On the expressive power of planar perfect matching and permanents of bounded treewidth matrices

Uffe Flarup[1], Pascal Koiran[2] and Laurent Lyaudet[2]

[1] Department of Mathematics and Computer Science
Syddansk Universitet, Campusvej 55, 5230 Odense M, Denmark
e–mail: {flarup}@imada.sdu.dk; fax: +45 65 93 26 91
[2] Laboratoire de l'Informatique du Parallélisme**
Ecole Normale Supérieure de Lyon, 46, allée d'Italie, 69364 Lyon Cedex 07, France
e–mail: {pascal.koiran,laurent.lyaudet}@ens-lyon.fr; fax: +33 4 72 72 80 80

Abstract. Valiant introduced some 25 years ago an algebraic model of computation along with the complexity classes VP and VNP, which can be viewed as analogues of the classical classes P and NP. Prominent examples of difficult (that is, VNP-complete) problems in this model includes the permanent and hamiltonian polynomials. In this paper we investigate the expressive power of easy special cases of these polynomials. We show that the permanent and hamiltonian polynomials for matrices of bounded treewidth both are equivalent to arithmetic formulas. Also, arithmetic weakly skew circuits are shown to be equivalent to the sum of weights of perfect matchings of planar graphs.

1 Introduction

Our focus in this paper is on easy special cases of otherwise difficult to evaluate polynomials, and their connection to various classes of arithmetic circuits. In particular we consider the permanent and hamiltonian polynomials for matrices of bounded treewidth, and sum of weights of perfect matchings for planar graphs. It is a widely believed conjecture that the permanent is hard to evaluate. Indeed, in Valiant's framework [15, 16] the permanent is complete for the class VNP. This is an algebraic analogue of his ♯P-completeness result for the permanent [14]. For a book-length treatment of Valiant's algebraic complexity theory one may consult [4]. The same results (♯P-completeness in the boolean framework, and VNP-completeness in the algebraic framework) also apply to the hamiltonian polynomial. The sum of weights of perfect matchings in an (undirected) graph G is yet another example of a presumably hard to compute polynomial since it reduces to the permanent when G is bipartite. However, all three polynomials are known to be easy to evaluate in special cases. In particular, the permanent and hamiltonian polynomials can be evaluated in a polynomial

** UMR 5668 ENS Lyon, CNRS, UCBL, INRIA.

number of arithmetic operations for matrices of bounded treewidth [6]. An earlier result of this flavour is Kasteleyn's theorem [7] which states that the sum of weights of perfect matchings of a planar graph can be computed in a polynomial number of arithmetic operations. One can try to turn these three efficient algorithms into general-purpose evaluation algorithms by means of reductions (this is the approach followed by Valiant in [17], where he exhibits polynomial time algorithms for several problems which previously only had exponential time algorithms, by means of holographic reductions to perfect matchings of planar graphs). For instance, in order to evaluate a polynomial P one can try to construct a matrix of bounded treewidth A such that: Entries of A are constants or variables of P, and the permanent of A is equal to P.

The same approach can be tried for the hamiltonian and the sum of weights of perfect matchings in a planar graph. The goal of this paper is to assess the power of these polynomial evaluation methods. It turns out that the three methods are all universal - that is, every polynomial can be expressed as the sum of weights of perfect matchings in a planar graph, and as a permanent and hamiltonian of matrices of bounded treewidth. ¿From a complexity-theoretic point of view, these methods are no longer equivalent. Our main findings are that:

- The permanents and hamiltonians of matrices of polynomial size and bounded treewidth have the same expressive power, namely, the power of polynomial size arithmetic formulas. This is established in Theorem 1.
- The sum of weights of perfect matchings in polynomial size planar graphs has at least the same power as the above two representations, and in fact it is more powerful under a widely believed conjecture. Indeed, this representation has the same power as polynomial size (weakly) skew arithmetic circuits. This is established in Theorem 7. We recall that (weakly) skew arithmetic circuits capture the complexity of computing the determinant [13]. It is widely believed that the determinant cannot be expressed by polynomial size arithmetic formulas.

Our three methods therefore capture (presumably proper) subsets of the class VP of easy to compute polynomial families. By contrast, if we drop the bounded treewidth or planarity assumptions, the class VNP is captured in all three cases.

Various notions of graph "width" have been defined in the litterature besides treewidth (e.g. pathwidth, cliquewidth, rankwidth). They should be worth studying from the point of view of their expressive power. Also, Barvinok [1] has shown that if the underlying matrix has bounded rank, both the permanent and the hamiltonian polynomials can be evaluated in a polynomial number of arithmetic operations.

2 Definitions

2.1 Arithmetic circuits

Definition 1. *An arithmetic circuit is a finite, acyclic, directed graph. Vertices have indegree 0 or 2, where those with indegree 0 are referred to as* inputs. *A* *single vertex must have outdegree 0, and is referred to as* output. *Each vertex of indegree 2 must be labeled by either + or ×, thus representing computation. Vertices are commonly referred to as* gates *and edges as* arrows.

By interpreting the input gates either as constants or variables it is easy to prove that each arithmetic circuit naturally represents a polynomial.

In this paper various subclasses of arithmetic circuits will be considered: For *weakly skew* circuits we have the restriction that for every multiplication gate, at least one of the incoming arrows is from a subcircuit whose only connection to the rest of the circuit is through this incoming arrow. For *skew* circuits we have the restriction that for every multiplication gate, at least one of incoming arrows is from an input gate. For *formulas* all gates (except output) have outdegree 1. Thus, reuse of partial results is not allowed. For a detailed description of various subclasses of arithmetic circuits, along with examples, we refer to [12].

Definition 2. *The* size *of a circuit is the total number of* gates *in the circuit. The* depth *of a circuit is the length of the longest path from an input gate to the output gate.*

A family (f_n) belongs to the complexity class VP if f_n can be computed by a circuit C_n of size polynomial in n, and if moreover the degree of f_n is bounded by a polynomial function of n.

2.2 Treewidth

Treewidth for undirected graphs is most commonly defined as follows:

Definition 3. *Let $G = \langle V, E\rangle$ be a graph. A k-tree-decomposition of G is a tree $T = \langle V_T, E_T\rangle$ such that:*

(i) For each $t \in V_T$ there is a subset $X_t \subseteq V$ of size at most $k + 1$.
(ii) For each edge $(u, v) \in E$ there is a $t \in V_T$ such that $\{u, v\} \subseteq X_t$.
(iii) For each vertex $v \in V$ the set $\{t \in V_T | v \in X_t\}$ forms a subtree of T.

The treewidth of G is the smallest k s.t. there exists a k-tree-decomposition for G.

An equivalent definition is in terms of graph grammars (HR algebras [5]):

Definition 4. *A graph G has a k-tree-decomposition iff there exist a set of source labels of cardinality $k + 1$ such that G can be constructed using a finite number of the following operations:*

(i) ver_a, $loop_a$, $edge_{ab}$ (basic constructs: create a single vertex with label a, a single vertex with label a and a looping edge, two vertices labeled a and b connected by an edge)
(ii) $ren_{a\leftrightarrow b}(G)$ (rename all labels a as labels b and all labels b as labels a)
(iii) $forg_a(G)$ (forget all labels a)
(iv) $G_1 \mathbin{//} G_2$ (composition of graphs: two vertices with same label are identified as one vertex)

The treewidth of a directed graph is defined as the treewidth of the underlying undirected graph. The treewidth of an $(n \times n)$ matrix $M = (m_{i,j})$ is defined as the treewidth of the directed graph $G_M = \langle V_M, E_M, w \rangle$ where $V_M = \{1, \ldots, n\}$, $(i, j) \in E_M$ iff $m_{i,j} \neq 0$, and $w(i, j) = m_{i,j}$.

2.3 Permanent and hamiltonian polynomials

We take a graph theoretic approach to deal with permanent and hamiltonian polynomials. The reason for this being that a natural way to define the treewidth of a matrix, is by the treewidth of the underlying graph, see also e.g. [10].

Definition 5. *A* cycle cover *of a directed graph is a subset of the edges, such that these edges form disjoint, directed cycles (loops are allowed). Furthermore, each vertex in the graph must be in one (and only one) of these cycles. The* weight *of a cycle cover is the* product *of weights of all participating edges.*

Definition 6. *The* permanent *of an $(n \times n)$ matrix $M = (m_{i,j})$ is the sum of weights of all cycle covers of G_M.*

The *hamiltonian* polynomial $\mathrm{ham}(M)$ is defined similarly, except that we only sum over cycle covers consisting of a *single* cycle (hence the name).

3 Matrices of bounded treewidth

In this section we work with directed graphs. All paths and cycles are assumed to be directed, even if this word is omitted.

In [6] it is shown that the permanent and hamiltonian polynomials are in VP for matrices of bounded treewidth. Here we show that both the permanent and hamiltonian polynomials for matrices of bounded treewidth are equivalent to arithmetic formulas. This is an improvement on the result of [6] since the set of polynomial families representable by polynomial size arithmetic formulas is a (probably strict) subset of VP.

Theorem 1. *Let (f_n) be a family of polynomials with coefficients in a field K. The three following properties are equivalent:*

- *(f_n) can be represented by a family of polynomial size arithmetic formulas.*
- *There exists a family (M_n) of poly. size, bounded treewidth matrices such that entries of M_n are constants from K or variables of f_n, and $f_n = \mathrm{per}(M_n)$.*
- *There exists a family (M_n) of poly. size, bounded treewidth matrices such that entries of M_n are constants from K or variables of f_n, and $f_n = \mathrm{ham}(M_n)$.*

Theorem 1 follows immediately from Theorems 2, 3, 5 and 6.

Theorem 2. *Every arithmetic formula can be expressed as the permanent of a matrix of treewidth at most 2 and size at most $(n + 1) \times (n + 1)$ where n is the size of the formula. All entries in the matrix are either 0, 1, or variables.*

Proof. The first step is to construct a directed graph that is a special case of a *series-parallel* (SP) graph, in which there is a connection between weights of directed paths and the value computed by the formula. The overall idea behind the construction is quite standard, see e.g. [4] and [12]. SP graphs in general can between any two adjacent vertices have multiple directed edges. But we construct an SP graph in which there is at most one directed edge from any vertex u to any vertex v. This property will be needed in the second step, in which a connection between cycle covers and the permanent of a given matrix will be established.

SP graphs have distinguished *source* and *sink* vertices, denoted by s and t. By $SW(G)$ we denote the sum of weights of all directed paths from s to t, where the weight of a path is the *product* of weights of participating edges.

Let φ be a formula of size e. For the first step of the proof we will by induction over e construct a weighted, directed SP graph G such that $val(\varphi) = SW(G)$. For the base case $\varphi = w$ we construct vertices s and t and connect them by a directed edge from s to t with weight w.

Assume $\varphi = \varphi_1 + \varphi_2$ and let G_i be the graph associated with φ_i by the induction hypothesis. Introduce one new vertex s and let G be the union of the three graphs $\langle\{s\}\rangle$, G_1 and G_2 in which we identify t_1 with t_2 and denote it t, add an edge of weight 1 from s to s_1, and add an edge of weight 1 from s to s_2. By induction hypothesis the resulting graph G satisfies $SW(G) = 1 \cdot SW(G_1) + 1 \cdot SW(G_2) = val(\varphi_1) + val(\varphi_2)$. Between any two vertices u and v there is at most one directed edge from u to v. We introduced one new vertex, but since t_1 was identified with t_2 the number of vertices used equals $|V_1|+|V_2| \leq size(\varphi_1) + 1 + size(\varphi_2) + 1 = size(\varphi) + 1$.

Assume $\varphi = \varphi_1 * \varphi_2$. We construct G by making the disjoint union of G_1 and G_2 in which we identify t_1 with s_2, identify s_1 as s in G and identify t_2 as t in G. For every directed path from s_1 to t_1 in G_1 and for every directed path from s_2 to t_2 in G_2 we can find a directed path from s to t in G of weight equal to the product of the weights of the paths in G_1 and G_2, and since all (s, t) paths in G are of this type we get $SW(G) = SW(G_1) \cdot SW(G_2)$. The number of vertices used equals $|V_1| + |V_2| - 1 \leq size(\varphi_1) + size(\varphi_2) + 1 < size(\varphi) + 1$.

For the second step of the proof we need to construct a graph G' such that there is a relation between cycle covers in G' and directed paths from s to t in G. We construct G' by adding an edge of weight 1 from t back to s, and loops of weight 1 at all vertices different from s and t. Now, for every (s, t) path in G we can find a cycle in G' visiting the corresponding nodes. For nodes in G' not in this cycle, we include them in a cycle cover by the loops of weight 1. Because there is at most one directed edge from any vertex u to any vertex v in G' we can find a matrix M of size at most $(n + 1) \times (n + 1)$ such that $G_M = G'$ and $per(M) = val(\varphi)$. It can be shown that G' can be constructed using an HR algebra with only 3 source labels. □

Theorem 3. *Every arithmetic formula of size n can be expressed as the hamiltonian of a matrix of treewidth at most 6 and size at most $(2n + 1) \times (2n + 1)$. All entries in the matrix are either 0, 1, or variables of the formula.*

Proof. The first step is to produce the graph G as shown in Theorem 2. The next step is to show that the proof of universality for the hamiltonian polynomial in [11] can be done with treewidth at most 6. Their construction for universality of the hamiltonian polynomial introduces $|V_G| - 1$ new vertices to G in order to produce G', along with appropriate directed edges (all of weight 1). The proof is sketched in Figure 1.

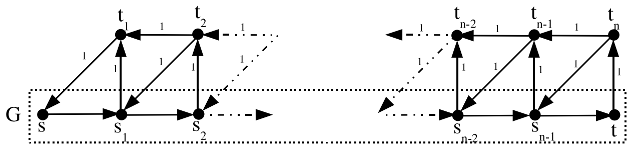

Fig. 1. Universality of the hamiltonian polynomial

The additional vertices t_i and edges permit to visit any subset of vertices of G with a directed path of weight 1 from t to s using all t_i's. Hence, any path from s to t in G can be followed by a path from t to s to obtain a hamiltonian cycle of same weight. If one just need to show universality, then it is not important exactly which one of the vertices t_i that has an edge to a given vertex among s_i. But in order to show bounded treewidth one carefully need to take into account which one of the vertices of t_i that has an edge to a particular s_i vertex. We show such a construction with bounded treewidth, by giving an HR algebra which can express a graph similar to the one in Figure 1 using 7 source labels.

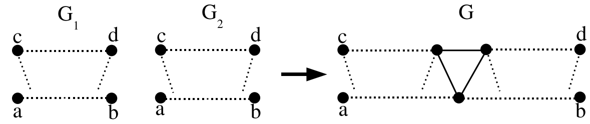

Fig. 2. Series composition (simulating multiplication)

Series composition is done using the following operations (also see Figure 2):

$$forg_e[forg_f[forg_g[ren_{d\leftrightarrow f}(ren_{b\leftrightarrow e}(G_1)) \mathbin{//} ren_{c\leftrightarrow g}(ren_{a\leftrightarrow e}(G_2)) \mathbin{//} edge_{ef} \mathbin{//} edge_{eg} \mathbin{//} edge_{fg}]]]$$

Labels a, b, c and d in Figures 2 and 3 plays the roles of s, t, t_1 and t_n respectively in Figure 1. The above construction does not take into account, that G_1 and/or G_2 are graphs generated from the base case. For base cases vertices c and d are replaced by a single vertex. However, it is clear that the above construction can be modified to work for these simpler cases as well.

For parallel composition an additional vertex was introduced. It can be done using the following operations (also see Figure 3):

$$\begin{array}{l} forg_f[ren_{a\leftrightarrow e}(ren_{c\leftrightarrow g}(\\ \quad forg_a[forg_c[edge_{ag} \mathbin{//} edge_{cg} \mathbin{//} ren_{d\leftrightarrow f}(edge_{ae} \mathbin{//} edge_{ac} \mathbin{//} G_1)]] \mathbin{//} \\ \quad forg_a[forg_c[edge_{af} \mathbin{//} edge_{cf} \mathbin{//} edge_{ae} \mathbin{//} edge_{ac} \mathbin{//} G_2]] \\ ))] \end{array}$$

The final step in the construction, after all series and parallel composition have been done, is to connect vertices a and c and connect vertices b and d. $\square$

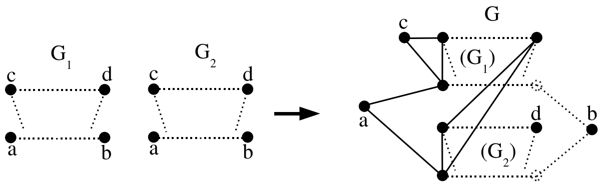

Fig. 3. Parallel composition (simulating addition)

By Definition 6, computing the permanent of a matrix M amounts to computing the sum of the weights of all cycle covers of G_M. In our algorithm we need to consider partial covers, which are generalizations of cycle covers.

Definition 7. *A partial cover of a directed graph is a union of paths and cycles such that every vertex of the graph belongs to at most one path (and to none of the cycles), or to at most one cycle (and to none of the paths). The weight of a partial cover is the product of the weights of all participating edges. More generally, for any set S of edges the weight $w(S)$ of S is defined as the product of the weights of the elements of S.*

In contrast to cycle covers, for a partial cover there is no requirement that all vertices be covered.

The following theorem from [2] is a standard tool in the design of parallel algorithms for graphs of bounded treewidth (see also [3] and [8]).

Theorem 4. *Let $G = \langle V, E\rangle$ be a graph of treewidth k with n vertices. Then there exists a tree-decomposition $\langle T, (X_t)_{t\in V_T}\rangle$ of G of width $3k+2$ such that $T = \langle V_T, E_T\rangle$ is a binary tree of depth at most $2\lceil log_{\frac{5}{4}}(2n)\rceil$.*

Theorem 5. *The permanent of a $n \times n$ matrix M of bounded treewidth k can be expressed as a formula of size $O(n^{O(1)})$.*

Proof. We show how to construct a circuit of depth $O(log(n))$, which can then be expressed as a formula of size $O(n^{O(1)})$. Consider the graph $G = G_M$ and apply Theorem 4 to obtain a balanced, binary tree-decomposition T of bounded width k'. For each node t of T, we denote by T_t the subtree of T rooted at t, and we denote by $X(T_t)$ the set of vertices of G which belong to X_u for at least one of the nodes u of T_t. We denote by G_t the subgraph of G induced by the subset of vertices $X(T_t)$.

Consider a partial cover C of G_t. Any given edge $(u, v) \in X_t^2$ is either used or unused by C. Likewise, any given vertex of X_t has indegree 0 or 1 in C, and outdegree 0 or 1. We denote by $\lambda_t = I_t(C)$ the list of all these data for every edge $(u, v) \in X_t^2$ and every element of X_t. By abuse of language, we will say that an edge in X_t^2 is used by λ_t if it is used by one partial cover satisfying $I_t(C) = \lambda_t$ (or equivalently, by all partial cover satisfying $I_t(C) = \lambda_t$).

We will compute for each possible list λ_t a weight $w(\lambda_t)$, defined as the sum of the weights of all partial covers C of G_t satisfying the following three properties:

(i) the two endpoints of all paths of C belong to X_t;
(ii) all uncovered vertices belong to X_t;
(iii) $I_t(C) = \lambda_t$.

Note that the number of weights to be computed at each node of T is bounded by a constant (which depends on k'). When t is the root of T we can easily compute the permanent of M from the weights $w(\lambda_t)$: it is equal to the sums of the $w(\lambda_t)$ over all λ_t which assign indegree 1 and outdegree 1 to all vertices of X_t. Also, when t is a leaf of T we can compute the weights in a constant number of arithmetic operations since G_t has at most k' vertices in this case. It therefore remains to explain how to compute the weights $w(\lambda_t)$ when t is not a leaf.

Our algorithm for this proceeds in a bottom-up manner: we will compute the weights for t from the weights already computed for its left child (denoted l) and its right child (denoted r). The idea is that we can obtain a partial cover of G_t by taking the union of a partial cover of G_l and of a partial cover of G_r, and adding some additional edges. Conversely, a partial cover of G_t induces a partial cover of G_l and a partial cover of G_r. In order to avoid counting many times the same partial cover, we must define the considered partial covers of G_l and G_r to ensure that the partial cover of G_t induces a unique partial cover of G_l and a unique partial cover of G_r. We will say that (λ_l, λ_r) is compatible with λ_t if and only if the following holds:

- no edge in X_t^2 is used in λ_l or λ_r;
- for every vertex $x \in X_t$ at most one of λ_l, λ_r assigns indegree 1 to x;
- for every vertex $x \in X_t$ at most one of λ_l, λ_r assigns outdegree 1 to x;
- for every vertex $x \in X_t$ if λ_l or λ_r assigns indegree 1 to x then λ_t assigns indegree 1 to x and no edge entering x is used by λ_t ;
- for every vertex $x \in X_t$ if λ_l or λ_r assigns outdegree 1 to x then λ_t assigns outdegree 1 to x and no edge leaving x is used by λ_t ;
- every vertex $x \in X_l \backslash X_t$ has indegree 1 and outdegree 1 in λ_l;
- every vertex $x \in X_r \backslash X_t$ has indegree 1 and outdegree 1 in λ_r.

We now have to prove two things. If there is a partial cover C of G_t which satisfies the properties (i) and (ii) such that $I_t(C) = \lambda_t$ then it induces a partial cover C_l of G_l and a partial cover C_r of G_r such that C_l and C_r satisfy (i) and (ii), $I_l(C) = \lambda_l$, $I_r(C) = \lambda_r$, and (λ_l, λ_r) is compatible with λ_t. Conversely, if (λ_l, λ_r) is compatible with λ_t, and C_l and C_r are partial covers of G_l and G_r satisfying (i), (ii), $I_l(C) = \lambda_l$, and $I_r(C) = \lambda_r$, then there exists a unique partial cover C of G_t containing C_l and C_r such that $I_t(C) = \lambda_t$.

Consider a partial cover C of G_t which satisfies the properties (i) and (ii) defined above. We can assign to C a unique triple (C_l, C_r, S) defined as follows. First, we define S as the set of edges of $C \cap X_t^2$. Then we define C_l as the set of edges of C which have their two endpoints in $X(T_l)$, and at least one of them outside of X_t. Finally, we define C_r as the set of edges of C which have their two endpoints in $X(T_r)$, and at least one of them outside of X_t. Note that $w(C) = w(C_l) \cdot w(C_r) \cdot w(S)$ since (C_l, C_r, S) forms a partition of the edges of C. Moreover, C_l is a partial cover of G_l and properties (i) and (ii) are satisfied: the endpoints of the paths of C_l and the uncovered vertices of $X(T_l)$ all belong to $X_l \cap X_t$. Likewise, C_r is a partial cover of $X(T_r)$ and properties (i) and (ii) are satisfied. If $I_l(C) = \lambda_l$ and $I_r(C) = \lambda_r$, it is clear that (λ_l, λ_r) is compatible with λ_t. Any other partition of C in three parts with one partial cover of G_l, one partial cover of G_r, and a subset of edges in X_t^2 would have an edge of X_t^2 used by C_l or C_r. Hence (λ_l, λ_r) would not be compatible with λ_t.

Suppose now (λ_l, λ_r) is compatible with λ_t, and C_l and C_r are partial covers of G_l and G_r satisfying (i), (ii), $I_l(C) = \lambda_l$, and $I_r(C) = \lambda_r$. We define S_{λ_t} as the set of edges of X_t^2 which are used by λ_t. It is clear that S_{λ_t}, C_l and C_r are disjoint. Consider $C = S_{\lambda_t} \cup C_l \cup C_r$. Since (λ_l, λ_r) is compatible with λ_t, C is a partial cover satisfying (i) and (ii). It is also clear that C is the only partial cover containing C_l and C_r such that $I_t(C) = \lambda_t$. This leads to the formula:

$$w(\lambda_t) = \sum_{(\lambda_l, \lambda_r)} w(\lambda_l) \cdot w(\lambda_r) \cdot w(S_{\lambda_t}).$$

The sum runs over all pairs (λ_l, λ_r) that are compatible with λ_t. The weight $w(\lambda_t)$ can therefore be computed in a constant number of arithmetic operations. Since the height of T is $O(log(n))$ the above algorithm can be executed on a circuit of height $O(log(n))$ as well, which then can be expressed as a polynomial size formula by duplicating subcircuits. □

Using techniques similar to that of Theorem 5 - but considering solely partial cycle covers consisting of paths - one can prove the following theorem as well:

Theorem 6. *The hamiltonian of a $n \times n$ matrix M of bounded treewidth k can be expressed as a formula of size $O(n^{O(1)})$.*

4 Perfect matchings of planar graphs

Definition 8. *A* perfect matching *of a graph $G = \langle V, E\rangle$ is a subset E' of E such that every vertex in V is incident to* exactly one *edge in E'. The* weight *of a perfect matching E' is the product of weights of all edges in E'. By $SPM(G)$ we denote the sum of weights of all perfect matchings of G.*

In 1967 Kasteleyn showed in [7] that $SPM(G)$ can be computed efficiently if G is planar. His observations was that for planar graphs $SPM(G)$ could be expressed as a Pfaffian.

Theorem 7. *Let (f_n) be a family of polynomials with coefficients in a field K. The three following properties are equivalent:*

(i) *(f_n) can be computed by a family of polynomial size weakly skew circuits.*
(ii) *(f_n) can be computed by a family of polynomial size skew circuits.*
(iii) *There exists a family (G_n) of polynomial size planar graphs with edges weighted by constants from K or variables of f_n such that $f_n = SPM(G_n)$.*

The equivalence of (i) and (ii) is established in [12] and [13]. In [12] the complexity class VDET is defined as the class of polynomial families computed by polynomial size (weakly) skew circuits, and it is shown that the determinant is VDET-complete. We have therefore shown that computing $SPM(G)$ for a planar graph G is equivalent to computing the determinant. Previously it was known that $SPM(G)$ could be reduced to computing Pfaffians [7]. The equivalence of (iii) with (i) and (ii) follows immediately from Theorem 8 and Theorem 9.

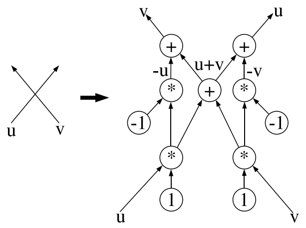

Fig. 4. Planar crossover widget for skew circuits

Theorem 8. *The output of every skew circuit of size n can be expressed as $SPM(G)$ where G is a weighted, planar, bipartite graph with $O(n^2)$ vertices. The weight of each edge of G is equal to 1, to -1, or to an input variable of the circuit.*

Proof. Let φ be a skew circuit; that is, for each multiplication gate *at least* one of the inputs is an input gate of φ (w.l.o.g. we assume it is *exactly* one). Furthermore, by making at most a linear amount of duplication we can assume all input gates have outdegree 1. Thus, every input gate of φ is either input to exactly one addition gate or input to exactly one multiplication gate.

Consider a drawing of φ in which all input gates which are input to an addition gate, are placed on a straight line, and all other gates are drawn on the same side of that line. Assume all arrows in the circuit are drawn as straight lines. This implies at most a quadratic number of places where two arrows cross each other.We replace crossings with the planar crossover widget from Figure 4.

For each multiplication gate we have that exactly one of the input gates is an input gate of φ, so these input gates can be placed arbitrarily close to the multiplication gate in which they are used. Thus we obtain a *planar* skew circuit φ' computing the same value as φ.

The overall idea is that every monomial in the polynomial represented by φ' will be encoded in our graph by a path-like subgraph from input s to output t. Each such subgraph has a perfect matching with weight equal to the monomial encoded on that path.

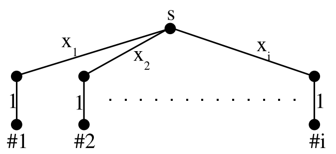

Fig. 5. Initialization for input gates which are input to addition gates

Consider a topological ordering of the gates in φ' in which input gates that are input to multiplication gates have numbers less than 1, and input gates that are input to addition gates have the numbers 1 through i (where i is the number of input gates that are input to addition gates). Let m be the number of the output gate in this topological ordering of φ'. Steps 1 through i in the construction of G are shown in Figure 5. Edge weight x_i denote the input at the gate with topological number i in φ'.

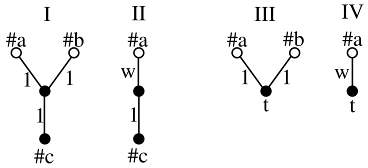

Fig. 6. I) Non-output add. II) Non-output mult. III) Output add. IV) Output mult.

For each step $i < m' \leq m$ an addition or multiplication gate is handled as shown in Figure 6. White vertices indicate vertices that are already present in the graph, whereas black vertices indicate new vertices that are introduced during that step. For simulating an addition gate we add 2 vertices and 3 edges each of weight 1. The edges are used to connect the 2 vertices that represent inputs to the addition gate. For simulating a multiplication gate we append a path of length 2 to an existing vertex. The edge weight w denote the value of the input gate of φ', which is input to that multiplication gate. Finally, the output gate of φ' is handled in a special way.

Correctness can be shown by induction using the following observation. For each step $1 \leq m' < m$ in the construction of G the following properties will hold for the graph generated so far: The labels $\sharp 1, \sharp 2, \ldots, \sharp m'$ have been assigned to m' distinct vertices. For all $1 \leq j \leq m'$ if the vertex with label $\sharp j$ is removed (along with all adjacent edges), then SPM of the remaining graph equals the value computed at gate with topological number j in φ'. $\square$

Theorem 9. *For any weighted, planar graph G with n vertices, $SPM(G)$ can be expressed as the output of a skew circuit of size $O(n^{O(1)})$. Inputs to the skew circuit are either constants or weights of the edges of G.*

Proof. Let H be a weighted graph and $\overrightarrow{H}$ an oriented version of H. Then the Pfaffian is defined as:

$$Pf(\overrightarrow{H}) = \sum_{\mathcal{M}} sgn(\mathcal{M}) \cdot w(\mathcal{M}),$$

where $\mathcal{M}$ ranges over all perfect matchings of $\overrightarrow{H}$. The Pfaffian depends on how the edges of $\overrightarrow{H}$ are oriented, since the sign of a perfect matching depends on this orientation (details on how sign depends on orientation are not needed here).

It is known from Kasteleyn's work [7] that all planar graphs have a *Pfaffian orientation* of the edges. A Pfaffian orientation is an orientation of the edges such that each term in the above sum has positive sign $sgn(\mathcal{M})$. So for planar graphs computing $SPM(G)$ reduces to computing Pfaffians.

A Pfaffian orientation of G does not depend on the weights of the edges, it only depends on the planar layout of G. In our reduction to a skew circuit we can therefore assume that a Pfaffian orientation $\overrightarrow{G}$ is given with G, so computing $SPM(G)$ by a skew circuit is reduced to computing $Pf(\overrightarrow{G})$ by a skew circuit.

From Theorem 12 in [9] we have that $Pf(\overrightarrow{G})$ can be expressed as $SW(G')$ where G' is a weighted, acyclic, directed graph with distinguished source and sink vertices denoted s and t (recall $SW(G')$ from Theorem 2). The size of G' is polynomial in the size of $\overrightarrow{G}$.

The last step is to reduce G' to a polynomial size skew circuit representing the same polynomial. Consider a topological ordering of the vertices of G'. The vertex s is replaced by an input gate with value 1. For a vertex v of indegree 1 in G', assume u is the vertex such that there is a directed edge from u to v in G', and assume the weight of this edge is w. We then replace v by a multiplication gate, where one arrow leading to this gate comes from the subcircuit representing u, and the other arrow leading to this gate comes from a new input gate with value w. Vertices of indegree $d > 1$ are replaced by a series of $d - 1$ addition gates, adding weights of all paths leading here. $\square$

References

1. A. Barvinok. Two algorithmic results for the traveling salesman problem. Mathematics of Operations Research, 21:65-84 (1996).
2. H. L. Bodlaender. NC-algorithms for graphs with small treewidth. In *Proc. 14th Workshop Graph-Theoretic Concepts in Computer Science WG'88,* p. 1-10. Springer Verlag, Lecture Notes in Computer Science 344 (1989).
3. H. L. Bodlaender and T. Hagerup. Parallel Algorithms with Optimal Speedup for Bounded Treewidth. In Z. Fulop and F. Gecseg, editors, *Proceedings 22nd International Colloquium on Automata, Languages and Programming,* pages 268-279, Berlin. Springer-Verlag, Lecture Notes in Computer Science 944 (1995).
4. P. Bürgisser. *Completeness and Reduction in Algebraic Complexity Theory.* Number 7 in Algorithms and Computation in Mathematics. Springer, 2000.
5. B. Courcelle. Graph Grammars, Monadic Second-Order Logic And The Theory Of Graph Minors. Contemporary Mathematics, Volume 147, p. 565-590 (1993).
6. B. Courcelle, J. A. Makowsky and U. Rotics. On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic. Discrete Applied Mathematics, 108:23-52 (2001).
7. P. W. Kasteleyn. Graph theory and crystal physics. In F. Harary, editor, Graph Theory and Theoretical Physics, p. 43-110. Academic Press (1967).
8. A. K. Mackworth and Y. Zhang. Parallel and Distributed Finite Constraint Satisfaction. Technical Report 92-30, Department of Computer Science, University of British Columbia, Vancouver, B. C. Canada (1992).
9. M. Mahajan, P. R. Subramanya and V. Vinay. The combinatorial approach yields an NC algorithm for computing Pfaffians. Discrete Applied Mathematics 143, p.1-16 (2004).
10. J. A. Makowsky and K. Meer. Polynomials of bounded treewidth. Foundations of Computational Mathematics, Proceedings of the Smalefest 2000, Felipe Cucker and J. Maurice Rojas, edts., World Scientific 2002, p. 211-250 (2002).
11. G. Malod. Polynômes et coefficients. Ph.D. thesis (2003).
12. G. Malod and N. Portier. Characterizing Valiant's Algebraic Complexity Classes. In MFCS 2006, Proceedings of the 31st International Symposium on Mathematical Foundations of Computer Science, volume 4162 of Lecture Notes in Computer Science, p. 704-716 Springer Verlag (2006).
13. S. Toda. Classes of arithmetic circuits capturing the complexity of computing the determinant. IEICE Transactions on Information and Systems, E75-D, p. 116-124 (1992).

14. L. G. Valiant. The complexity of computing the permanent. Theoretical Computer Science 8, p.181-201 (1979).
15. L. G. Valiant. Completeness classes in algebra. In *Proc. 11th ACM Symposium on Theory of Computing*, pages 249–261, 1979.
16. L. G. Valiant. Reducibility by algebraic projections. In *Logic and Algorithmic (an International Symposium held in honour of Ernst Specker)*, pages 365–380. Monographie n^o 30 de L'Enseignement Mathématique, 1982.
17. L. G. Valiant. Holographic algorithms. In *Proc. 45th Annual IEEE Symposium on Foundations of Computer Science,* IEEE Press, p. 306-315 (2004).