm{AP}(X)$ and $\mathrm{IPP}(X)$ for any poset X.

We do not know how to reduce a family of antichain polynomials into ideal polynomials in general, but in the case of the family $(\mathrm{AP}(X_n))$, since the structure of the family (G''_n) is particular, the problem is easier. We claim the following.

Theorem 6 *For all integer n, we have:* $\mathrm{AP}(X_n) = \mathrm{IPP}(X_n)$

The proof will be given at the end of this section.

Corollary 1 *There exists a* VNP*-complete family of polynomials of the form* $(\mathrm{IPP}(X_n))$*.*

To conclude, we note that the ideal polynomial in a poset $(X, \leq)$ may be expressed as a polynomial associated to a S-formula. Namely, we associate to each $x_i \in X$ a boolean variable ε_i with the intended meaning that x_i belongs to an ideal when ε_i is true. For every pair $(x_i, x_j) \in X$ such that $x_i \leq x_j$, the condition $x_j \in I \Rightarrow x_i \in I$ may be expressed by $(\varepsilon_j \Rightarrow \varepsilon_i)$, or $(\varepsilon_i \vee \overline{\varepsilon_j})$. Thus, we have

$$\mathrm{IPP}(X) = \sum_{\varepsilon \in \{0,1\}^{|X|}} \Big[\bigwedge_{(i,j): x_i \leq x_j} \varepsilon_i \vee \overline{\varepsilon_j} \Big] \overline{X}^{\overline{\varepsilon}},$$

and as a result:

Theorem 7 *There exists a* VNP*-complete family of polynomials associated to a p-family of* $\{\mathrm{OR}_1\}$*-formulas.*

To complete this section, we now provide the proof of Theorem 6. Let us fix an integer n. We recall that in the bipartite graph $G''_n = (V_1, V_2, E)$ constructed in Lemma 2, each vertex of V_1 has weight -1. We also know that $|V_1|$ is even, since the elements of V_1 are added two by two in the transformation from G'_n to G''_n.

Fortunately, by changing the correspondence between antichains and ideals, we can preserve the weights: we will construct in Lemma 3 a bijection from the antichains to the ideals of X_n that preserves the weights, and thus we have:

$$\mathrm{AP}(X_n) = \mathrm{IPP}(X_n).$$

Lemma 3 *There exists a bijection (different from the natural one considered previously) from the antichains to the ideals of X_n, this one keeping the weights unchanged.*

Proof. To an antichain A of X_n, we associate the set I such that:

- A and I coincide on V_2.
- $I \cap V_1$ is the complement of $A \cap V_1$ in V_1.

The map $A \mapsto I$ is clearly injective, and one can verify that the image I is an ideal: given $x \in X_n$ and $y \in I$ such that $x \leq y$, we have that $x \in V_1$ and $y \in V_2$. Therefore, $y \in A$, and x cannot belong to A as the elements of A are incomparable. Thus, x belong to I. Our map is thus a bijection from the antichains to the ideals of X_n.

Since all the elements of V_1 have weight -1 and $|V_1|$ is even, the weights of I and A differ by a factor $(-1)^{|V_1|} = 1$. □

5 The general case

In this section we complete the proof of Theorem 2. The proof of this result is an analogue of the proof of the #P-completeness of the corresponding counting problems given in [5]. We will adapt this proof to our context. The authors use the notion of perfect and faithful implementation (definition 5.1 in [5]):

Definition 8 *A conjunction of α boolean constraints $\{f_1, \ldots, f_\alpha\}$ over a set of variables $\overline{x} = \{x_1, \ldots, x_n\}$ and $\overline{y} = \{y_1, \ldots, y_n\}$ is a* perfect and faithful implementation *of a boolean formula $f(\overline{x})$, if and only if*

1. *for any assignment of values to $\overline{x}$ such that $f(\overline{x})$ is true, there exists a unique assignment of values to $\overline{y}$ such that all the constraints $f_i(\overline{x}, \overline{y})$ are satisfied.*

2. *for any assignment of values to $\overline{x}$ such that $f(\overline{x})$ is false, no assignment of values to $\overline{y}$ can satisfy more than $(\alpha - 1)$ constraints.*

We refer to the set $\overline{x}$ as the function variables and to the set $\overline{y}$ as the auxiliary variables.

We say, that a set S of logical relations *implements perfectly and faithfully* a boolean formula $f(\overline{x})$ if there is a S-formula that implements $f(\overline{x})$ perfectly and faithfully. We also extend the definition to logical relations: a set S of logical relations implements perfectly and faithfully a logical relation f if S implements perfectly and faithfully every application of f to a set of variables $\overline{x}$.

Let us denote F the unary relation $F(x) = \overline{x}$. From [5], lemma 5.30, we have:

Lemma 4 *If a logical relation f is not affine, then $\{f, \mathrm{F}\}$ implements at least one of the three logical relations* OR_0, OR_1 *or* OR_2 *perfectly and faithfully.*

The following lemma, analogue to lemma 5.15 from [5], shows that perfect and faithful implementation provide a mechanism to do projections from the polynomials associated to sets of logical relations.

Lemma 5 *Let S and S' be two sets of logical relations such that every relation of S can be perfectly and faithfully implemented by S'. Then every p-family of polynomials associated to a p-family of S-formulas is a projection of a p-family of polynomials associated to a p-family of S'-formulas.*

Proof. Let (ϕ_n) be a p-family of S-formulas, and let us fix an integer n.

Let $\overline{x} = \{x_1, \ldots, x_p\}$ be the set of variables of the formula ϕ_n. This formula ϕ_n is a conjunction of logical relations $f_i \in S$ applied on variables from $\{x_1, \ldots, x_p\}$. If we replace each of those relations f_i by a perfect and faithful implementation using constraints in S', using for each f_i a new set of auxiliary variables, we obtain a conjunction ψ_n of logical relations from S' applied on variable set $\overline{x} \cup \overline{y}$, where $\overline{y} = \{y_1, \ldots, y_q\}$ is the union of the auxiliary variables sets added for each logical relation f_i.

Since all implementations are perfect and faithful, every assignment to $\overline{x}$ that satisfies all constraints of ϕ_n can be extended by a unique assignment to $\overline{x} \cup \overline{y}$ that satisfies all constraints of ψ_n. Conversely, for an assignment to $\overline{x}$ that does not satisfy all constraints of ϕ_n, no assignment to $\overline{x} \cup \overline{y}$ can extend the previous one and satisfy every constraint of ψ_n.

Since ψ_n is a conjunction of logical relations from S' applied on a set of variables $\overline{x} \cup \overline{y}$, ψ_n is a S'-formula. Furthermore, the number of constraints of ψ_n is bounded by the product of the number of constraints of ϕ_n and the maximum number of logical relations from S' needed to implement a logical

relation from S - which does not depend on n. The size of ψ_n is therefore polynomially linear in the size of ϕ_n. We have:

$$\begin{aligned} P(\phi_n)(X_1,\ldots,X_p) &= \sum_{\overline{\varepsilon}\in\{0,1\}^p} \phi_n(\overline{\varepsilon})\overline{X}^{\overline{\varepsilon}} \\ &= \sum_{\overline{\varepsilon}\in\{0,1\}^p,\overline{y}\in\{0,1\}^q} \psi_n(\overline{\varepsilon},\overline{y})\overline{X}^{\overline{\varepsilon}} \\ &= \sum_{\overline{\varepsilon}\in\{0,1\}^p,\overline{y}\in\{0,1\}^q} \psi_n(\overline{\varepsilon},\overline{y})\overline{X}^{\overline{\varepsilon}}1^{y_1}\ldots 1^{y_q} \\ &= P(\psi_n)(X_1,\ldots,X_p,1,\ldots,1) \end{aligned}$$

Finally, the family $(P(\phi_n))$ is a projection of the family $(P(\psi_n))$, which is a p-family of polynomials associated to S'-formulas. □

From the two previous lemmas, and from the VNP-completeness of families of polynomials associated to $\{\mathrm{OR}_0\}$- , $\{\mathrm{OR}_1\}$- and $\{\mathrm{OR}_2\}$-formulas, we conclude that for every set of logical relations S such that S contains non affine relations, there exists a VNP-complete family of polynomials associated to $S\cup\{\mathrm{F}\}$-formulas. To get rid of the logical relation $\{\mathrm{F}\}$, the authors of [5] need to re-investigate the expressiveness of a non affine relation, and distinguish various cases. For our polynomial problems, we can easily force a boolean variable to be set to false by giving to the associated polynomial variable the value 0. We can now give the proof of Theorem 2:

Proof. Let (ϕ_n) be a p-family of $S\cup\{\mathrm{F}\}$-formulas such that $(P(\phi_n))$ is VNP-complete. The existence of such a family is ensured by lemmas 4 and 5.

Let us consider an integer n. $\phi_n(x_1,\ldots,x_n)$ is a conjunction of logical relations from S applied to variables from $\overline{x}$ and and constraints of the form $(x_i = 0)$. We remark, that if $\phi_n(\overline{x})$ contains the constraint $(x_i = 0)$, then the variable X_i does not appear in the polynomial $P(\phi_n)(X_1,\ldots,X_n)$: all the monomials containing the variable X_i have null coefficients. If we suppress from the conjunction the constraint $(x_i = 0)$, and instead replace the corresponding variable X_i by 0, we obtain exactly the same polynomial: the monomials such that X_i appears in it have null coefficients; the others correspond to assignments such that $x_i = 0$. Let us denote ψ_n the formula obtained by suppressing from ϕ_n all the constraints of the form $(x_i = 0)$.

Since $P(\phi_n)(X_1,\ldots,X_n) = P(\psi_n)(y_1,\ldots,y_n)$, where y_i is 0 if the constraint $(x_i = 0)$ was inserted in ϕ_n, and X_i otherwise, we have, that $(P(\phi_n))$ is a projection from $(P(\psi_n))$. Thus, the family $(P(\psi_n))$ is VNP-complete. Since ψ_n is a S-formula, we have constructed a p-family of S-formulas such that the associated family of polynomials is VNP-complete. □

6 #P-completeness proofs

Up to now, we have studied vertex weighted graphs mostly from the point of view of algebraic complexity theory. Putting weights on edges, or on vertices, can also be useful as an intermediate step in #P-completeness proofs [15, 7]. Here we follow this method to obtain new #P-completeness results. Namely, we prove #P-completeness under many-one reductions for several problems which were only known to be #P-complete under oracle reductions.

Theorem 8 *The following problems are* #P*-complete under many-one reductions.*

1. *Vertex Cover: counting the number of vertex covers of a given a graph.*
2. *Independent Set: counting the number of independent sets of a given graph.*
3. *Bipartite Vertex Cover: the restriction of vertex cover to bipartite graphs.*
4. *Bipartite Independent Set: the restriction of independent set to bipartite graphs.*
5. *Antichain: counting the number of antichains of a given poset.*
6. *Ideal: counting the number of ideals of a given poset.*
7. *Implicative 2-*SAT*: counting the number of satisfying assignments of a conjunction of implicative 2-clauses.*
8. *Positive 2-*SAT*: counting the number of satisfying assignments of a conjunction of positive 2-clauses.*
9. *Negative 2-*SAT*: counting the number of satisfying assignments of a conjunction of negative 2-clauses.*

Remark 1 #P*-completeness under oracle reductions is established in [12] for the first six problems, in [10] for the 7th problem and in [16] for the last two. In Section 2, the last three problems are denoted* #SAT(S) *where* S *is respectively equal to* $\{\mathrm{OR}_1\}$*,* $\{\mathrm{OR}_0\}$ *and* $\{\mathrm{OR}_2\}$*.*

Proof. Provan and Ball establish in [12] the equivalence of Problems 1 and 2, 3 and 4, and 5 and 6; they produce many-one reductions from 1 to 8 and from 4 to 5, and Linial gives in [10] a many-one reduction from 6 to 7. Problems 8 and 9 are clearly equivalent.

Therefore, to obtain #P-completeness under many-one reductions for all those problems, we just need to show the #P-completeness of Problem 1 and

to produce a many-one reduction from problem 1 to Problem 3 (replacing the oracle reduction from [12]).

In order to prove the #P-completeness of Problem 1, we first establish a many-one reduction from the #P-complete problem of computing the permanent of $\{0,1\}$-matrices to the problem of computing the vertex cover polynomial of a weighted graph with weights in $\{0,1,-1\}$. In [2], Bürgisser attributes to Jerrum a projection from the permanent to the partial permanent, with the use of the constant -1. Applied to a $\{0,1\}$-matrix, this gives a many-one reduction from the permanent on $\{0,1\}$-matrices to the partial permanent on $\{0,1,-1\}$-matrices. By Theorem 5, the $n \times n$ partial permanent is equal to the independent set polynomial of the graph G'_n, which is computable in time polynomial in n. Moreover, by Lemma 1 this polynomial is the projection of the vertex cover polynomial of G_n, with the use of the constant -1. The partial permanent on entries in $\{0,1,-1\}$ therefore reduces to the vertex cover polynomial on graphs with weights in $\{0,1,-1\}$.

Let G be such a vertex weighted graph, with weights in $\{0,1,-1\}$. A vertex cover of nonzero weight does not contain any vertex of weight 0, and in order to cover the edges that are incident to such a vertex it must contain every vertex u that is adjacent to a vertex of weight 0. One can therefore remove all vertices of weight 0, and replace each edge from such a vertex v to another vertex u by a self-loop (an edge from u to u). This transformation does not change the vertex cover polynomial, and it is always feasible since there are no adjacent vertices of weight zero in the graph G obtained by the projection from Lemma 1. Thus, we obtain a graph G' with weights in $\{1,-1\}$ such that $\mathrm{VCP}(G) = \mathrm{VCP}(G')$.

To deal with the weights -1, we use a method similar to [15]. Since $\mathrm{VCP}(G')$ is the value of a permanent on a $\{0,1\}$-matrix, it is positive. We will construct an integer N and a graph H such that the number of vertex covers of H modulo N is equal to $\mathrm{VCP}(G')$. This will establish a reduction from the boolean permanent to counting vertex covers.

We choose N larger than the maximum value of the number of vertex covers of G': $N = 2^{v(G')} + 1$ will suit our purposes. Now that we compute the number of vertex covers modulo N, we can replace each -1 weight in G' by the weight $N - 1 = 2^{v(G')}$. As shown in Figure 2, we can simulate such a weight on a vertex by adding $v(G')$ leaves for each vertex of weight -1.

Finally, to prove the #P-completeness of Problem 3 we construct a reduction from vertex cover to bipartite vertex cover. From Lemma 2, we have a projection from the vertex cover polynomial of a graph to the vertex cover of a bipartite graph, with the use of -1 weights. To eliminate these weights, we can follow the method used in our above proof of the #P-completeness of Problem 1. Indeed, since the leaves added to the graph preserve bipartiteness, we obtain a reduction from counting vertex covers in a general graph to counting vertex covers in a bipartite graph. □

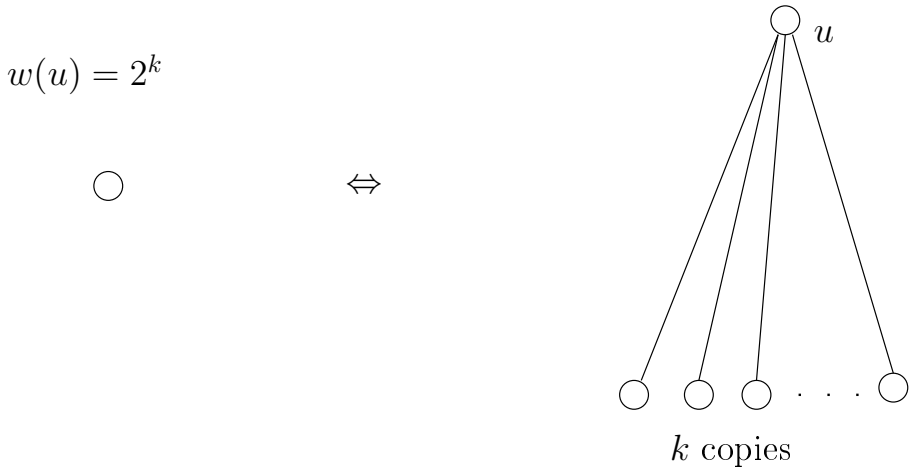

Figure 2: Simulation of a weight 2^k

The proof of Creignou and Hermann's dichotomy theorem [4, 5] is based on many-one reductions from the last 3 problems of Theorem 8. We have just shown that these 3 problems are #P-complete under many-one reductions. As a result, we have the following corollary to Theorem 8.

Corollary 2 *Theorem 1 still holds for* #P*-completeness under many-one reduction.*

References

[1] P. Bürgisser. On the structure of Valiant's complexity classes. *Discrete Mathematics and Theoretical Computer Science*, 3:73–94, 1999.

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

[3] A. Bulatov. A dichotomy theorem for contraint satisfaction problems on a 3-element set. *Journal of the ACM*, 53(1):66–120, 2006.

[4] N. Creignou and M. Hermann. Complexity of generalized satisfiability counting problems. *Information and Computation*, 125:1–12, 1996.

[5] N. Creignou, S. Khanna, and M. Sudan. *Complexity classification of boolean constraint satisfaction problems*. SIAM monographs on discrete mathematics. 2001.

[6] F. M. Dong, M. D. Hendy, K. L. Teo, and C. H. C. Little. The vertex-cover polynomial of a graph. *Discrete Mathematics*, 250(1-3):71–78, 2002.

[7] M. Jerrum. Two-dimensional monomer-dimer systems are computationally intractable. *Journal of Statistical Physics*, 48:121–134, 1987.

[8] M. Jerrum. *Counting, Sampling and Integrating : Algorithms and Complexity*. Lectures in Mathematics - ETH Zürich. Birkhäuser, Basel, 2003.

[9] P. Koiran and K. Meer. On the expressive power of CNF formulas of bounded tree- and clique- width. In *Proceedings of WG'08 (34th International Workshop on Graph-Theoretic Concepts in Computer Science), LNCS 5344*. Springer, 2008.

[10] N. Linial. Hard enumeration problems in geometry and combinatorics. *SIAM Journal of Algebraic and Discrete Methods*, 7(2):331–335, 1986.

[11] M. Lotz and J. A. Makowsky. On the algebraic complexity of some families of coloured Tutte polynomials. *Advances in Applied Mathematics*, 32(1):327–349, January 2004.

[12] J. S. Provan and M. O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal of Computing*, 12(4):777–788, 1983.

[13] T. J. Schaefer. The complexity of satisfiability problems. In *Conference Record of the 10th Symposium on Theory of Computing*, pages 216–226, 1978.

[14] L. G. Valiant. Completeness classes in algebra. In *Proc. 11th ACM Symposium on Theory of Computing*, pages 249–261, 1979.

[15] L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.

[16] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal of Computing*, 8(3):410–421, 1979.

A The case of affine constraints

Recall that an affine formula is a conjunction of constraints of the form $a_1 \oplus \cdots \oplus a_p = 0$ or $a_1 \oplus \cdots \oplus a_p = 1$, where the a_i's are boolean variables. In this section we study S-formulas where S contains only affine constraints.

A conjunction of affine formulas may be seen as a system of linear equations over the field $\mathbb{Z}/2\mathbb{Z}$. The number of satisfying assignments of the conjunction is computable in polynomial time by e.g. Gaussian elimination.

More generally, we would like to know whether the evaluation of the polynomials $P(\phi_n)$ associated to a p-family of affine formulas is in VP. This problem remains open, but we provide efficient evaluation algorithms in a couple of cases.

A.1 Evaluation at special points

Here we show that $P(\phi_n)$ can be evaluated efficiently when its variables take values in $\{-1, 0, 1\}$.

Let $(\phi_n)_{n\in\mathbb{N}}$ be a family of affine formulas of polynomial size. Let us fix an integer n, and let m the number of variables of ϕ_n. As pointed out above, we can solve the formula ϕ_n in polynomial time. The solution set S is an affine subspace of $\{0,1\}^n$. Its cardinality is 2^k, where k is the dimension of S. Thus we know how to compute $P(\phi_n)$ when all variables take the value 1. To allow values from $\{0, 1\}$, we can simply add the constraint $\varepsilon_i = 0$ for each X_i which takes a zero value. This brings us back to the previous case.

Next we explain how to deal with values from $\{-1, 1\}$. Since ϕ_n accepts only the elements of S, we have:

$$P(\phi_n)(X_1, \ldots, X_m) = \sum_{\overline{\varepsilon}\in\{0,1\}^m} \phi(\overline{\varepsilon})\overline{X}^{\overline{\varepsilon}} = \sum_{\overline{\varepsilon}\in S} \overline{X}^{\overline{\varepsilon}}$$

Let $a = (a_1, \ldots, a_m)$ be an element of S, let k be the dimension of S, and let $(e_1, \ldots, e_k)$ be a basis of the corresponding linear subspace. For all i in $[0, k]$, we denote $e_i = (e_{i,1}, \ldots, e_{i,k})$. We have

$$P(\phi_n)(X_1, \ldots, X_m) = \sum_{\eta\in\{0,1\}^k} X_1^{a_1\oplus\eta_1 e_{1,1}\oplus\cdots\oplus\eta_k e_{k,1}} \cdots X_m^{a_m\oplus\eta_1 e_{1,m}\oplus\cdots\oplus\eta_k e_{k,m}}$$

Since this expression has an exponential number of monomials, it is not computable directly. We do not know how to evaluate it efficiently in general, but the problem becomes easier in the case of entries in $\{1, -1\}$. In this case, each X_i verifies the equation $X_i^2 = 1$. Thus,

$$X_i^{a_1\oplus\eta_1 e_{1,i}\oplus\cdots\oplus\eta_k e_{k,i}} = X_i^{a_1+\eta_1 e_{1,1}+\cdots+\eta_k e_{k,1}}.$$

The polynomial $P(\phi_n)$ may be factored as follows:

$$P(\phi_n) = (\prod_{i=1\ldots m} X_i^{a_i}) \cdot (\prod_{i=1\ldots k} \sum_{\eta_i \in \{0,1\}} X_1^{\eta_i e_{i,1}} \cdots X_m^{\eta_i e_{i,m}})$$

and this expression can be evaluated efficiently.

Finally, to deal with entries from $\{-1, 0, 1\}$ we can use the trick explained at the beginning of this subsection: adding the constraint $\varepsilon_i = 0$ for each variable X_i which takes the value zero brings us back to the case of $\{-1, 1\}$-valued variables.

A.2 Few linear equations

A second case where we have an efficient evaluation algorithm for $P(\phi_n)$ is when S can be defined by a "small" number of affine equations. For the sake of clarity, consider first the case where ϕ is defined by a single affine constraint $\alpha_1 x_1 \oplus \cdots \oplus \alpha_m x_m = b$, where $\alpha_i \in \{0, 1\}$. We can compute recursively the polynomials P_i^1 and P_i^0 associated respectively to the two constraints

$$\alpha_1 x_1 \oplus \cdots \oplus \alpha_i x_i = 1,\ \alpha_1 x_1 \oplus \cdots \oplus \alpha_i x_i = 0.$$

Indeed, for $\alpha_i = 1$ we have

$$P_i^1(X_1, \ldots, X_i) = X_i \times P_{i-1}^0(X_1, \ldots, X_{i-1}) + P_{i-1}^1(X_1, \ldots, X_{i-1})$$

and

$$P_i^0(X_1, \ldots, X_i) = X_i \times P_{i-1}^1(X_1, \ldots, X_{i-1}) + P_{i-1}^0(X_1, \ldots, X_{i-1}).$$

For $\alpha_i = 0$ and any $\varepsilon \in \{0, 1\}$, we have

$$P_i^\varepsilon(X_1, \ldots, X_i) = (1 + X_i) P_{i-1}^\varepsilon(X_1, \ldots, X_{i-1}).$$

By dynamic programming, we can evaluate our polynomial in $O(m)$ arithmetic operations. This method is still valid with a conjunction of $r > 1$ affine constraints, but now we need to compute the intermediate polynomials for all combination of possible constants $(b_1, \ldots, b_r) \in \{0, 1\}^r$ in the right side of the linear equations defining S. So this method is efficient only when the number r of linear equations grows logarithmically.

B The permutation function is not affine

Let $\phi_n = \text{PERMUTATION}_n$ be the boolean function in n^2 variables that accepts the $n \times n$ permutation matrices. This function is the coefficient function of the permanent. Namely, for a matrix $M = (X_{i,j})_{(i,j)\in[1,n]^2}$ we have

$$\text{PER}(M) = \sum_{\varepsilon\in\{0,1\}^{n\times n}} \phi_n(\overline{\varepsilon})\overline{X}^{\overline{\varepsilon}} \tag{6}$$

where as usual $\overline{X}^{\overline{\varepsilon}}$ represents the product $\prod_{(i,j)\in[1,n]^2} X_{i,j}^{\varepsilon_{i,j}}$.

In this section we show that ϕ_n is not affine (recall that a boolean function is affine if it expressible as a system of affine equations over $\mathbb{Z}/2\mathbb{Z}$). This is to be expected since, by (6), the results of section A.1 would otherwise lead to an efficient evaluation algorithm for the 0/1 permanent. The goal of this section is to give an unconditional proof of the fact that ϕ_n is not affine.

Proposition 1 *For $n \geq 3$, PERMUTATION_n is not an affine function.*

Proof. Let `1-in-3`(x, y, z) be the boolean function which takes the value 1 when exactly one of its 3 variables is equal to 1. We will show that this function is a projection of $\phi_n = \text{PERMUTATION}_n$, but that it is not the projection of any affine function.

To obtain the first part of this claim, denote by $\phi'(x, y, z)$ the boolean function obtained by applying ϕ_n to the boolean matrix

$$\begin{pmatrix} x & y & z & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & & \ddots & \ddots & \ddots & z \\ z & 0 & & \ddots & \ddots & y \\ y & z & 0 & \cdots & 0 & x \end{pmatrix}.$$

Since a permutation matrix has exactly one 1 in each row and each column, ϕ' will accept exactly the entries $(1, 0, 0)$, $(0, 1, 0)$ or $(0, 0, 1)$. Hence ϕ' is nothing but the `1-in-3` function.

On the other hand, let $\psi'(x, y, z)$ be a projection of a boolean function ψ: this means that ψ' is obtained from ψ by replacing its variables by variables from $\{x, y, z\}$, or by boolean constants. If ψ is affine, ψ' must be affine too. The set of its satisfying assignements is therefore a linear space over $\mathbb{Z}/2\mathbb{Z}$, and its cardinality must be a power of two. As a result, such a ψ' cannot accept the set $\{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}$. □

The bound $n \geq 3$ in this proposition is optimal since ϕ_2 is an affine function. Note also that when the matrix used in the definition of ϕ' is the matrix of a permutation, it is also the matrix of a Hamilton cycle; when it is not, it is not even the matrix of an injective partial map from $[1, n]$ to $[1, n]$. The same reasoning therefore provides two other examples of non affine functions: the boolean function that accepts the matrices of Hamilton cycles, and the boolean function that accepts the matrices of injective maps from $[1, n]$ to $[1, n]$ (these boolean functions are the coefficient functions of the Hamilton cycle polynomial and of the partial permanent).