The combinatorial approach yields an NC algorithm for computing Pfaffians*

Meena Mahajan†
The Institute of Mathematical Sciences, Chennai, INDIA.
meena@imsc.ernet.in

P. R. Subramanya, V. Vinay‡
Dept. of Computer Science & Automation,
Indian Institute of Science, Bangalore, INDIA.
{subbu,vinay}@csa.iisc.ernet.in

November 21, 2003

Keywords: Pfaffian, matching, combinatorics, algorithm, NC.

Abstract

The Pfaffian of an oriented graph is closely linked to Perfect Matching. It is also naturally related to the determinant of an appropriately defined matrix. This relation between Pfaffian and determinant is usually exploited to give a fast algorithm for computing Pfaffians.

We present the first NC algorithm for computing the Pfaffian. (Previous determinant-based methods computed it in NC only up to the correct sign, while previous polynomial-time algorithms did not lend themselves to parallelization.) Our algorithm is completely combinatorial in nature. Furthermore, it is division-free and works over arbitrary commutative rings. Over integers, we show that it can be implemented in the complexity class GapL. This upper bound was not known before, and establishes that computing the Pfaffian for integer skew-symmetric matrices is complete for GapL. Our proof techniques generalize the recent combinatorial characterization of determinant [MV97].

As a corollary, we show that under reasonable encodings of a planar graph, Kasteleyn's algorithm [Kas67] for counting the number of perfect matchings in a planar graph is also in GapL.

*An extended abstract describing these results appeared in the Proceedings of the Fifth Annual International Computing and Combinatorics Conference COCOON 1999, in the Springer-Verlag Lecture Notes in Computer Science series Volume 1627, pp. 134–143.

†Part of this work was done when this author was supported by the NSF grant CCR-9734918 on a visit to Rutgers University during summer 1999.

‡This work was initiated when this author was visiting DIMACS at Rutgers University during summer 1998.

1 Introduction

The main results of this paper are

1. a new combinatorial characterization of the Pfaffian of an oriented graph, and

2. the first NC algorithm for computing the Pfaffian, based on this characterization, and

3. pinpointing the computational complexity of Pfaffian as complete for the class GapL.

Our combinatorial characterisation and algorithm are similar in spirit to a recent result of Mahajan & Vinay [MV97] who give a combinatorial algorithm for computing the determinant of a matrix. See [Rot01] for a unified presentation of [MV97] and this work, along with various extensions.

Our algorithm pinpoints the computational complexity of Pfaffians: we establish that computing the Pfaffian of a graph is complete for the complexity class GapL. GapL is the class of functions that can be expressed as the difference in the number of accepting paths of two nondeterministic logspace machines (i.e. the difference of two $\sharp\mathsf{L}$ functions). This defines a space analog of the important counting classes GapP and $\sharp\mathsf{P}$. It is known that taking the difference of two $\sharp\mathsf{L}$ functions is equivalent to computing the determinant of an integer matrix [Vin91, Dam91, Val92, Tod91]. In other words, GapL is also the class of functions that are logspace reducible to computing the determinant of an integer matrix. GapL is contained in NC^2, the class of functions computable by polynomial size circuits of $\log^2 n$ depth.

Pfaffians are a well-studied object in algebra (see e.g. [Lan65]), and play an important role in matching theory [LP86]. They are intimately connected to determinants. In fact, Pfaffians can be considered to be more fundamental than determinants, in the sense that determinants are merely the bipartite special case of a general sum over matchings, see for instance [Knu96, Rot01]. In particular, for a matrix A of dimension n with $n \equiv 0$ or $1 \pmod 4$

$$\mathsf{det}(A) = \mathsf{Pf}\begin{pmatrix} 0 & A \\ -A^T & 0 \end{pmatrix}$$

implying that computing the Pfaffian is hard for GapL.

However, no matching upper bound was known till now. It is known that the square of the Pfaffian of an oriented graph is equal to the determinant of a related matrix. Thus, computing the Pfaffian correct up to the sign is known to be in NC. This is, however, not adequate to imply a GapL algorithm as (1) we do not know if GapL is closed under square roots (of positive integers), and (2) this does not give the correct sign of the Pfaffian. This does not even imply any NC algorithm for the sign. The Pfaffian (including the sign) can be computed in polynomial time [Bar83, GM94] using cross-eliminations (akin to Gaussian elimination for determinants) and choosing pivot elements carefully, see also [Von99]. However, this method suffers the same drawback as Gaussian elimination of not lending itself to efficient parallelization. In [GM94] the authors explicitly state that current techniques which allow determinant computations to be performed in NC do not appear to generalize to Pfaffians, and that no NC algorithm for Pfaffians is known. The subsequent determinant algorithm of [MV97] uses techniques that do generalize to Pfaffians, yielding the algorithm described in this work[1]. Our algorithm is thus the first to place Pfaffians inside NC, and more precisely, in GapL. It follows that computing integer Pfaffians exactly characterizes the class GapL.

The importance of computing the sign cannot be underestimated. As pointed out in [GM94], the computation of the Pfaffian can be substituted by that of the determinant (followed by a square-root operation) in the solution of existence versions of various problems, but to solve the corresponding exact value problem (see eg [CGM92]), computing the Pfaffian becomes essential. The algorithm of [MV00] is a case in point: it requires a linear combination of certain Pfaffians and thus cannot be implemented in NC unless the Pfaffian *with its sign* can be computed in NC.

One of the motivations for this work is to understand the complexity of Perfect Matching. Perfect Matching is not known to be in NC, but is known to be in RNC [MVV87]. This result has been recently improved by Allender, Reinhardt and Zhou [ARZ99] who show that Perfect Matching is in the class SPL (non-uniformly). (SPL is that subclass of GapL where functions take the value 0 or 1. Refer to [ARZ99] for details.) Interestingly, [ARZ99] make use of the Mahajan-Vinay clow sequences [MV97] critically to establish their result.

Pfaffians arise naturally in the study of matchings; the Pfaffian of an oriented graph is just the sum over all possible perfect matchings except that each matching has an associated sign as well, dictated by the orientation. This gives it a flavour similar to that of a determinant. In the absence of the sign, it would calculate the number of perfect matchings in a graph, a problem that is well-known to be complete for ♯P [Val79]. Also, in the case of special graphs, it is known that the graph may be oriented in such a way that all the terms of the Pfaffian turn out to be positive. This obviously means that there would be no cancellation and hence the Pfaffian would count the number of perfect matchings in the underlying graph. Such orientations of graphs are called Pfaffian orientations.

It is easy to construct graphs which *do not* admit a Pfaffian orientation; $K_{3,3}$ is one such graph. A celebrated result of Kasteleyn [Kas67] proves that all planar graphs admit a Pfaffian orientation. This result was subsequently improved by [Lit74] who showed that all $K_{3,3}$-free graphs admit a Pfaffian orientation. Finding such an orientation was shown to be in NC by Vazirani [Vaz89]. In this paper, we partially improve Vazirani's result to show that for planar graphs presented by reasonable encodings, a Pfaffian orientation can be found in deterministic logspace L. Combining this with our combinatorial algorithm for Pfaffians, we thus show that under reasonable encodings of planar graphs, the problem of counting the number of perfect matchings in a planar graph is in GapL as well. The problem of extending our result to $K_{3,3}$-free graphs remains to be investigated. Recent work in [MV00] uses our Pfaffian algorithm and the results of [GL99] to extend this result in another direction: under reasonable encodings, counting the number of perfect matchings in a graph of small (logarithmic) genus is in NC.

In section 2, a few definitions and preliminaries are stated. In section 3, we set up the combinatorial framework for Pfaffians. Section 4 focuses on the combinatorial algorithm for computing Pfaffians. We show in section 5 that finding Pfaffian orientations of planar graphs is in L, and hence counting the number of perfect matchings in a planar graph is in GapL.

[1]Our main theorem is similar in spirit to Theorem 1 in [MV97], though there are essential differences in the proof.

2 Preliminaries & Definitions

Let D be an $n \times n$ matrix. S_n is the permutation group on $\{1, 2, \ldots, n\}$ (denoted $[n]$). The permanent and determinant of D, $\mathsf{per}(D)$ and $\mathsf{det}(D)$, are defined as,

$$\mathsf{per}(D) = \sum_{\sigma \in S_n} \prod_i d_{i\sigma(i)} \qquad\qquad \mathsf{det}(D) = \sum_{\sigma \in S_n} sgn(\sigma) \prod_i d_{i\sigma(i)}$$

where $sgn(\sigma)$ is -1 if σ has an odd number of inversions, $+1$ otherwise. An equivalent definition of the sign of a permutation is in terms of the number of cycles in its cycle decomposition.

We associate with the matrix D the graph G_D, which is the complete directed graph on n vertices (with self-loops), having the matrix elements as edge weights. We denote by $wt(e)$ the weight of edge e. Every permutation $\sigma \in S_n$ can be decomposed into a set of cycles in G_D. These cycles are non-intersecting (i.e. simple), disjoint and they cover every vertex in the graph, i.e. these are **cycle covers**. The sign of a cycle cover is defined in terms of the number of even length cycles constituting it. The sign is $+1$ if there are an even number of such cycles, else it is -1.

A **clow** in G_D is a walk that starts at some vertex (called *head*), visits vertices larger than the head any number of times, and returns to the head. This cycle in G_D is not always a simple cycle. Formally,

Definition 1 [MV97]

1. *A clow is an ordered sequence of edges $C = \langle e_1, e_2, \ldots, e_m \rangle$ such that $e_i = \langle v_i, v_{i+1} \rangle$ and $e_m = \langle v_m, v_1 \rangle$, $v_1 \neq v_j$ for $j \in \{2, 3, \ldots, m\}$ and $v_1 = min\{v_1, \ldots, v_m\}$. The vertex v_1 is called the head of the clow and denoted $h(C)$. The length of the clow is $|C| = m$, and the weight of the clow is $\prod_{i=1}^{m} wt(e_i)$ and is denoted $wt(C)$. [Note: $C = \langle e \rangle$ where $e = \langle v, v \rangle$, i.e. a* self-loop*, is also a clow, of length one.]*

2. *A clow sequence is an ordered sequence of clows $C = (C_1, \ldots, C_k)$ such that $h(C_1) < \ldots < h(C_k)$ and $\sum_{i=1}^{k} |C_i| = n$.*

Kasteleyn [Kas67] introduced the use of Pfaffians to count the number of dimer coverings of a lattice graph. We define matchings and Pfaffians more formally. We consider only simple graphs (with no multiple edges) without self-loops.

Definition 2 *Given a simple loopless undirected graph $G = (V, E)$ with $V = \{1, 2, \ldots, n\}$:*

1. *A* **matching** *$\mathcal{M}$ is a subset of the edges of G such that no two edges have a vertex in common. That is, a matching is a set $\mathcal{M} \subseteq E(G)$ such that*

$$[(i_1, j_1), (i_2, j_2) \in \mathcal{M} \wedge (i_1 = i_2)] \Leftrightarrow j_1 = j_2$$

2. *A matching $\mathcal{M}$ is a* **perfect matching** *if every vertex $i \in V(G)$ occurs as the end-point of some edge in $\mathcal{M}$.*

3. *The* **weight** *of a matching $\mathcal{M}$, denoted $wt(\mathcal{M})$, is the product of the weights of its edges.*

Thus a perfect matching is a partition of the vertices of G into $\frac{n}{2}$ unordered pairs, where each pair is an edge. We will in the sequel prove our results for graphs with integer weights on edges, although our results can easily be seen to hold over arbitrary commutative rings.

Given an undirected graph G with no loops or multiple edges and with integer weights on the edges, assign orientations to edges of G to get a directed graph $\vec{G}$. The *Tutte Matrix* associated with $\vec{G}$ is the skew-symmetric adjacency matrix defined as

$$\begin{aligned} A_s(\vec{G})_{ij} &= \quad wt(i,j) \quad \text{if } \langle i,j \rangle \text{ is an edge in } \vec{G} \\ &= -wt(i,j) \quad \text{if } \langle j,i \rangle \text{ is an edge in } \vec{G} \\ &= \qquad 0 \quad \text{if } (i,j) \text{ is not an edge in } G \end{aligned}$$

Here $wt(i,j)$ refers to the weight of the undirected edge (i,j) in G.

Let D be a skew-symmetric matrix representing an orientation $\vec{G}$ of an undirected graph G. Let σ be a permutation in S_n. We can think of σ as representing the perfect matching $\{\langle\sigma(1),\sigma(2)\rangle, \langle\sigma(3),\sigma(4)\rangle, \ldots, \langle\sigma(n-1),\sigma(n)\rangle\}$. Several permutations can correspond to the same matching because the edges in the matching are neither oriented nor ordered (in fact, there are exactly $2^{\frac{n}{2}} \cdot (\frac{n}{2})!$ distinct permutations representing each matching). The weight of the permutation is defined as $wt(\sigma) = \prod_{i=1}^{\frac{n}{2}} d_{\sigma(2i-1)\sigma(2i)}$

Consider a perfect matching $\mathcal{M}$ in G. Irrespective of which permutation σ one chooses to represent $\mathcal{M}$, the term $sgn(\sigma)wt(\sigma)$ as computed with respect to $\vec{G}$ is invariant[2]. This invariant can be written as $p(\mathcal{M}) = sgn(\mathcal{M})wt(\mathcal{M})$. Thus orienting the graph G assigns a sign to each perfect matching. The Pfaffian of the oriented graph sums up these signed weights of perfect matchings.

Formally,

Definition 3 *Given a skew-symmetric matrix D, or equivalently, an orientation $\vec{G}$ of an undirected graph G over n vertices,*

- *The weight of a permutation $\sigma \in S_n$ with respect to D is defined as*

$$wt(\sigma) = \prod_{i=1}^{\frac{n}{2}} d_{\sigma(2i-1)\sigma(2i)}$$

- *The Pfaffian term $p(\mathcal{M})$ of a perfect matching $\mathcal{M}$ is defined to be $sgn(\sigma)wt(\sigma)$, where $\sigma \in S_n$ is any permutation satisfying $\mathcal{M} = \{\langle\sigma(1),\sigma(2)\rangle, \langle\sigma(3),\sigma(4)\rangle, \ldots, \langle\sigma(n-1),\sigma(n)\rangle\}$.*

[2] Why? Let σ and σ' both represent $\mathcal{M}$. If σ differs from σ' only in one edge being flipped, then the signs of the permutations are different but so are their weights (recall that D is skew-symmetric). If σ and σ' differ only in the arrangement of edges, then the number of transpositions to convert σ to σ' is even, and therefore their signs are the same. If more than one edge is flipped, and/or the arrangement of edges is different, the argument can be extended.

- *The Pfaffian of D (or equivalently, of $\vec{G}$) is defined as*

$$\mathsf{Pf}(D) = \sum_{\mathcal{M}} p(\mathcal{M})$$

where the sum ranges over all perfect matchings $\mathcal{M}$.

The canonical permutation for any matching $\mathcal{M}$, denoted $\sigma_{\mathcal{M}}$, is the permutation where edges are from smaller to larger vertices and are listed in increasing order of the smaller vertices in each edge, i.e. $\sigma_{\mathcal{M}}(2l-1) < \sigma_{\mathcal{M}}(2l)$ for $l = 1, \ldots, \frac{n}{2}$, and $\sigma_{\mathcal{M}}(1) < \sigma_{\mathcal{M}}(3) < \ldots < \sigma_{\mathcal{M}}(n-1)$. Using these, the Pfaffian may be defined as

$$\mathsf{Pf}(D) = \sum_{\mathcal{M}} sgn(\sigma_{\mathcal{M}}) \cdot wt(\sigma_{\mathcal{M}})$$

where the sum ranges over all perfect matchings $\mathcal{M}$.

Consider the graph given in Figure 1. The edge weights are represented by indeterminates x_a, x_b, x_c, x_d, x_e. If all edges are oriented from their smaller endpoint to their larger endpoint, then the associated matrix D is

$$D = \begin{bmatrix} 0 & x_a & x_b & x_e \\ -x_a & 0 & x_c & 0 \\ -x_b & -x_c & 0 & x_d \\ -x_e & 0 & -x_d & 0 \end{bmatrix}$$

Possible Matchings	**Terms**
(1 2) (3 4)	$+1.x_a.x_d$
(1 4) (2 3)	$+1.x_e.x_c$
(1 3) (2 4)	$-1.x_b.0$

$$\mathsf{Pf}(D) = x_a \cdot x_d + x_c \cdot x_e$$

Each Pfaffian term corresponds to a possible perfect matching in the graph. The non-vanishing terms correspond to feasible perfect matchings.

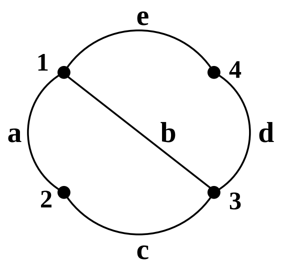

Figure 1: An Example Graph

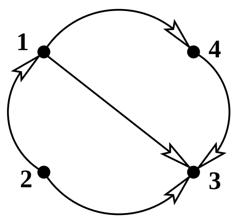

Figure 2: An Oriented Example Graph

Fig 2 imposes another orientation on the graph in Fig 1. Assuming that the weights of all the edges in the graph are equal to $+1$, this amounts to assigning $+1$ to x_b, x_c, x_e and -1 to x_a and x_d, and results in the matrix D given below. Now, comparing $\mathsf{per}(D)$, $\mathsf{det}(D)$ and $\mathsf{Pf}(D)$, we have

$$D = \begin{bmatrix} 0 & -1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ -1 & -1 & 0 & -1 \\ -1 & 0 & 1 & 0 \end{bmatrix} \qquad \begin{array}{c} \mathsf{per}(D) = 2 \\ \mathsf{det}(D) = 4 \\ \mathsf{Pf}(D) = 2 \end{array}$$

Linear algebra yields the following properties of skew symmetric matrices D.

- If D has an odd number of rows, then $\mathsf{det}(D) = 0$.
- If D has an even number of rows, then $\mathsf{det}(D) = (\mathsf{Pf}(D))^2$.

Definition 4 *The* **Gap-Path** *function is defined as follows:*

Input *: A directed acyclic graph G with integer weights on its edges; three special vertices s, t_+ and t_- of G.*

Output *: The function f defined below.*

$$f = \sum_{\rho:\ s \rightsquigarrow t_+} wt(\rho) \quad - \quad \sum_{\eta:\ s \rightsquigarrow t_-} wt(\eta)$$

where ρ iterates over all paths from s to t_+, η over all s to t_- paths, and $wt(\rho)$ or $wt(\eta)$ is the weight of the path (i.e. the product of the weights of all the edges on the path).

GapL is precisely the class of functions that can be formulated (using only logarithmic space) in this fashion. Equivalently, GapL consists of those functions that are the difference of two $\sharp\mathsf{L}$ functions, where $\sharp\mathsf{L}$ is the counting class for NL. As stated earlier, GapL is also, equivalently, the class of languages logspace reducible to computing the integer determinant. GapL is known to be contained in NC; thus problems in GapL, Gap-Path included, have parallel algorithms requiring $O(n^{c_1})$ processors and $O(\log^{c_2} n)$ parallel time, for some constants c_1 and c_2.

The Mahajan-Vinay GapL algorithm for the determinant [MV97] formulates the determinant in the first way described above. This GapL algorithm can be used to compute $\mathsf{det}(D)$, viz. $[\mathsf{Pf}(D)]^2$. However, this does not immediately yield a GapL algorithm even for the absolute value of the Pfaffian itself, because GapL is not known to be closed under square roots.

Let F_1 and F_2 be perfect matchings in a graph G. Their superposition, $F_1 \cup F_2$, is the graph obtained by including all closed walks along edges alternately from F_1 and F_2. Start at a vertex and walk along its matched edge in F_1. Next, walk along an adjacent edge in F_2. If this closes a cycle, pick an unvisited vertex and start the closed walks on the remaining vertices. Else, continue walking till there are no more matched edges. $F_1 \cup F_2$ is a cycle cover of G where each cycle is an alternating cycle and has even length. Note that each cycle in $F_1 \cup F_2$ can be routed in either of two possible directions. Generalizing Kasteleyn's notation for cycle covers on the regular 2-D lattice, we call the two possible routings clockwise and anti-clockwise. By clockwise routing we mean that routing where the first vertex is the smallest vertex in the cycle and the first edge of a cycle is picked from F_1.

Suppose we impose an orientation on the edges of G to get a directed graph $\vec{G}$. A cycle C in $F_1 \cup F_2$, when routed in any particular way, may traverse some edges according to their orientation in $\vec{G}$ and some edges in a direction opposite to their orientation. An edge e on cycle C is said to be *properly oriented* with respect to a routing of C if this routing of C traverses e according to its orientation in $\vec{G}$.

A routing of C is said to have an even orientation with respect to $\vec{G}$ if, in that routing of C, the number of properly oriented edges is even. Otherwise, the routing of C has an odd orientation. As every cycle in the superposition of two matchings is of even length, the orientation of both routings (clockwise or anti-clockwise) of such a cycle is the same. Hence for superposition cycles we refer to the orientation of the cycle itself rather than of a routing of the cycle. (For instance, for the graph of Figure 1 routed as in Figure 2, the cycle `aedc` has odd orientation because if we route the cycle edges in the order `e d c a`, then the three edges `e,d,a` are properly oriented. Similarly, if we route the cycle edges in the order `e a c d`, then only the edge `c` is properly oriented. The cycle `edb` is evenly oriented if routed as `edb` and oddly oriented if routed as `ebd`.)

3 A Combinatorial Setting for Pfaffians

In this section, we build the combinatorial framework for Pfaffians using a variant of *clow sequences*. This yields a combinatorial characterisation of Pfaffians, which we will utilize to develop a combinatorial algorithm for computing Pfaffians.

We first state some results which are essentially rephrasings of or easy derivations from standard material; see for instance [Lit74, Kas67]. For completeness, we sketch proofs as well.

The following is a variant of a standard lemma (See Lemma 8.3.1 from [LP86]) for the cases when the edges have arbitrary integer weights.

Lemma 5 *Let $\vec{G}$ be an arbitrary orientation of an undirected graph G. Let F_1 and F_2 be two* perfect matchings *of G. Let k be the number of evenly oriented alternating cycles in $F_1 \cup F_2$. Then,*

$$p(F_1) \cdot p(F_2) = (-1)^k \cdot wt(F_1) \cdot wt(F_2)$$

<u>Sketch of Proof:</u> $F_1 \cup F_2$ consists of cycles of even length. Consider the case when $F_1 \cup F_2$ consists of just one non-trivial cycle C. Choose the clockwise routing of C. Represent F_1 and F_2 by those permutations π and τ respectively, where the edges in C are listed in the order in which they appear in this clockwise routing, and the other edges are listed identically. Now it is clear that to go from the permutation π to the permutation τ, we need an odd number of transpositions. So the signs of these permutations are opposing.

(For instance, let $F_1 = (1,2)(3,4)(5,6)(7,8)$ and $F_2 = (1,6)(2,3)(4,5)(7,8)$. The cycle `123456` in $F_1 \cup F_2$ implies
$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{pmatrix}$ and $\tau = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 2 & 3 & 4 & 5 & 6 & 1 & 7 & 8 \end{pmatrix}$.
In order to convert π to τ, 1 has to be moved right over *five* vertices.)

As regards the weights, the edges for trivial cycles contribute the same to the weights of both permutations, and so the product is identical to the product of the weights of these edges in the matchings. A non-trivial cycle contributes the remaining weight with a -1 thrown in for each edge traversed wrongly. So, if the number of wrongly traversed edges is odd (i.e. the cycle is oddly oriented with respect to $\vec{G}$), then the net contribution is a -1 over and above the weights of the matchings. i.e. $wt(\pi) \cdot wt(\tau) = -wt(F_1) \cdot wt(F_2)$. This -1 will nullify the corresponding -1 in the product of the signs. On the other hand, if the number of edges traversed wrongly is even, then $wt(\pi) \cdot wt(\tau) = wt(F_1) \cdot wt(F_2)$, and the -1 in the sign is not nullified.

So, if the lone non-trivial cycle C in $F_1 \cup F_2$ is oddly oriented, then $p(F_1) \cdot p(F_2) = wt(F_1) \cdot wt(F_2)$. It is when C is evenly oriented that a -1 is introduced, i.e. $p(F_1) \cdot p(F_2) = -wt(F_1) \cdot wt(F_2)$.

Extending this argument, it is evident that if there are several non-trivial cycles in $F_1 \cup F_2$, then a -1 is introduced by each evenly oriented cycle. This proves the lemma. ■

The first idea used for computing the Pfaffian efficiently is to compute the signs of all Pfaffian terms with respect to a canonical orientation chosen as follows: Given a graph H and an orientation $\vec{H}$ whose Pfaffian is to be computed, let D be the skew-symmetric adjacency matrix $A_s(\vec{H})$. Given any skew-symmetric matrix A, there is a unique undirected graph G, with appropriate edge weights, such that orienting each edge of G in the forward direction (from its smaller endpoint to its larger endpoint) gives an oriented graph G^f with skew-symmetric adjacency matrix A. Consider the graph G and its orientation G^f obtained as above from the matrix D. Then $\mathsf{Pf}(\vec{H}) = \mathsf{Pf}(D) = \mathsf{Pf}(G^f)$, and we compute the signs of Pfaffian terms as in $\mathsf{Pf}(G^f)$ instead of directly in $\mathsf{Pf}(H)$.

The second idea is to compute all these signs with respect to some reference matching (possibly of zero weight). We choose the reference matching $\mathcal{I}$ corresponding to the identity permutation. And we use canonical permutations to represent each matching. Consider a matching $\mathcal{M}$ and its superposition with $\mathcal{I}$.

Using Lemma 5, we can show the following.

Corollary 6

$$sgn(\sigma_{\mathcal{M}}) = (-1)^k$$

where k is the number of cycles in $\mathcal{M} \cup \mathcal{I}$ that are evenly oriented with respect to G^f.

Proof: From Lemma 5, we have

$$\begin{aligned} sgn(\sigma_{\mathcal{M}}) \cdot wt(\sigma_{\mathcal{M}}) \cdot wt(\sigma_{\mathcal{I}}) &= [sgn(\sigma_{\mathcal{M}}) \cdot wt(\sigma_{\mathcal{M}})] \cdot [sgn(\sigma_{\mathcal{I}}) \cdot wt(\sigma_{\mathcal{I}})] \\ &= p(\mathcal{M}) \cdot p(\mathcal{I}) \\ &= (-1)^k \cdot wt(\sigma_{\mathcal{M}}) \cdot wt(\sigma_{\mathcal{I}}) \end{aligned}$$

Therefore, $sgn(\sigma_{\mathcal{M}}) = (-1)^k$. ■

The standard way to characterize the sign of a Pfaffian term is by the number of evenly oriented cycles. We show below that the number of cycles plus the number of properly oriented edges also characterizes the sign.

Lemma 7 *Let $\mathcal{M}$ be a partition of $[n]$ into $\frac{n}{2}$ unordered pairs and $\sigma_{\mathcal{M}}$ be its canonical permutation. Let $\mathcal{I}$ be the reference partition corresponding to the identity permutation. Let $\mathcal{M} \cup \mathcal{I}$ have l cycles, and let $\mathcal{C}$ denote the cycle cover obtained by the clockwise routing of each cycle in $\mathcal{M} \cup \mathcal{I}$. The sign of $\mathcal{M}$ in the Pfaffian is*

$$sgn(\sigma_{\mathcal{M}}) = (-1)^{|\{\langle i,j \rangle \;:\; \langle i,j \rangle \,\in\, \mathcal{C},\; i<j\}| \;+\; l}$$

Proof:

Let $\mathcal{C}$ have k even cycles and m odd cycles with respect to the forward orientation; $l = k + m$. Define $E = \{\langle i, j\rangle : \langle i, j\rangle \in \mathcal{C}, i < j\}$, the set of properly oriented edges. Let the contributions to $|E|$ from each of the even and odd oriented cycles be e_i and o_j for $1 \leq i \leq k$ and $1 \leq j \leq m$. Thus $|E| = \sum_{i=1}^{k} e_i + \sum_{j=1}^{m} o_j$. Note that each e_i is even and each o_j is odd. Thus $|E| + l = \sum_{i=1}^{k} e_i + \sum_{j=1}^{m} o_j + k + m = \sum_{i=1}^{k} e_i + \sum_{j=1}^{m}(o_j + 1) + k \equiv k \bmod 2$. Now the result follows from Corollary 6. ■

Corollary 8 *Let $\mathcal{M}$ be a partition as above and $\mathcal{I}$ be the identity permutation. The sign of $\mathcal{M}$ in the Pfaffian is*

$$sgn(\sigma_{\mathcal{M}}) = (-1)^{|M_2|+|M_3|+l}$$

where l is the number of cycles in $\mathcal{M} \cup \mathcal{I}$, $\mathcal{C}$ is the orientation of $\mathcal{M} \cup \mathcal{I}$ with each cycle routed in the clockwise sense and, M_2 and M_3 are disjoint subsets of $\mathcal{M}$ defined as,

$$\begin{aligned} M_2 &= \{\langle i, 2j\rangle \;:\; \langle i, 2j\rangle \;\in\; \mathcal{M} \cap \mathcal{C},\; i < 2j\;\} \\ M_3 &= \{\langle i, 2j-1\rangle \;:\; \langle i, 2j-1\rangle \;\in\; \mathcal{M} \cap \mathcal{C},\; i > 2j-1\;\} \end{aligned}$$

Proof: Lemma 7 tells us that we need to keep track of the parity of properly oriented (i.e. forward) edges in the clockwise routing of cycles in $\mathcal{M} \cup \mathcal{I}$. Instead here, let us focus on the edges of $\mathcal{M}$ alone, and hold an edge of $\mathcal{M}$ responsible for the following $\mathcal{I}$ edge. Each edge of $\mathcal{M}$ contributes 0, 1 or 2 forward edges to $\mathcal{M} \cup \mathcal{I}$. For instance, suppose the clockwise routing encounters edge $\langle i, 2j-1\rangle$ from $\mathcal{M}$, where $i < 2j-1$. Then it also encounters the edge $\langle 2j-1, 2j\rangle$ from $\mathcal{I}$, and both these edges are properly oriented. The other cases can be argued similarly.

The table below shows the partition of the edges of $\mathcal{M}$ into 4 sets. So, to evaluate the parity of forward edges, it suffices to keep track of the edges of M_2 and M_3, while M_1 and M_4 can be ignored.

Part	Edge of $\mathcal{M}$ as in $\mathcal{C}$	condition	number of forward edges in $\mathcal{C}$
M_1 forward edge ending in odd vertex	$\langle i, 2j-1\rangle$	$i < 2j-1$	2
M_2 reverse edge ending in odd vertex	$\langle i, 2j-1\rangle$	$i > 2j-1$	1
M_3 forward edge ending in even vertex	$\langle i, 2j\rangle$	$i < 2j$	1
M_4 reverse edge ending in even vertex	$\langle i, 2j\rangle$	$i > 2j$	0

■

We now introduce a new combinatorial object called **pclows** which we will use in our combinatorial setting for Pfaffians. (**Pclows** expand to Pfaffian Closed Walks and **p-edge** stands for Pfaffian-edge.)

Definition 9 • *A pair of edges $E = (e_1, e_2)$ is a* **p-edge** *if for some $i \in [1, n]$, either*

1. *$e_1 = \langle i, 2j\rangle$ for some j and $e_2 = \langle 2j, 2j-1\rangle$, or*

2. $e_1 = \langle i, 2j-1\rangle$ *for some* j *and* $e_2 = \langle 2j-1, 2j\rangle$.

Figures 3 & 4 below indicate the cases when a pair of consecutive edges forms a p-edge.

- *A* **pclow** *is a clow with its ordered sequence of edges being* $P = \langle E_1, E_2, \ldots, E_m\rangle$ *where each* E_i *is a p-edge. The length of the pclow, denoted* $|P|$*, is* $2m$*. A pclow traversal begins from its smallest vertex (called the* head*).*

- *A* **pclow sequence** *is an ordered sequence of pclows,* $\mathcal{P} = \langle P_1, \ldots, P_l\rangle$ *with heads in strictly increasing order, and with* $\sum_{i=1}^{l} |P_i| = n$*.*

- *The* **sign of a pclow sequence** *is the parity of the number of evenly oriented pclows with respect to* G^f*.*

- *The* **weight of a** p**-edge** $E = (e_1, e_2)$ *is the weight of the edge* e_1 *in its forward direction, i.e. if* $e_1 = (i, j)$*, its weight is* w_{ij} *if* $i < j$ *and* w_{ji} *otherwise. The second edge,* e_2*, always contributes a* **1** *to the weight of* E*. The* **weight of a pclow** *is the product of the p-edge weights. The* **weight of a pclow sequence** *is the product of the weights of its pclows.*

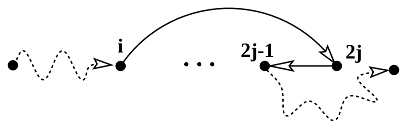

Figure 3: Selecting $\langle i, 2j\rangle$ from i.

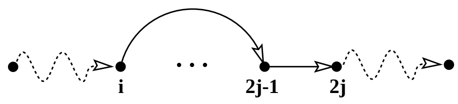

Figure 4: Selecting $\langle i, 2j-1\rangle$ from i.

Thus, if a pclow sequence $\mathcal{P}$ actually represents a perfect matching $\mathcal{M}$, then its weight is $wt(\sigma_{\mathcal{M}})$, and its sign is the sign of $\sigma_{\mathcal{M}}$. The results of Lemma 7 and Corollary 8 generalize to pclow sequences as well; generalising Lemma 7 gives us the following Lemma:

Lemma 10 *Let* $\mathcal{P}$ *be a pclow sequence with* l *pclows. Its sign is given by*

$$\mathsf{sign}(\mathcal{P}) = (-1)^{|\{\langle i,j\rangle \,:\, \langle i,j\rangle \,\in\, \mathcal{P},\ i<j\}| \;+\; l}$$

That is, the sign of a pclow sequence is the parity of the number of pclows in it plus the number of edges traversed in the forward *direction.*

Using pclow sequences, we prove a novel and powerful characterization of the Pfaffian. This provides the basis for our combinatorial algorithm for computing Pfaffians.

Theorem 11 *Let* D *be a skew-symmetric matrix. Its Pfaffian is given by:*

$$\mathsf{Pf}(D) = \sum_{\mathcal{W}:\ pclow\ sequence} sgn(\mathcal{W}) \cdot wt(\mathcal{W})$$

Proof: Pclow sequences that are cycle covers are the superposition of the reference matching with a perfect matching. We need to show that pclow sequences that are not cycle covers do not contribute to the summation. We establish an involution on the set of pclow sequences. Non-cycle covers get mapped onto non-cycle covers of opposite signs. The fixed points of the involution are the cycle covers.

Our technique would be to pair a pclow sequence with another having the same set of edges but with an opposite sign. Consequently, they cancel each other's contribution to the summation.

Note that all pclows are, by definition, even in length. However, a given sequence could have an odd length simple cycle in a pclow as shown in Fig 5 & 6. To pair such sequences, pick the pclow with the smallest head that has an odd simple sub-cycle. Walk down this pclow from its head, until you realize that you have gone around an odd cycle. Simply reverse the orientation of all the edges in this cycle. This defines a new pclow sequence. Conversely, starting with the new sequence, our mapping will consider the same (sub-)cycle and reversing its edges will give us the old pclow sequence; so they pair. Their total contribution is zero, since reversing an odd number of edges changes the parity of the number of properly oriented edges and so contributes a negative sign. The pclows in Figures 5 & 6 are an example of the above bijection.

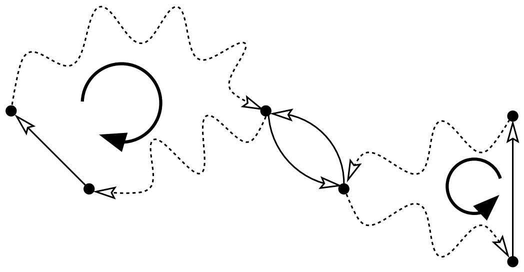

Figure 5: Pclow with odd sub-cycle

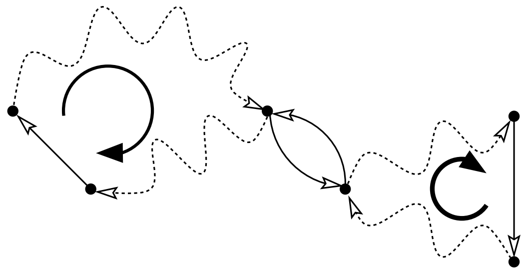

Figure 6: Pclow with odd sub-cycle reversed

We are left with pclow sequences in which all sub-cycles in all pclows are even. Let $\mathcal{P} = \langle P_1, \ldots, P_k \rangle$ be such a pclow sequence. Pick the smallest i such that P_{i+1} to P_k are disjoint simple cycles. If $i = 0$, then $\mathcal{P}$ is a cycle cover and $\mathcal{P}$ maps onto itself. Else, traverse P_i till one of the following happens,

1. We hit a vertex that meets one of P_{i+1} to P_k.

2. We hit a vertex that completes an even length simple cycle in P_i.

Let v be this vertex. Note that, these two conditions are mutually exclusive because of the way we have traversed P_i. We never hit a vertex that simultaneously satisfies both the conditions.

Case 1: Suppose v touches P_j, for some $j \in \{i+1, \ldots, k\}$. Let w be the partner of v in $\mathcal{I}$. (If v is odd, then $w = v + 1$, else $w = v - 1$.) Either the predecessor or the successor of v in P_j has to be w. But in P_i, w must be the successor of v; if w had been the predecessor of v in P_i, then we would have stopped our traversal at w itself.

The orientation of the (v, w) edge in P_j gives rise to two cases.

(a) *If the edge in P_j is from v to w:* (v,w) is identically oriented in P_i and P_j. We simply stick P_j into P_i at v. Formally, map $\mathcal{P}$ to a pclow sequence

$$\mathcal{P}' = \langle P_1, \ldots, P_{i-1}, P_i', P_{i+1}, \ldots, P_{j-1}, P_{j+1}, \ldots P_k \rangle$$

P_i' is obtained from P_i by inserting into it the simple cycle P_j at the first occurrence of v. Figure 7 illustrates this case.

(b) *If the edge in P_j is from w to v:* (v,w) has opposite orientations in P_i and P_j. We cannot stick P_j into P_i as is, because then P_i would lose the alternating property; it would use two edges from the reference matching consecutively. So first reverse the orientation of all edges in P_j, and then insert this pclow into P_i at the first occurrence of v. Figure 8 shows the mapping.

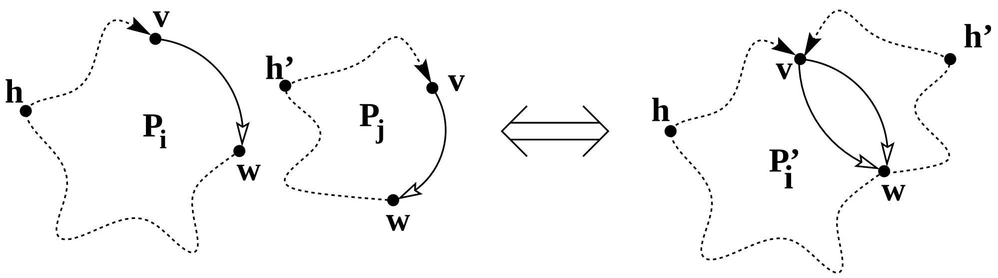

Figure 7: P_i and P_j have (v,w) oriented the same way.

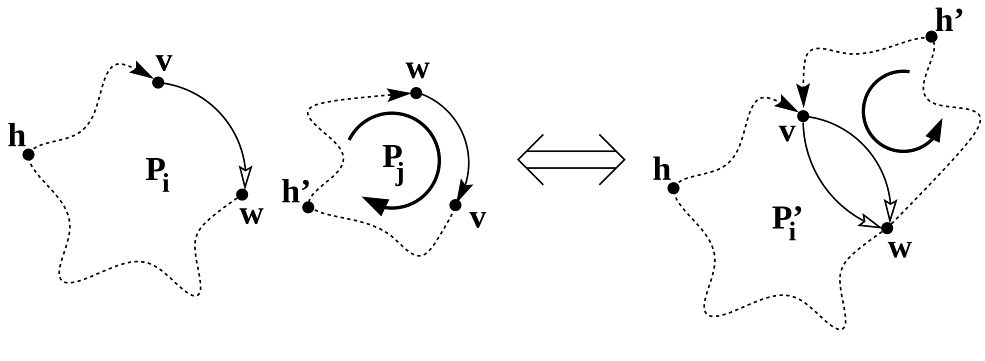

Figure 8: P_i and P_j have (v,w) oriented differently.

<u>**Case 2:**</u> Suppose v completes a simple cycle P in P_i. P must be disjoint from all the later cycles. Again, let w be the partner of v in $\mathcal{I}$. In P_i, w must be the successor of v; if w had been the predecessor of v in P_i, then a simple even cycle would have been detected at w before v and we would have stopped our traversal at w itself.

We modify the pclow sequence $\mathcal{P}$ by plucking out P from P_i and introducing it as a new pclow. P's position will be to the right of P_i as the head of P_i would be smaller than that of P. Let h' be the smallest vertex within P. There are now two sub-cases to consider:

(a) *In P_i, h' occurs at the start of a p-edge:* In this case, P forms a syntactically valid pclow. We just place P in the sequence to the right of P_i in the appropriate position dictated by P's head. Recall that in a valid pclow sequence, heads of pclows must be in strictly increasing order. This condition uniquely determinies the position of P. Formally, map $\mathcal{P}$ to a pclow sequence

$$\mathcal{P}' = \langle P_1, \ldots, P_{i-1}, P_i', P_{i+1}, \ldots, P_{j-1}, P, P_j, \ldots P_k \rangle$$

P_i' is obtained from P_i by deleting P from it, and the index j is such that $h(P_{j-1}) < h' = h(P) < h(P_j)$. This resulting sequence will satisfy Case 1(a) and applying the transformation there will give back $\mathcal{P}$.

(b) *In P_i, h' occurs in the middle of a p-edge:* In this case, P does not form a syntactically valid pclow. However, the simple cycle P' obtained by reversing all edges of P does, since now we can start a traversal from h' alternately using arbitrary edges and edges from $\mathcal{I}$. (Note that we have spliced and changed the p-edges of P.) Again, as in the preceding case, the position of P' in the plcow sequence is uniquely determined by h', and we map the pclow sequence to

$$\mathcal{P}' = \langle P_1, \ldots, P_{i-1}, P_i', P_{i+1}, \ldots, P_{j-1}, P', P_j, \ldots P_k \rangle$$

where P_i' and j are as before. This resulting sequence will satisfy Case 1(b) and applying the transformation there will give back $\mathcal{P}$.

We need to argue the correctness of these mappings. It should be clear that the new sequences map back to the original sequences and hence the mapping is an involution. We now show that the mapped pclow sequences have opposing signs, and as their weights are identical, they cancel each other's contribution.

Recall that the sign is characterized by the number of pclows and the number of properly oriented edges. In finding the mapped sequences, we change the parity of the number of pclows. The parity of the number of properly oriented edges remains unchanged, because the reversal of a pclow or an even length sub-cycle preserves this. Thus, the mapped pclow sequences indeed have opposing signs.

Pclow sequences arising from the superposition of the identity permutation with some perfect matching map onto themselves. These are the sole survivors. ■

4 A Combinatorial Algorithm for Pfaffians

In this section we describe a combinatorial algorithm for computing the Pfaffian. We construct a layered directed acyclic graph H_D with three special vertices s, t_+ and t_-, and show that the Pfaffian of D is precisely the Gap-Path function (recall definition 4) evaluated on this graph. In this model of computation, all $s \rightsquigarrow t_+$ ($s \rightsquigarrow t_-$) paths of positive (negative) sign in H_D are in 1-1 correspondence with pclow sequences of positive (negative) sign.

H_D has the vertex set, $\{s, t_+, t_-\} \cup \{[p, h, u, i] | p \in \{0, 1\}, h, u \in [1, n], i \in \{0, \ldots, n-1\} \}$. A path from s to $[p, h, u, i]$ indicates that in the pclow sequence being constructed along this path, p is the parity of the pclow sequence, h is the head of the current pclow, u is the current vertex on the pclow and i is the number of edges seen so far. An $s \rightsquigarrow t_+$ ($s \rightsquigarrow t_-$) path corresponds to a pclow sequence having a positive (negative) sign.

H_D has n layers and layer i has vertices of the form $[\_, \_, \_, i]$. The edges from layer (2j-1) to layer $2j$ are fixed and independent of D. The edges in H_D are:

1. $\langle s, [0, h, h, 0]\rangle$ for $h = 2i - 1$, where $i \in [1, \frac{n}{2}]$; edge weight is 1.
2. $\langle [p, h, u, 2i], [\bar{p}, h, v, 2i + 1]\rangle$, $v > h$, $v > u$, $i \in [0, \frac{n}{2} - 1]$; edge weight is d_{uv}.
3. $\langle [p, h, u, 2i], [p, h, v, 2i + 1]\rangle$, $v > h$, $v < u$, $i \in [0, \frac{n}{2} - 1]$; edge weight is d_{vu}.
4. $\langle [p, h, 2j - 1, 2i - 1], [\bar{p}, h, 2j, 2i]\rangle$ if $2j - 1 > h$, $2i < n$; edge weight is 1.
5. $\langle [p, h, 2j, 2i - 1], [p, h, 2j - 1, 2i]\rangle$ if $2j - 1 > h$, $2i < n$; edge weight is 1.
6. $\langle [p, h, h + 1, 2i - 1], [\bar{p}, h', h', 2i]\rangle$ if $h' > h$, h' is odd, $2i < n$; edge weight is 1.
7. $\langle [0, h, h + 1, n - 1], t_-\rangle$ and $\langle [1, h, h + 1, n - 1], t_+\rangle$ if $h = 2i - 1$, $i \in [1, \frac{n}{2}]$; edge weight is 1.

Theorem 12 *Given a skew symmetric matrix D, let H_D be the graph described above. Then,*

$$\mathsf{Pf}(D) = \sum_{\rho:\ s \rightsquigarrow t_+} wt(\rho) - \sum_{\eta:\ s \rightsquigarrow t_-} wt(\eta)$$

Proof: We show a one-to-one correspondence between $s \rightsquigarrow t_+$ ($s \rightsquigarrow t_-$) paths and pclow sequences of positive (negative) sign. Then, from Theorem 11 the result is immediate.

We utilize our characterization of the sign of a Pfaffian term as stated in Lemma 10.

Let $\mathcal{W} = \langle P_1, \ldots, P_k\rangle$ be a pclow sequence. Let h_i be the head of pclow P_i, n_i the number of forward edges in P_i, $p_i = (i + \sum_{j=1}^{i} n_j) \bmod 2$ the parity of the partial pclow sequence $\langle P_1, \ldots, P_i\rangle$, and m_i the total number of edges of the partial pclow sequence $\langle P_1, \ldots, P_i\rangle$. The path we construct for $\mathcal{W}$ goes through the vertices $[p_i, h_{i+1}, h_{i+1}, m_i]$. We use an inductive argument to prove our result.

Suppose, after traversing $\langle P_1, \ldots, P_i\rangle$, we are at the vertex $[p_i, h_{i+1}, h_{i+1}, m_i]$. In order to establish the inductive argument, it suffices to show that starting the traversal of P_{i+1} from this vertex, we will correctly reach $[p_{i+1}, h_{i+2}, h_{i+2}, m_{i+1}]$.

Let $P_{i+1} = \langle h_{i+1}, v_1, \ldots, v_l\rangle$. As P_{i+1} is a valid pclow, there is an edge from $[p_i, h_{i+1}, h_{i+1}, m_i]$ to $[p_i, h_{i+1}, v_1, m_i + 1]$ in H_D. As we traverse P_{i+1}, there will be vertices of the form $[p, h_{i+1}, v_j, m_i + j]$ where p is the parity of p_i and the number of forward edges upto v_j in P_{i+1}. When we reach the last vertex $v_l = h_{i+1} + 1$ of P_{i+1}, we would have changed signs as many as $n_{i+1} - 1$ times. The last edge of any pclow is always wrongly oriented and we reach $[p_{i+1}, h_{i+2}, h_{i+2}, m_{i+1}]$. Lemma 10 tells us that this is the proper way to calculate the sign of a pclow.

At layer n, depending on whether p_n is $+1$ or -1, H_D will have an edge to t_+ or t_-.

To show the other direction, consider a path $s \rightsquigarrow t_+$. If we were to list out the path, it will be a non-decreasing sequence with respect to the second component of each vertex.

Segments having the same second component correspond to a pclow whose head is the second component. The number of parity changes along this segment will exactly equal the number of forward edges along the path plus one. This generates a pclow sequence corresponding to the $s \rightsquigarrow t_+$ path and of even orientation parity. Similarly, each $s \rightsquigarrow t_-$ path corresponds to a pclow sequence of odd orientation parity. ■

Using simple dynamic programming techniques we can evaluate $\mathsf{Pf}(D)$ in polynomial time. The algorithm proceeds in n stages, where in the i^{th} stage we compute the sum of the weighted paths from s to any vertex x in layer i. Layer n has vertices t_+ and t_-, and we compute the difference of the weighted paths from s to t_+ and t_-. This algorithm looks at an edge in H_D once and hence is a polynomial-time algorithm ($O(n^4)$ ring operations).

There are NC algorithms for evaluating the Gap-Path function using standard divide-and-conquer techniques. Hence, to evaluate $\mathsf{Pf}(D)$, a parallel algorithm would be to construct a description of H_D as described above, and then use a parallel algorithm for Gap-Path. When D has integer entries, the entire parallel algorithm is in GapL. Thus we have:

Theorem 13 *Computing the Pfaffian of a skew-symmetric matrix over integers is in* GapL.

Corollary 14 *Computing the Pfaffian of a skew-symmetric matrix over integers is complete for* GapL *under uniform projections, even when the matrix is restricted to have entries from* $\{-1, 0, +1\}$.

When the entries of D are from an arbitrary commutative ring, we can still use the same algorithms, but we assume that ring operations are unit cost. With this model, the algorithm for computing Pfaffian has complexity as described below.

Theorem 15 *The Pfaffian of a skew-symmetric $n \times n$ matrix over any commutative ring can be computed by an arithmetic circuit with $O(n^4)$ gates and depth $O(\log n)$. The gates of the circuit are of two types: (1) unbounded fanin gates computing ring addition, and (2) bounded fanin gates computing ring multiplication. Alternatively, the Pfaffian can be computed by an OROW PRAM performing $O(n^6)$ work and running in $O(\log^2 n)$ parallel time, assuming unit cost per ring operation.*

Complete proofs of the above theorems is omitted here because they are identical to analogous results about determinants appearing in [MV97]. For more details about the efficient parallel implementations and the GapL implementation, see Sections 6.1 and 6.2 in [MV97], where similar algorithms for counting all clow sequences (the definition of sign and weight is slightly different) are described; see also [Rot01] for a different NC algorithm based on the same combinatorial characterisation of Theorem 11.

5 Finding Pfaffian orientations for Planar graphs

Counting the number of perfect matchings in a graph via Pfaffians requires:

1. Finding a Pfaffian orientation of the graph. An orientation of a graph is said to be Pfaffian if it gives the same sign to all perfect matchings.

2. Computing the Pfaffian of the associated matrix, assuming all edge weights in the undirected graph are $+1$.

We know how to do the latter from the previous section. We also know that the general problem of counting the number of perfect matchings in a graph is $\sharp$P-Complete. In this section, we show that by restricting ourselves to planar graphs, we can find Pfaffian orientations in GapL. This means that counting perfect matchings in planar graphs is in GapL.

We follow notation from [Kas67]. A superposition cycle is an even length cycle C which lies in the superposition of any two perfect matchings i.e. an even length simple cycle for which $G - V(C)$ has a perfect matching. Let G be an undirected graph, and consider an orientation of it yielding an oriented graph $\vec{G}$. The orientation is said to be *admissible* if every superposition cycle has an odd orientation in $\vec{G}$.

Lemma 16 [Kas67] *Admissible orientations assign the same sign to all perfect matchings, i.e. they are Pfaffian.*

Admissible orientations ensure that each Pfaffian term is positive, and hence the Pfaffian correctly computes the number of perfect matchings in the graph.

We show that an admissible orientation of a planar graph can be found in GapL by using a variant of Kasteleyn's algorithm.

Lemma 17 *Given a planar graph G along with*

1. *a planar combinatorial embedding, and*
2. *an ordering of the faces such that each face has an edge not shared with any of the earlier faces,*

finding an admissible orientation of G is logspace reducible to the problem of evaluating a **parity tree**.

__**Proof:**__ We first describe, without proof, Kasteleyn's algorithm [Kas67] to uncover an admissible orientation. Assume that the planar graph is so encoded that the faces seen so far form a simple connected component. Start with any face and do the following,

1. Orient all unoriented edges except one arbitrarily.
2. For the last edge, pick an orientation so that the cycle bounding the face has odd orientation parity when traversed clockwise.
3. Continue if there are unoriented faces remaining. Pick an adjoining face such that this face together with the other oriented faces form a simply connected region. Go to step 1.

Planar graphs have the property that no edge is common to more than two faces. We utilize this property in our *logspace* reduction.

Suppose we are given the faces of the input planar graph. We can order them in many ways as per the requirements of Kasteleyn's algorithm. Actually, the real requirement of

Kasteleyn's algorithm is not so much that the faces seen so far form a simple connected region, but that each face has at least one edge not shared with the earlier faces. One way of finding such an ordering of the faces, given any planar combinatorial embedding, is described in [LP86]; fix any face F (say the infinite face), and list faces in decreasing order of distance from F. Let us assume that any such ordering is fixed, say $C_1, C_2, \ldots, C_k$. With respect to this ordering and Kasteleyn's scheme, each face can be uniquely associated with an edge that has to have a specific orientation in order to maintain odd orientation parity. We denote such an edge as the **critical edge** for the face with respect to the face ordering. Figure 9 provides an illustration of critical edges associated with faces.

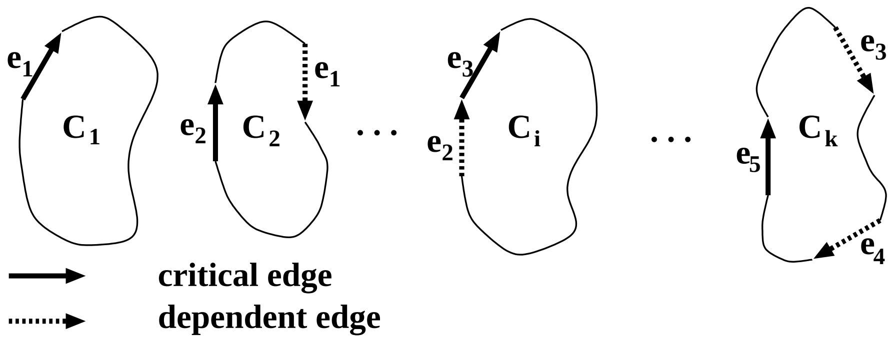

Figure 9: Critical Edges associated with each cycle

Consider a cycle C_i in Fig 9. There can be three types of edges on C_i that determine the orientation of its critical edge.

1. Non-critical edges whose orientations were fixed in cycles C_j, $j < i$.
2. Critical edges from the earlier cycles C_j, $j < i$.
3. Unoriented (or fresh) edges other than the critical edge of C_i.

We have reformulated the problem of finding an admissible orientation of a planar graph to one of finding the orientations of critical edges so that each face has an odd number of properly oriented edges. Let us pick a cycle and find the orientation for its critical edge. We must examine the orientations of all other edges of this cycle; they are of three types as described above.

- Oriented non-critical edges: As their orientations are fixed, we need know the parity of those among them that are properly oriented.
- Oriented critical edges: The orientations of these are fixed in earlier cycles. The difference being that we need to recompute them. Once computed, we take their parity.
- Fresh (unoriented) edges: We orient these clockwise, and hence we need to know the parity of such edges.

Computing the orientation of the critical edge of a cycle requires us to know the parity of the properly oriented edges in the cycle. During this computation, on encountering a critical edge of an earlier cycle, its orientation is the outcome of another parity computation on the edges constituting its cycle. This structure repeats along every critical edge to give us a computation graph. We shall show that this graph is actually a tree where every internal node is a parity node. Figure 10 illustrates this structure.

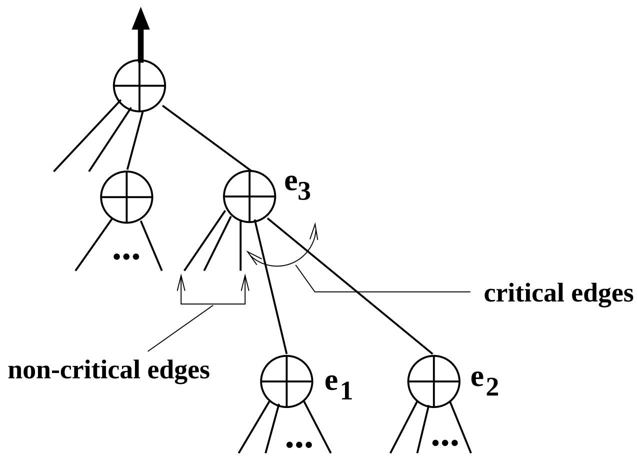

Figure 10: Parity Tree for finding the orientation of critical edges on cycles

Let e_j be the critical edge of an earlier face C_j appearing in face C_i (i.e. $j < i$). Finding e_j's orientation requires us to do a parity on the properly oriented edges in C_j. Consider some other critical edge e_l also appearing in C_i. The computation path from C_i along the parity node corresponding to e_l will never encounter e_j. This is because our input is a planar graph, and hence an edge is common to at most two faces. Therefore, all paths from a parity node are non-intersecting, and we have our parity tree.

We need to show that this is a *logspace* reduction. Note that we are assuming that the input is nicely encoded. Determining whether the current edge of a face is critical, non-critical or fresh can be easily done in *logspace* by scanning the preceding input. Therefore, given a parity node of the tree, we can identify the incoming arcs to it within *logspace*.

We deviate from Kasteleyn's scheme of identifying the critical edge on a cycle. We make the first unoriented edge on a cycle as the critical edge. By doing this, things are simpler because we now do not need to spend valuable computational resources to identify the last edge on the cycle. ■

We now show that parity tree evaluation is not a problem of high complexity; in fact,

Lemma 18 *Parity Tree Evaluation can be done in logspace.*

Proof: Parity is associative and commutative. Evaluating the parity of a sequence of elements requires one to remember only the parity of the elements seen so far. Hence, the parity tree can be collapsed. The parity of the leaves is therefore that of the tree. Systematically finding these leaves can be done by a logspace machine. ■

From Theorem 13 and lemmas 17 and 18, we have

Theorem 19 *Given a planar graph G along with*

1. *a planar combinatorial embedding, and*
2. *an ordering of the faces such that each face has an edge not shared with any of the earlier faces,*

counting the number of perfect matchings in G can be done within GapL.

6 Discussion

We have described the first NC algorithm for computing the Pfaffian with its sign. This algorithm is entirely combinatorial in nature. What is more, this algorithm is division-free. Is there any algebraic characterisation of the sign, perhaps using divisions, that may yield an algebraic NC algorithm for the problem? This and similar questions are also raised in [Rot01].

We have shown that given a "reasonable" encoding of a planar graph, counting the number of perfect matchings in it is in GapL. However, accepted versions of "reasonableness" differ. What would be more satisfying is to know the complexity of counting the number of perfect matchings in a graph, given that the graph is planar. While this is clearly still in NC, it is not immediately clear that this is still in GapL. Note that given a planar combinatorial embedding, extracting an encoding suitable for the algorithm of Section 5 can be done in nondeterministic logspace NL. Further, a recent paper [AM00] shows that a planar combinatorial embedding can be found, if one exists, in symmetric logspace SL. However, it is not known if the characteristic functions of NL languages are computable in GapL.

A related question that immedately arises is: what is the complexity of planarity testing itself? Can this be done in GapL? The best known upper bound so far (parallel deterministic algorithm) is that planarity testing can be done on a CRCW PRAM in $O(\log n)$ time [RR94], and hence is in AC^1. This algorithm also constructs a planar embedding, if one exists. The paper [AM00] further shows that planarity testing is sandwiched between the complexity classes deterministic logspace and symmetric logspace, and in the non-uniform setting where these classes coincide, is complete for that class. However its exact complexity in the uniform setting is still not known.

For most problems, the decision, search and counting versions are ordered that way in increasing difficulty. For perfect matchings, when we restrict our attention to planar graphs, the order is reversed: counting is easy (read NC), decision is easy but only because one can count, and search is not even known to be in NC. This is a baffling situation; clearly, one expects an NC algorithm for search. If the planar graph is further restricted to be bipartite, search is also known to be in NC [MN95]. This result was recently extended to small genus bipartite graphs [MV00]. However, we do not know of any approach to removing the bipartite condition.

Of course, the big question still remains open: what exactly is the complexity of both the decision and counting versions of perfect matchings in general graphs?

Acknowledgments

The authors would like to thank Günter Rote for very helpful comments on an earlier version of the paper. The presentation in the current version is heavily influenced by extensive discussions the first author had with Jaikumar Radhakrishnan and Kasturi Varadarajan. Thanks are also due to anonymous referees, whose remarks helped immensely in improving the presentation.

References

[AM00] Eric Allender and Meena Mahajan. The complexity of planarity testing. In *Proc. Symposium on Theoretical Aspects of Computing (STACS). LNCS vol. 1770.* Springer, 2000. To appear in *Information and Computation.*

[ARZ99] E. Allender, K. Rheinhardt, and S. Zhou. Isolation, matching and counting: uniform and nonuniform upper bounds. *Journal of Computer and System Sciences*, 59:164–181, 1999.

[Bar83] F Barahona. Balancing signed toroidal graphs in polynomial time. Preprint, University of Chile, 1983.

[CGM92] P M Camerini, G Galbiati, and F Maffioli. Random pseudo-polynomial algorithms for exact matroid problems. *Journal of Algorithms*, 13:258–273, 1992.

[Dam91] C. Damm. DET=$L^{(\#L)}$. Technical Report Informatik–Preprint 8, Fachbereich Informatik der Humboldt–Universität zu Berlin, 1991.

[GL99] A Galluccio and M Loebl. On the theory of Pfaffian orientations I: Perfect matchings and permanents. *The Electronic Journal of Combinatorics*, R6, 1999.

[GM94] G Galbiati and F Maffioli. On the computation of Pfaffians. *Discrete Applied Mathematics*, 51:269–275, 1994.

[Kas67] P W Kastelyn. Graph theory and crystal physics. In F Harary, editor, *Graph Theory and Theoretical Physics*, pages 43–110. Academic Press, 1967.

[Knu96] D E Knuth. Overlapping Pfaffians. *The Electronic Journal of Combinatorics*, 3(2):R5, 1996.

[Lan65] S Lang. *Algebra.* Addison-Wesley, Reading, MA, 1965.

[Lit74] C H C Little. An extension of Kasteleyn's method of enumerating the 1-factors of planar graphs. In *Combinatorial Mathematics, Proc.* 2^{nd} *Australian Conference*, pages 63–72. D.Holton, *Lecture Notes in Mathematics*, 403, 1974.

[LP86] L Lovasz and M Plummer. *Matching Theory.* North-Holland, 1986. Annals of Discrete Mathematics 29.

[MN95] G Miller and J Naor. Flow in planar graphs with multiple sources and sinks. *SIAM Journal on Computing*, 24:1002–1017, 1995.

[MV97] M. Mahajan and V Vinay. Determinant: combinatorics, algorithms, complexity. *Chicago Journal of Theoretical Computer Science* `http://www.cs.uchicago.edu/publications/cjtcs`, 1997:5, dec 1997. Preliminary version in Proceedings of the *Eighth Annual ACM-SIAM Symposium on Discrete Algorithms* SODA 1997, 730–738.

[MV00] M. Mahajan and K. Varadarajan. A new NC-algorithm for finding a perfect matching in planar and bounded genus graphs. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing (STOC)*, pages 351–357, 2000.

[MVV87] K Mulmuley, U Vazirani, and V Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–131, 1987.

[Rot01] Günter Rote. Division-free algorithms for the determinant and the pfaffian: Algebraic and combinatorial approaches. In H. Alt, editor, *Computational Discrete Mathematics: Advanced Lectures*, volume LNCS 2122, pages 119–135. Springer, 2001.

[RR94] V. Ramachandran and J. Reif. Planarity testing in parallel. *Journal of Computer and System Sciences*, 49:517–561, 1994.

[Tod91] S. Toda. Counting problems computationally equivalent to the determinant. Technical Report CSIM 91-07, Dept of Comp Sc & Information Mathematics, Univ of Electro-Communications, Chofu-shi, Tokyo, 1991.

[Val79] L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.

[Val92] L. G. Valiant. Why is boolean complexity theory difficult? In M. S. Paterson, editor, *Boolean Function Complexity*. Cambridge University Press, 1992. London Mathematical Society Lecture Notes Series 169.

[Vaz89] V Vazirani. NC algorithms for computing the number of perfect matchings in $K_{3,3}$-free graphs and related problems. *Information and Computation*, 80(2):152–164, 1989.

[Vin91] V Vinay. Counting auxiliary pushdown automata and semi-unbounded arithmetic circuits. In *Proceedings of 6th Structure in Complexity Theory Conference*, pages 270–284, 1991.

[Von99] Jan Vondrák. Implementation and testing of a new algorithm for the Max-Cut problem. Master's thesis, Charles University, Prague, 1999.