On the Expressive Power of Permanents and Perfect Matchings of Matrices of Bounded Pathwidth/Cliquewidth (Extended Abstract)

Uffe Flarup[1] and Laurent Lyaudet[2]

[1] Department of Mathematics and Computer Science
Syddansk Universitet, Campusvej 55, 5230 Odense M, Denmark
Fax: +45 65 93 26 91
flarup@imada.sdu.dk

[2] Laboratoire de l'Informatique du Parallélisme
École Normale Supérieure de Lyon, 46, allée d'Italie, 69364 Lyon Cedex 07, France
Fax: +33 4 72 72 80 80
laurent.lyaudet@ens-lyon.fr

Abstract. Some 25 years ago Valiant introduced an algebraic model of computation in order to study the complexity of evaluating families of polynomials. The theory was introduced along with the complexity classes VP and VNP which are analogues of the classical classes P and NP. Families of polynomials that are difficult to evaluate (that is, VNP-complete) includes the permanent and hamiltonian polynomials.

In a previous paper the authors together with P. Koiran studied the expressive power of permanent and hamiltonian polynomials of matrices of bounded treewidth, as well as the expressive power of perfect matchings of planar graphs. It was established that the permanent and hamiltonian polynomials of matrices of bounded treewidth are equivalent to arithmetic formulas. Also, the sum of weights of perfect matchings of planar graphs was shown to be equivalent to (weakly) skew circuits.

In this paper we continue the research in the direction described above, and study the expressive power of permanents, hamiltonians and perfect matchings of matrices that have bounded pathwidth or bounded cliquewidth. In particular, we prove that permanents, hamiltonians and perfect matchings of matrices that have bounded pathwidth express exactly arithmetic formulas. This is an improvement of our previous result for matrices of bounded treewidth. Also, for matrices of bounded weighted cliquewidth we show membership in VP for these polynomials.

1 Introduction

In this paper we continue the work that was started in [8]. Our focus is on easy special cases of otherwise difficult to evaluate polynomials, and their relation to various classes of arithmetic circuits. It is conjectured that the permanent and hamiltonian polynomials are hard to evaluate. Indeed, in Valiant's model

[17,18] these families of polynomials are both VNP-complete (VP and VNP are analogues of the classical classes P and NP). In the boolean framework they are complete for the complexity class ♯P [19]. However, for matrices of bounded treewidth the permanent and hamiltonian polynomials can efficiently be evaluated - the number of arithmetic operations being polynomial in the size of the matrix [4].

The sum of weights of perfect matchings in a weighted (undirected) graph is another hard to evaluate polynomial, but for planar graphs it can be evaluated efficiently due to Kasteleyn's theorem [11].

By means of reductions these evaluation methods can all be seen as general-purpose evaluation algorithms for certain classes of polynomials. As an example, if an arithmetic formula represents a polynomial P then one can construct a matrix A of bounded treewidth such that:

(i) The entries of A are variables of P, or constants from the underlying field.
(ii) The permanent of A is equal to P.

It turns out that the converse holds as well. Here we would like to study to what extent this can be done, when the matrix A has bounded pathwidth or bounded cliquewidth instead of bounded treewidth. In [8] the following results (with abuse of notation) were established:

(i) permanent/hamiltonian(bounded treewidth matrix) $\equiv$ arithmetic formulas.
(ii) perfect matchings(planar matrix) $\equiv$ arithmetic skew circuits.

One can also by similar proofs show that:

(iii) perfect matchings(bounded treewidth matrix) $\equiv$ arithmetic formulas.

In this paper we establish the following results:

(i) per/ham/perfect matchings(bounded pathwidth matrix) $\equiv$ arithmetic skew circuits of bounded width $\equiv$ arithmetic weakly skew circuits of bounded width $\equiv$ arithmetic formulas.
(ii) arithmetic formulas $\subseteq$ per/ham/perf. match.(bounded weighted cliquewidth matrix) $\subseteq$ VP.

Weighted cliquewidth is a natural extension of cliquewidth for weighted graphs defined in this paper. It differs from the cliquewidth of the underlying unweighted graph since this notion is not adapted to our problematic.

Overview of paper: The second section of the paper introduces definitions used throughout the paper and gives some small technical results related to graph-widths. Sections 3 and 4 are devoted to our main results, namely the expressiveness of the permanent, hamiltonian, and perfect matchings of graphs of bounded pathwidth and of bounded weighted cliquewidth respectively. We prove in Section 3 that the permanent, hamiltonian, and perfect matchings limited to bounded pathwidth graphs express arithmetic formulas. In Section 4, we show that for all three polynomials the complexity is between arithmetic formulas and VP for graphs of bounded weighted cliquewidth.

Due to space restriction, we had to remove several proofs that can be found in the full version [9]. We will not emphasize this point but all reductions can be done in polynomial time (most of them can be done in linear time).

2 Definitions and Preliminary Results

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*.*

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 the 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 [15].

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.*

2.2 Pathwidth and Treewidth

Since the definition of pathwidth is closely related to the definition of treewidth (bounded pathwidth is a special case of bounded treewidth) we also include the definition of treewidth in this paper. 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:*

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

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

A k-path-decomposition of G is then a k-tree-decomposition where the "tree" T is a path (each vertex $t \in V_T$ has at most one child in T).

The pathwidth (treewidth) of a directed, weighted graph is naturally defined as the pathwidth (treewidth) of the underlying, undirected, unweighted graph. The pathwidth (treewidth) of an $(n \times n)$ matrix $M = (m_{i,j})$ is defined as the pathwidth (treewidth) of the directed graph

2.3 Cliquewidth, NLCwidth and m-Cliquewidth

We have seen that the definition of pathwidth and treewidth for weighted graphs was straightforwardly defined as the width of the underlying, unweighted graph. This is a major difference compared to cliquewidth. We can see that if we consider non-edges as edges of weight 0 then every weighted graph has a clique (which has bounded cliquewidth 2) as its underlying, unweighted graph.

One motivation for studying bounded cliquewidth matrices was to obtain efficient algorithms for evaluating polynomials like the permanent and hamiltonian for such matrices. For this reason, it is not reasonable to define the cliquewidth of a weighted graph as the cliquewidth of the underlying, unweighted graph, because then computing the permanent of a matrix of cliquewidth 2 is as difficult as the general case. Hence, we put restrictions on how weights are assigned to edges: Edges added in the same operation between vertices having the same pair of labels will all have the same weight.

We now recall the definitions of cliquewidth, NLCwidth and m-cliquewidth for undirected, unweighted graph, and then introduce the new notions of W-cliquewidth, W-NLCwidth and W-m-cliquewidth which are variants of the preceding ones for *weighted*, directed graphs.

For the definitions of W-cliquewidth, W-NLCwidth and W-m-cliquewidth, we will consider simple, weighted, directed graphs where the weights are in some set W. Below, operations solely for the weighted case are indicated by **bold** font. In the three following constructions, a (directed) arc from a vertex x to a vertex y is only added by relevant operations if there is not already an arc from x to y.

Definition 4 ([3,5]). *A graph G has cliquewidth (W-**cliquewidth**) at most k iff there exists a set of source labels $\mathcal{S}$ of cardinality k such that G can be constructed using a finite number of the following operations (named clique operations **or W-clique operations**):*

- *(i) ver_a, $a \in \mathcal{S}$ (basic construct: create a single vertex with label a).*
- *(ii) $\rho_{a \to b}(H)$, $a, b \in \mathcal{S}$ (rename all vertices with label a to have label b instead).*
- *(iii) $\eta_{a,b}(H)$, $a, b \in \mathcal{S}$, $a \neq b$ (add edges between all pairs of vertices where one of them has label a and the other has label b).*
- ***(iii)'** $\alpha^{w}_{a,b}(H)$, $a, b \in \mathcal{S}$, $a \neq b$, $w \in W$ (add arcs of weight w from all vertices with label a to all vertices with label b).*
- *(iv) $H \oplus H'$ (disjoint union of graphs).*

Definition 5 ([20]). *A graph G has NLCwidth (W-**NLCwidth**) at most k iff there exists a set of source labels $\mathcal{S}$ of cardinality k such that G can be constructed using a finite number of the following operations (named NLC operations or W-**NLC operations**):*

(i) ver_a, $a \in \mathcal{S}$ *(basic construct: create a single vertex with label a).*

(ii) $\circ_R(H)$ *for any mapping R from* $\mathcal{S}$ *to* $\mathcal{S}$ *(for every source label* $a \in \mathcal{S}$ *rename all vertices with label a to have label* $R(a)$ *instead).*

(iii) $H \times_E H'$ *for any* $E \subseteq \mathcal{S}^2$ *(disjoint union of graphs to which are added edges between all couples of vertices* $x \in H$ *(with label* l_x*),* $y \in H'$ *(with label* l_y*) having* $(l_x, l_y) \in E$*).*

(iii)' $H \times_E H'$ *for any partial function* $E : \mathcal{S}^2 \times \{-1, 1\} \to W$ *(disjoint union of graphs to which arcs of weight w are added for each couple of vertices* $x \in H$*,* $y \in H'$ *whose labels* l_x, l_y *are such that* $E(l_x, l_y, s) = w$*; the arc is from x to y if* $s = 1$ *and from y to x if* $s = -1$*).*

One important distinction between cliquewidth, NLCwidth on one side and m-cliquewidth (to be defined below) on the other side is that in the first two each vertex is assigned exactly *one* label, and in the last one each vertex is assigned a *set* of labels (possibly empty).

Definition 6 ([6]). *A graph G has m-cliquewidth (W**-m-cliquewidth**) at most k iff there exists a set of source labels* $\mathcal{S}$ *of cardinality k such that G can be constructed using a finite number of the following operations (named m-clique operations or W**-m-clique operations**):*

(i) ver_A *(basic construct: create a single vertex with a set of labels A,* $A \subseteq \mathcal{S}$*).*

(ii) $H \otimes_{E,h,h'} H'$ *for any* $E \subseteq \mathcal{S}^2$ *and any* $h, h' : \mathcal{P}(\mathcal{S}) \to \mathcal{P}(\mathcal{S})$ *(disjoint union of graphs to which are added edges between all couples of vertices* $x \in H$*,* $y \in H'$ *whose sets of labels* L_x, L_y *contain a couple of labels* l_x, l_y *such that* $(l_x, l_y) \in E$*. Then the labels of vertices from H are changed via h and the labels of vertices from H' are changed via h').*

(ii)' $H \otimes_{E,h,h'} H'$ *for any partial function* $E : \mathcal{S}^2 \times \{-1, 1\} \to W$ *and any* $h, h' : \mathcal{P}(\mathcal{S}) \to \mathcal{P}(\mathcal{S})$ *(disjoint union of graphs to which arcs of weight w are added for each couple of vertices* $x \in H$*,* $y \in H'$ *whose sets of labels* L_x, L_y *contain* l_x, l_y *such that* $E(l_x, l_y, s) = w$*; the arc is from x to y if* $s = 1$ *and from y to x if* $s = -1$*. Then the labels of vertices from H are changed via h and the labels of vertices from H' are changed via h').*

In the last operation for W-m-cliquewidth, there is a possibility that two (or more) arcs are added from a vertex x to a vertex y during the same operation and then the obtained graph is not simple. For this reason, we will consider as well-formed terms only the terms where this does not occur.

It is stated in [6] (a proof sketch of some of this result is given in [6], and one of the inequalities is proven in [10]) that

$$mcwd(G) \leq wd_{NLC}(G) \leq cwd(G) \leq 2^{mcwd(G)+1} - 1.$$

Hence, cliquewidth, NLC-width and m-cliquewidth are equivalent with respect to boundedness.

The following theorem shows that the inequalities between the three widths are still valid in the weighted case. It justifies our definitions of weighted cliquewidth. A proof of this result can be obtained by collecting the ideas in [6,10] and combining them with our weighted definitions.

Theorem 1. *For any weighted graph G,*

$$Wmcwd(G) \leq Wwd_{NLC}(G) \leq Wcwd(G) \leq 2^{Wmcwd(G)+1} - 1.$$

The three preceding constructions of graphs can be extended to weighted graphs with loops by adding the basic constructs $verloop_a^w$ or $verloop_A^w$ which creates a single vertex with a loop of weight w and label a or set of labels A. One can then easily show the following result.

Let G be a weighted graph (directed or not) with loops. Let $Unloop(G)$ denote the weighted graph (directed or not) obtained from G by removing all loops. Then:

- $Wcwd(G) = Wcwd(Unloop(G))$.
- $Wwd_{NLC}(G) = Wwd_{NLC}(Unloop(G))$.
- $Wmcwd(G) = Wmcwd(Unloop(G))$.

2.4 Permanent and Hamiltonian Polynomials

In this paper we take a graph theoretic approach to deal with permanent and hamiltonian polynomials. The reason for this is that a natural way to define pathwidth, treewidth or cliquewidth of a matrix M, is by the width of the graph with adjacency matrix M, also see e.g. [13].

Definition 7. *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 8. *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 permanent of M is usually defined by the formula $\text{per}(M) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} m_{i,\sigma(i)}$ but here we emphasize on the graph theoretic approach. The equivalence with Definition 8 is clear since any permutation can be written down as a product of disjoint cycles, and this decomposition is unique. The *hamiltonian* polynomial $\text{ham}(M)$ is defined similarly, except that we only sum over cycle covers consisting of a *single* cycle (hence the name).

There is a natural way of representing polynomials by permanents. Indeed, if the entries of M are variables or constants from some field K, $f = \text{per}(M)$ is a polynomial with coefficients in K (in Valiant's terminology, f is a projection of the permanent polynomial). In the next section we study the power of this representation in the case where M has bounded pathwidth or bounded cliquewidth.

2.5 Connections between Permanents and Sum of Weights of Perfect Matchings

Another combinatorial characterization of the permanent is by sum of weights of perfect matchings in a bipartite graph. We will use this connection to deduce results for the permanent from results for the sum of weights of perfect matchings, and reciprocally.

Definition 9. *Let G be a directed graph (weighted or not). We define the* inside-outside graph *of G, denoted $IO(G)$, as the bipartite, undirected graph (weighted or not) obtained as follows:*

– *we split each vertex $u \in V(G)$ in two vertices u^+ and u^-;*
– *each arc uv (of weight w) is replaced by an edge between u^+ and v^- (of weight w). A loop on u (of weight w) is replaced by an edge between u^+ and u^- (of weight w).*

It is well-known that the permanent of a matrix M can be defined as the sum of weights of all perfect matchings of $IO(G_M)$. We can see that the adjacency matrix of $IO(G_M)$ is $\begin{pmatrix} 0 & M \\ M^t & 0 \end{pmatrix}$. The two following lemmas can be easily proved.

Lemma 1. *If G has treewidth (pathwidth) k, then $IO(G)$ has treewidth (pathwidth) at most $2k+1$.*

Lemma 2. *If G has W-cliquewidth k, then $IO(G)$ has W-cliquewidth at most $2k$.*

3 Expressiveness of Matrices of Bounded Pathwidth

In this section we study the expressive power of permanents, hamiltonians and perfect matchings of matrices of bounded pathwidth. We will prove that in each case we capture exactly the families of polynomials computed by polynomial size skew circuits of bounded width. A by-product of these proofs will be a proof of the equivalence between polynomial size skew circuits of bounded width and polynomial size *weakly* skew circuits of bounded width. This equivalence can not be immediately deduced from the already known equivalence between polynomial size skew circuits and polynomial size weakly skew circuits in the unbounded width case [16] (the proofs in [16] use a combinatorial characterization of the complexity of the determinant as the sum of weights of s,t-paths in a graph of polynomial size with distinguished vertices s and t. The additional difficulties to extend these proofs to circuits and graphs of bounded width would be equivalent to the ones we deal with). We will then prove that skew circuits of bounded width are equivalent to arithmetic formulas.

Definition 10. *An arithmetic circuit φ has* width $k \geq 1$ *if there exists a finite set of totally ordered layers such that:*

- *Each gate of φ is contained in exactly 1 layer.*
- *Each layer contains at most k gates.*
- *For every non-input gate of φ if that gate is in some layer n, then both inputs to it are in layer $n+1$.*

Theorem 2. *The polynomial computed by a weakly skew circuit of bounded width can be expressed as the permanent of a matrix of bounded pathwidth. The size of the matrix is polynomial in the size of the circuit. All entries in the matrix are either 0, 1, constants of the polynomial, or variables of the polynomial.*

Proof. Let φ be a weakly skew circuit of bounded width $k \geq 1$ and $l > 1$ the number of layers in φ. The graph G we construct will have pathwidth at most $4 \cdot k - 1$ (each bag in the path-decomposition will contain at most $4 \cdot k$ vertices) and the number of bags in the path-decomposition will be $l - 1$. G will have two distinguished vertices s and t, and the sum of weights of all directed paths from s to t equals the value computed by φ.

Since φ is a weakly skew circuit we consider a decomposition of it into disjoint subcircuits defined recursively as follows: The output gate of φ belongs to the *main subcircuit.* If a gate in the main subcircuit is an addition gate, then both of its input gates are in the main subcircuit as well. If a gate g in the main subcircuit is a multiplication gate, then we know that at least one input to g is the output gate of a subcircuit which is disjoint from φ except for its connection to g. This subcircuit forms a *disjoint multiplication-input subcircuit.* The other input to g belongs to the main subcircuit. If some disjoint multiplication-input subcircuit φ' contains at least one multiplication gate, then we make a decomposition of φ' recursively. Note that such a decomposition of a weakly skew circuit is not necessarily unique, because *both* inputs to a multiplication gate can be disjoint from the rest of the circuit, and any one of these two can be chosen as the one that belongs to the main subcircuit.

Let $\varphi_0, \varphi_1, \ldots, \varphi_d$ be the disjoint subcircuits obtained in the decomposition (φ_0 is the main subcircuit). The graph G will have a vertex v_g for every gate g of φ and $d + 1$ additional vertices $s = s_0, s_1, \ldots, s_d$ (t will correspond to v_g where g is the output gate of φ). For every gate g in the subcircuit φ_i, the following construction will ensure that the sum of weights of directed paths from s_i to v_g is equal to the value computed at g in φ.

For the construction of G we process the *decomposition* of φ in a bottom-up manner. Let subcircuit φ_i be a leaf in the decomposition of φ (so φ_i consists solely of addition gates and input gates). Assume that φ_i is located in layers top_i through bot_i ($1 \geq top_i \geq bot_i \geq l$) of φ. First we add a vertex s_i to G in bag $bot_i - 1$, and for each input gate with value w in the bottom layer bot_i of φ_i we add a vertex to G also in bag $bot_i - 1$ along with an edge of weight w from s_i to that vertex. Let n range from $bot_i - 1$ to top_i: Add the already created vertex s_i to bag $n - 1$ and handle input gates of φ_i in layer n as previously described. For each addition gate of φ_i in layer n we add a new vertex to G (which is added to bags n and $n - 1$ of the path-decomposition of G). In bag n we already have two vertices that represent inputs to this addition gate, so we add edges of weight 1 from both of these to the newly added vertex. The vertex representing the output gate of the circuit φ_i is denoted by t_i. The sum of weighted directed paths from s_i to t_i equals the value computed by the subcircuit φ_i.

Let φ_i be a subcircuit in the decomposition of φ that contains multiplication gates. Addition gates and input gates in φ_i are handled as before. Let g be a multiplication gate in φ_i in layer n and φ_j the disjoint multiplication-input subcircuit that is one of the inputs to g. We know that vertices s_j and t_j already are in bag n, so we add an edge of weight 1 from the vertex representing the other input to g to the vertex s_j, and an edge of weight 1 from t_j to a newly created vertex v_g that represents gate g, and then v_g is added to bags n and $n-1$.

For every b $(1 \geq b \geq l-1)$ we need to show that only a constant number of vertices are added to bag b during the entire process. Every gate in layer b of φ is represented by a vertex, and these vertices may all be added to bag b. Every gates in layer $b+1$ are also represented by a vertex, and all of these may be added to bag b (because they are used as input here). In addition to this, a number of s_i vertices are also added to bag b. For each gate of subcircuit φ_j in layer b or $b+1$, we have the corresponding s_j vertex in bag b. In total up to $4 \cdot k$ vertices are added to bag b.

Note that in layer 1 of φ we just have the output gate. This gate is represented by the vertex t of G which is in bag 1 of the path-decomposition.

The sum of weights of all directed paths from s to t in G can by induction be shown to be equal to the value computed by φ. The final step in the reduction to the permanent polynomial is to add an edge of weight 1 from t back to s and loops of weight 1 at all nodes different from s and t. ◻

With a longer proof one can show that the graph constructed in the preceding proof has pathwidth at most $\lfloor \frac{7 \cdot k}{2} \rfloor - 1$. Also, it can be modified to work for the hamiltonian polynomial as well, by adapting the idea in [14]: For the permanent polynomial each bag in the path-decomposition contains at most $4 \cdot k$ vertices; for each of these vertices we now need to introduce one extra vertex in the same bag. In addition each bag must contain 2 more vertices in order to establish a connection to adjacent bags in the path-decomposition. In total each bag now contains at most $8 \cdot k + 2$ vertices.

Theorem 3. *The polynomial computed by a weakly skew circuit of bounded width can be expressed as the sum of weights of perfect matchings of a symmetric matrix of bounded pathwidth. The size of the matrix is polynomial in the size of the circuit. All entries in the matrix are either 0, 1, constants of the polynomial, or variables of the polynomial.*

Proof. It is a direct consequence of Theorem 2 and Lemma 1. ◻

Theorem 4. *The hamiltonian of a matrix of bounded pathwidth can be expressed as a skew circuit of bounded width. The size of the circuit is polynomial in the size of the matrix.*

Proof. Let M be a matrix of bounded pathwidth k and let G_M be the underlying, directed graph. Each bag in the path-decomposition of G_M contains at most $k+1$ vertices. We refer to one end of the path-decomposition as the *leaf* of the path-decomposition and the other as the *root* (recall that path-decompositions are special cases of tree-decompositions).

We process the path-decomposition of G_M from the leaf towards the root. The overall idea is the same as the proof of Theorem 5 in [8] - namely to consider weighted partial path covers (i.e. partial covers consisting solely of paths) of subgraphs of G_M that are induced by the path-decomposition of G_M. During the processing of the path-decomposition of G_M at every level distinct from the root, new partial path covers are constructed by taking one previously generated partial path cover and then add at most $(k+1)^2$ new edges, so all the multiplication gates we have in our circuit are skew. For any bag in the path-decomposition of G_M we only need to consider a number of partial path covers that depends solely on k, so the circuit we produce has bounded width. At the root we add sets of edges to partial path covers to form hamiltonian cycles. □

By a similar proof, one can show the following theorem.

Theorem 5. *The sum of weights of perfect matchings of a symmetric matrix of bounded pathwidth can be expressed as a skew circuit of bounded width. The size of the circuit is polynomial in the size of the matrix.*

Theorem 6. *The permanent of a matrix of bounded pathwidth can be expressed as a skew circuit of bounded width. The size of the circuit is polynomial in the size of the matrix.*

Proof. It is a direct consequence of Theorem 5 and Lemma 1. □

The following corollary can be deduced from Theorem 2 and Theorem 6.

Corollary 1. *A family of polynomials is computable by polynomial size skew circuits of bounded width if and only if it is computable by polynomial size weakly skew circuits of bounded width.*

We need the following theorem from [1] to prove the equivalence between polynomial size skew circuits of bounded width and polynomial size arithmetic formulas.

Theorem 7. *Any arithmetic formula can be computed by a linear bijection straight-line program of polynomial size that uses three registers.*

This result of Ben-Or and Cleve is a generalisation of the celebrated result of Barrington in boolean complexity proving the equivalence between NC^1 and bounded width branching programs.

Let $R_1, \ldots, R_m$ be a set of m registers, a linear bijection straight-line (LBS) program is a vector of m initial values given to the registers plus a sequence of instructions of the form

(i) $R_j \leftarrow R_j + (R_i \times c)$, or
(ii) $R_j \leftarrow R_j - (R_i \times c)$, or
(iii) $R_j \leftarrow R_j + (R_i \times x_u)$, or
(iv) $R_j \leftarrow R_j - (R_i \times x_u)$,

where $1 \leq i, j \leq m$, $i \neq j$, $1 \leq u \leq n$, c is a constant, and $x_1, \ldots, x_n$ are variables (n is the number of variables). We suppose without loss of generality that the value computed by the LBS program is the value in the first register after all instructions have been executed.

Theorem 8. *A family of polynomials is computed by polynomial size skew circuits of bounded width if and only if it is a family of polynomial size arithmetic formulas.*

4 Expressiveness of Matrices of Bounded Weighted Cliquewidth

In this section we study the expressive power of permanents, hamiltonians and perfect matchings of matrices that have bounded weighted cliquewidth. We first prove that every arithmetic formula can be expressed as the permanent, hamiltonian, or sum of perfect matchings of a matrix of bounded W-cliquewidth using the results for the bounded pathwidth matrices and the following lemma.

Lemma 3. *Let G be a weighted graph (directed or not) with weights in W. If G has pathwidth k, then G has W-cliquewidth at most $k+2$.*

Theorem 9. *Every arithmetic formula can be expressed as the permanent of a matrix of W-cliquewidth at most 25 and size polynomial in n, where n is the size of the formula. All entries in the matrix are either 0, 1, constants of the formula, or variables of the formula.*

Proof. Let φ be a formula of size n. Due to the proof of Theorem 8, we know that it can be computed by a skew circuit of width 6 and size $O(n^{O(1)})$. Hence it is equal to the permanent of a graph of size $O(n^{O(1)})$, pathwidth at most $4 \cdot 6 - 1 = 23$ by Theorem 2, and W-cliquewidth at most $23 + 2 = 25$ by Lemma 3. $\square$

Similarly, one obtains the two following results.

Theorem 10. *Every arithmetic formula can be expressed as the hamiltonian of a matrix of W-cliquewidth at most $(8 \cdot 6 + 2 - 1) + 2 = 51$ and size polynomial in n, where n is the size of the formula. All entries in the matrix are either 0, 1, or constants of the formula, or variables of the formula.*

Theorem 11. *Every arithmetic formula can be expressed as the sum of weights of perfect matchings of a symmetric matrix of W-cliquewidth at most $25 \cdot 2 = 50$ and size polynomial in n, where n is the size of the formula. All entries in the matrix are either 0, 1, constants of the formula, or variables of the formula.*

We can modify the constructions for bounded treewidth graphs expressing formulas in [8] to obtain results similar to the ones above. These modifications require more work than the preceding proofs but we obtain smaller constants since we obtain graphs of W-cliquewidth at most 13/34/26 (instead of 25/51/50) whose permanent/hamiltonian/sum of perfect matchings are equal to formulas.

Due to our restrictions on how weights are assigned in our definition of bounded W-cliquewidth it is not true that weighted graphs of bounded treewidth also have bounded W-cliquewidth. In fact, if one tries to follow the proofs in [5,2] that show that graphs of bounded treewidth have bounded cliquewidth, then one obtains that a weighted graph G of treewidth k has W-cliquewidth at most $3 \cdot (|W_G| + 1)^{k-1}$ or $3 \cdot (\Delta + 1)^{k-1}$. W_G denotes the set of weights on the edges of G and Δ is the maximum degree of G. Weighted trees still have bounded weighted cliquewidth (the bound is 3). But we can show that there exist families of weighted graphs with treewidth two and unbounded W-cliquewidth [12].

We now turn to the upper bound on the complexity of permanent, hamiltonian, and perfect matchings of graphs of bounded weighted cliquewidth. We show that in all three cases the complexity is at most the complexity of VP.

The decision version of the hamiltonian cycle problem has been shown to be polynomial time solvable in [7] for matrices of bounded cliquewidth. Here we extend these ideas in order to compute the hamiltonian polynomial efficiently (in VP) for bounded W-m-cliquewidth matrices.

Definition 11. *A* path cover *of a directed graph G is a subset of the edges of G, such that these edges form disjoint, directed, non-cyclic paths in G. We require that every vertex of G is in (exactly) one path. For technical reasons we allow "paths" of length 0, by having paths that start and end in the same vertex. Such constructions do* not *have the same interpretation as a loop. The* weight *of a path cover is the product of weights of all participating edges (in the special case where there are no participating edges the weight is defined to be 1).*

Theorem 12. *The hamiltonian of an $n \times n$ matrix of bounded W-m-cliquewidth can be expressed as a circuit of size $O(n^{O(1)})$ and thus is in* VP*.*

Proof. Let M be an $n \times n$ matrix of bounded W-m-cliquewidth. By G we denote the underlying, directed, weighted graph for M. The circuit is constructed based on the parse tree T for G (T is the parse tree of a term over the W-m-clique algebra which constructs the graph G with a bounded number of labels). By T_t we denote the subtree of T rooted at t for some node $t \in T$. By G_t we denote the subgraph of G constructed from the parse tree T_t.

The overall idea is to produce a circuit that computes the sum of weights of all hamiltonian cycles of G. To obtain this there will be non-output gates that compute weights of all path covers of all G_t graphs, and then we combine these subresults. Of course, the total number of path covers can grow exponentially with the size of G_t, so we will not "describe" path covers directly by the edges participating in the covers. Instead we describe a path cover of some G_t graph by the labels associated with the start- and end-vertices of the paths in the cover. Such a description do not uniquely describe a path cover, because two different path covers of the same graph can contain the same number of paths and all these paths can have the same labels associated. However, we do not need the weight of each individual path cover. If multiple path covers of some graph G_t share the same description, then we just compute the sum of weights of these path covers.

For a leaf in the parse tree T of G we construct a single gate of constant weight 1, representing a path cover consisting of a single "path" of length 0, starting and ending in a vertex with the given labels. Per definition this path cover has weight 1.

For an internal node $t \in T$ the grammar rule describes which edges to add and how to relabel vertices. We obtain new path covers by considering a path cover from the left child of t and a path cover from the right child of t: For each such pair of path covers consider all subsets of edges added at node t, and for every subset of edges check if the addition of these edges to the pair of path covers will result in a valid path cover. If it does, then add a gate that computes the weight of this path cover, by multiplying the weight of the left path cover, the weight of the right path cover and the total weight of the newly added edges. After all pairs of path covers have been processed, check if any of the resulting path covers have the same description - namely that the number of paths in some path covers are the same, and that these paths have the same labels for start- and end-vertices. If multiple path covers have the same description then add addition gates to the circuit and produce a single gate which computes the sum of weights of all these path covers.

For the root node r of T we combine path covers from the children of r to produce hamiltonian cycles, instead of path covers. Finally, the output of the circuit is a summation of all gates computing weights of hamiltonian cycles.

Proof of correctness: The first step of the proof is by induction over the height of the parse tree T. We will show that for each non-root node t of T there is for every path cover description of G_t a corresponding gate in the circuit that computes the sum of weights of all path covers of G_t with that description. For the base cases - leaves of T - it is trivially true.

For the inductive step we consider two disjoint graphs that are being connected with edges at a node t of the parse tree T. Edges added at node t are *only* added in here, and not at any other nodes in T, so every path cover of G_t can be split into 3 parts: A path cover of G_{t_l}, a path cover of G_{t_r} and a polynomial number of edges added at node t. Consider a path cover description along with all path covers of G_t that have this description. All of these path covers can be split into 3 such parts, and by our induction hypothesis the weights of the path covers of G_{t_l} and G_{t_r} are computed in already constructed gates.

In order to complete the proof of correctness we have to handle the root t of T in a special way. At the root we do not compute weights of path covers, but instead compute weights of hamiltonian cycles. Every hamiltonian cycle of G can (similarly to path covers) be split into 3 parts: A path cover of G_{t_l}, a path cover of G_{t_r} and a polynomial number of edges added at the root of T. By our induction hypothesis all the needed weights are already computed.

The size of the circuit is polynomial since at each step the number of path cover descriptions is polynomially bounded once the W-m-cliquewidth is bounded. □

By a similar proof, one can prove the following theorem.

Theorem 13. *The sum of weights of perfect matchings of an $n \times n$ symmetric matrix of bounded W-NLCwidth can be expressed as a circuit of size $O(n^{O(1)})$ and thus is in* VP.

Theorem 14. *The permanent of an $n \times n$ matrix of bounded W-m-cliquewidth can be expressed as a circuit of size $O(n^{O(1)})$ and thus is in* VP.

Proof. It is a direct consequence of Theorem 13 and Lemma 2. □

References

1. Ben-Or, M., Cleve, R.: Computing Algebraic Formulas Using a Constant Number of Registers. In: STOC 1988, Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, pp. 254–257. ACM, New York (1988)
2. Corneil, D., Rotics, U.: On the Relationship Between Clique-Width and Treewidth. SIAM Journal on Computing 34, 825–847 (2005)
3. Courcelle, B., Engelfriet, J., Rozenberg, G.: Context-free Handle-rewriting Hypergraph Grammars. In: Graph-Grammars and Their Application to Computer Science, pp. 253–268 (1990)
4. Courcelle, B., Makowsky, J.A., Rotics, U.: On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic. Discrete Applied Mathematics 108, 23–52 (2001)
5. Courcelle, B., Olariu, S.: Upper bounds to the clique width of graphs. Discrete Applied Mathematics 101, 77–114 (2000)
6. Courcelle, B., Twigg, A.: Compact Forbidden-Set Routing. In: Thomas, W., Weil, P. (eds.) STACS 2007. LNCS, vol. 4393, pp. 37–48. Springer, Heidelberg (2007)
7. Espelage, W., Gurski, F., Wanke, E.: How to solve NP-hard graph problems on clique-width bounded graphs in polynomial time. In: Brandstädt, A., Le, V.B. (eds.) WG 2001. LNCS, vol. 2204. Springer, Heidelberg (2001)
8. Flarup, U., Koiran, P., Lyaudet, L.: On the expressive power of planar perfect matching and permanents of bounded treewidth matrices. In: Tokuyama, T. (ed.) ISAAC 2007. LNCS, vol. 4835, pp. 124–136. Springer, Heidelberg (2007)
9. Flarup, U., Lyaudet, L.: On the expressive power of permanents and perfect matchings of matrices of bounded pathwidth/cliquewidth. Research Report RR2008-05, ENS Lyon (2008), `http://aps.arxiv.org/pdf/0801.3408`
10. Johansson, O.: Clique-decomposition, NLC-decomposition, and modular decomposition - relationships and results for random graphs. Congressus Numerantium 132, 39–60 (1998)
11. Kasteleyn, P.W.: Graph theory and crystal physics. In: Harary, F. (ed.) Graph Theory and Theoretical Physics, pp. 43–110. Academic Press, London (1967)
12. Lyaudet, L., Todinca, I.: Private communication (2007)
13. Makowsky, J.A., Meer, K.: Polynomials of bounded treewidth. In: Cucker, F., Maurice Rojas, J. (eds.) Foundations of Computational Mathematics, Proceedings of the Smalefest 2000, vol. 2002, pp. 211–250. World Scientific, Singapore (2002)
14. Malod, G.: Polynômes et coefficients. Ph.D. thesis (2003)
15. Malod, G., Portier, N.: Characterizing Valiant's Algebraic Complexity Classes. In: Královič, R., Urzyczyn, P. (eds.) MFCS 2006. LNCS, vol. 4162, pp. 704–716. Springer, Heidelberg (2006)
16. Toda, S.: Classes of arithmetic circuits capturing the complexity of computing the determinant. IEICE Transactions on Information and Systems E75-D, 116–124 (1992)
17. Valiant, L.G.: Completeness classes in algebra. In: Proc. 11th ACM Symposium on Theory of Computing, pp. 249–261 (1979)
18. Valiant, L.G.: Reducibility by algebraic projections. In: Logic and Algorithmic (an International Symposium held in honour of Ernst Specker). Monographie n^o 30 de L'Enseignement Mathématique, pp. 365–380 (1982)
19. Valiant, L.G.: The complexity of computing the permanent. Theoretical Computer Science 8, 181–201 (1979)
20. Wanke, E.: k-NLC Graphs and Polynomial Algorithms. Discrete Applied Mathematics 54, 251–266 (1994)