Determinant: Old Algorithms, New Insights (Extended Abstract)

Meena Mahajan[1] and V Vinay[2]

[1] Institute of Mathematical Sciences, Chennai 600 113, INDIA.
meena@imsc.ernet.in
[2] Department of Computer Science and Automation,
Indian Institute of Science, Bangalore 560 012, INDIA.
vinay@csa.iisc.ernet.in

Abstract. In this paper we approach the problem of computing the characteristic polynomial of a matrix from the combinatorial viewpoint. We present several combinatorial characterizations of the coefficients of the characteristic polynomial, in terms of walks and closed walks of different kinds in the underlying graph. We develop algorithms based on these characterizations, and show that they tally with well-known algorithms arrived at independently from considerations in linear algebra.

1 Introduction

Computing the determinant, or the characteristic polynomial, of a matrix is a problem which has been studied for several years from the numerical analysis viewpoint. In the mid 40's, a series of algorithms which employed sequential iterative methods to compute the polynomial were proposed, the most prominent being due to Samuelson [Sam42], Krylov, Leverier; see, for instance, the presentation in [FF63]. Then, in the 80's, a series of parallel algorithms for the determinant were proposed, due to Csanky, Chistov, Berkowitz [Csa76,Chi85,Ber84]. This culminated in the result, shown independently by several complexity theorists including Vinay, Damm, Toda, Valiant [Vin91,Dam91,Tod91,Val92], that computing the determinant of an integer matrix is complete for the complexity class GapL, and hence computationally equivalent in a precise complexity-theoretic sense to iterated matrix multiplication or matrix powering.

In an attempt to unravel the ideas that went into designing efficient parallel algorithms for the determinant, Valiant studied Samuelson's algorithm and interpreted the computation combinatorially [Val92]. He presented a combinatorial theorem concerning closed walks in graphs, the correctness of which followed from that of Samuelson's algorithm. This was the first attempt to view determinant computations as graph-theoretic rather than linear algebraic manipulations. Inspired by this, and by Straubing's [Str83] purely combinatorial and extremely elegant proof of the Cayley-Hamilton Theorem, Mahajan and Vinay [MV97] described a combinatorial algorithm for computing the characteristic polynomial. The proof of correctness of this algorithm is also purely combinatorial and does not rely on any linear algebra or polynomial arithmetic.

In this paper, we follow up on the work presented in [Val92,Str83,MV97] and present a unifying combinatorial framework in which to interpret and analyse a host of algorithms for computing the determinant and the characteristic polynomial. We first describe what the coefficients of the characteristic polynomial of a matrix M represent as combinatorial entities with respect to the graph G_M whose adjacency matrix is M. We then consider various algorithms for evaluating the coefficients, and in each case we relate the intermediate steps of the computation to manipulation of similar combinatorial entities, giving combinatorial proofs of correctness of these algorithms.

In particular, in the graph-theoretic setting, computing the determinant amounts to evaluating the signed weighted sum of cycle covers. This sum involves far too many terms to allow evaluation of each, and we show how the algorithms of [Sam42,Chi85,Csa76] essentially expand this sum to include more terms i.e., generalizations of cycle covers, which eventually cancel out but which allow easy evaluation. The algorithm in [MV97] uses clow sequences explicitly; Samuelson's method [Sam42] implicitly uses prefix clow sequences; Chistov's method [Chi85] implicitly uses tables of tour sequences; and Csanky's algorithm [Csa76] hinges around Leverier's Lemma (see, for instance, [FF63]), which can be interpreted using loops and partial cycle covers. In each of these cases, we explicitly demonstrate the underlying combinatorial structures, and give proofs of correctness which are entirely combinatorial in nature. In this paper, we omit the combinatiorial proof of Csanky's algorithm for want of space; the full paper carries the proof.

In a sense, this paper parallels the work done by a host of combinatorialists in proving the correctness of matrix identities using the graph-theoretic setting. Foata [Foa65] used tours and cycle covers in graphs to prove the MacMohan master theorem; Straubing [Str83] reproved the Cayley-Hamilton theorem using counting over walks and cycle covers; Garsia [GE], Orlin [Orl78] and Tempereley [Tem81] independently found combinatorial proofs of the matrix-tree theorem and Chaiken [Cha82] generalized the proof to the all-minor matrix-tree theorem; Foata [Foa80] and then Zeilberger [Zei85] gave new combinatorial proofs of the Jacobi identity; Gessel [Ges79] used transitive tournaments in graphs to prove Vandermonde's determinant identity. More recently, Minoux [Min97] showed an extension of the matrix-tree theorem to semirings, again using counting arguments over arborescences in graphs. For beautiful surveys of some of these results, see Zeilberger's paper [Zei85] and chapter 4 of Stanton and White's book on Constructive Combinatorics [SW86]. Zeilberger ends with a host of "exercises" in proving many more matrix identities combinatorially.

Thus, using combinatorial interpretations and arguments to prove matrix identities has been around for a while. To our knowledge, however, a similar application of combinatorial ideas to interpret, or prove correctness of, or even develop new *algorithms* computing matrix functions, has been attempted only twice before: by Valiant [Val92] in 1992, and by the present authors in our earlier paper in 1997 [MV97]. We build on our earlier work and pursue a new thread of ideas here.

This paper is thus a collection of new interpretations and proofs of known results.

2 Matrices, Determinants and Graphs

Let A be a square matrix of dimension n. For convenience, we state our results for matrices over integers, but they apply to matrices over any commutative ring.

We associate matrices of dimension n with complete directed graphs on n vertices, with weights on the edges. Let G_A denote the complete directed graph associated with the matrix A. If the vertices of G_A are numbered $\{1, 2, \ldots, n\}$, then the weight of the edge $\langle i, j\rangle$ is a_{ij}. We use the notation $[n]$ to denote the set $\{1, 2, \ldots, n\}$.

The determinant of the matrix A, $\det(A)$, is defined as the signed sum of all weighted permutations of S_n as follows:

$$det(A) = \sum_{\sigma \in S_n} sgn(\sigma) \prod_i a_{i\sigma(i)}$$

where $sgn(\sigma)$ is $+1$ if the number of inversions in σ is even and -1 otherwise. The number of inversions is the cardinality of the set $\{\langle i, j\rangle \mid i < j, \sigma(i) > \sigma(j)\}$.

Each $\sigma \in S_n$ has a cycle decomposition, and it corresponds to a set of cycles in G_A. Such cycles of G_A have an important property: they are all simple (non-intersecting), disjoint cycles; when put together, they touch each vertex exactly once. Such sets of cycles are called cycle covers. Note that cycle covers of G_A and permutations of S_n are in bijection with each other.

We define weights of cycle covers to correspond to weights of permutations. The weight of a cycle is the product of the weights of all edges in the cycle. The weight of a cycle cover is the product of the weights of all the cycles in it. Thus, viewing the cycle cover $\mathcal{C}$ as a set of edges, $w(\mathcal{C}) = \prod_{e \in \mathcal{C}} w(e)$. Since the weights of the edges are dictated by the matrix A, we can write $w(\mathcal{C}) = \prod_{\langle i,j\rangle \in \mathcal{C}} a_{ij}$.

We can also define the sign of a cycle cover to be consistent with the sign of the corresponding permutation. A cycle cover is even (resp. odd) if it contains an even number (resp. odd) of even length cycles. Equivalently, the cycle cover is even (resp. odd) if the number of cycles plus the number of edges is even (resp. odd). Define sign of a cycle cover $\mathcal{C}$ to be $+1$ if $\mathcal{C}$ is even, and -1 if $\mathcal{C}$ is odd. Cauchy showed that with this definition, the sign of a permutation (based on inversions) and the sign of the associated cycle cover is the same. For our use, this definition of sign based on cycle covers will be more convenient.

Let $\mathcal{C}(G_A)$ denote the set of all cycle covers in the graph G_A. Then we have

$$det(A) = \sum_{\mathcal{C} \in \mathcal{C}(G_A)} sgn(\mathcal{C}) w(\mathcal{C}) = \sum_{\mathcal{C} \in \mathcal{C}(G_A)} sgn(\mathcal{C}) \prod_{\langle i,j\rangle \in \mathcal{C}} a_{ij}$$

Consider the characteristic polynomial of A,

$$\chi_A(\lambda) = \det(\lambda I_n - A) = c_0\lambda^n + c_1\lambda^{n-1} + \ldots + c_{n-1}\lambda + c_n$$

To interpret these coefficients, consider the graph $G_A(\lambda)$ whose edges are labeled according to the matrix $\lambda I_n - A$. The coefficient c_l collects part of the contribution to $\det(\lambda I_n - A)$ from cycle covers having at least $(n-l)$ self loops. (A self loop at vertex k now carries weight $\lambda - a_{kk}$.) This is because a cycle cover with i self loops has weight which is a polynomial of degree i in λ. For instance, with $n = 4$, consider the cycle cover $\langle 1,4\rangle, \langle 2,2\rangle, \langle 3,3\rangle, \langle 4,1\rangle$ in $G_A(\lambda)$. This has weight $(-a_{14})(\lambda - a_{22})(\lambda - a_{33})(-a_{41})$, contributing $a_{14}a_{22}a_{33}a_{41}$ to c_4, $-a_{14}a_{41}(a_{22} + a_{33})$ to c_3, $a_{14}a_{41}$ to c_2, and 0 to c_1.

Following Straubing's notation, we consider partial permutations, corresponding to partial cycle covers. A partial permutation σ is a permutation on a subset $S \subseteq [n]$. The set S is called the domain of σ, denoted $\mathrm{dom}(\sigma)$. The completion of σ, denoted $\hat{\sigma}$, is the permutation in S_n obtained by letting all elements outside $\mathrm{dom}(\sigma)$ be fixed points. This permutation $\hat{\sigma}$ corresponds to a cycle cover $\mathcal{C}$ in G_A, and σ corresponds to a subset of the cycles in $\mathcal{C}$. We call such a subset a partial cycle cover $\mathcal{PC}$, and we call $\mathcal{C}$ the completion of $\mathcal{PC}$. A partial cycle cover is defined to have the same parity and sign as its completion. It is easy to see that the completion need not be explicitly accounted for in the parity; a partial cycle cover $\mathcal{PC}$ is even (resp. odd) iff the number of cycles in it, plus the number of edges in it, is even (resp. odd).

Getting back to the characteristic polynomial, observe that to collect the contributions to c_l, we must look at all partial cycle covers with l edges. The $n - l$ vertices left uncovered by such a partial cycle cover $\mathcal{PC}$ are the self-loops, from whose weight the λ term has been picked up. Of the l vertices covered, self-loops, if any, contribute the $-a_{kk}$ term from their weight, not the λ term. And other edges, say $\langle i,j\rangle$ for $i \neq j$, contribute weights $-a_{ij}$. Thus the weights for $\mathcal{PC}$ evidently come from the graph G_{-A}. If we interpret weights over the graph G_A, a factor of $(-1)^l$ must be accounted for independently.

Formally,

Definition 1. *A cycle is an ordered sequence of m edges $C = \langle e_1, e_2, \ldots, e_m\rangle$ where $e_i = \langle u_i, u_{i+1}\rangle$ for $i \in [m-1]$ and $e_m = \langle u_m, u_1\rangle$ and $u_1 \leq u_i$ for $i \in [m]$ and all the u_i's are distinct. u_1 is called the head of the cycle, denoted $h(C)$. The length of the cycle is $|C| = m$, and the weight of the cycle is $w(C) = \prod_{i=1}^{m} w(e_i)$. The vertex set of the cycle is $V(C) = \{u_1, \ldots, u_m\}$.*

An l-cycle cover $\mathcal{C}$ is an ordered sequence of cycles $\mathcal{C} = \langle C_1, \ldots, C_k\rangle$ such that $V(C_i) \cap V(C_j) = \phi$ for $i \neq j$, $h(C_1) < \ldots < h(C_k)$ and $|C_1| + \ldots + |C_k| = l$.

The weight of the l-cycle cover is $wt(\mathcal{C}) = \prod_{j=1}^{k} w(C_j)$, and the sign is $sgn(\mathcal{C}) = (-1)^{l+k}$.

As a matter of convention, we call n-cycle covers simply cycle covers.

Proposition 1. *The coefficients of $\chi_A(\lambda)$ are given by*

$$c_l = (-1)^l \sum_{\mathcal{C} \text{ is an } l\text{-cycle cover in } G_A} sgn(\mathcal{C})wt(\mathcal{C})$$

3 Summing over permutations efficiently

As noted in Proposition 1, evaluating the determinant (or for that matter, any coefficient of the characteristic polynomial) amounts to evaluating the signed weighted sum over cycle covers (partial cycle covers of appropriate length). We consider four efficient algorithms for computing this sum. Each expands this sum to include more terms which mutually cancel out. The differences between the algorithms is essentially in the extent to which the sum is expanded.

3.1 From Cycle Covers to Clow Sequences

Genaralize the notion of a cycle and a cycle cover as follows:

A *clow* (for closed-walk) is a cycle in G_A (not necessarily simple) with the property that the minimum vertex in the cycle – called the *head* – is visited only once. An l-clow sequence is a sequence of clows where the heads of the clows are in strictly increasing order and the total number of edges (counting each edge as many times as it is used) is l. Formally,

Definition 2. *A clow is an ordered sequence of edges $C = \langle e_1, e_2, \ldots, e_m \rangle$ such that $e_i = \langle u_i, u_{i+1} \rangle$ for $i \in [m-1]$ and $e_m = \langle u_m, u_1 \rangle$ and $u_1 \neq u_j$ for $j \in \{2, \ldots, m\}$ and $u_1 = min\{u_1, \ldots, u_m\}$. The vertex u_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 $w(C) = \prod_{i=1}^{m} w(e_i)$.*

An l-clow sequence $\mathcal{C}$ is an ordered sequence of clows $\mathcal{C} = \langle C_1, \ldots, C_k \rangle$ such that $h(C_1) < \ldots < h(C_k)$ and $|C_1| + \ldots + |C_k| = l$.

The weight of the l-clow sequence is $wt(\mathcal{C}) = \prod_{j=1}^{k} w(C_j)$, and the sign is $sgn(\mathcal{C}) = (-1)^{l+k}$.

Note that the set of l-clow sequences properly includes the set of l-cycle covers on a graph. And the sign and weight of a cycle cover are consistent with its sign and weight when viewed as a clow sequence.

The theorem below establishes a connection between the coefficients of a characteristic polynomial and clow sequences.

Theorem 1 (Theorem 2.1 in [MV97]).

$$c_l = (-1)^l \sum_{\mathcal{C} \text{ is an } l\text{-clow sequence}} sgn(\mathcal{C}) wt(\mathcal{C})$$

3.2 Clow Sequences with the Prefix Property: Getting to Samuelson's Method

The generalization from cycle covers to clow sequences has a certain extravagance. The reason for going to clow sequences is that evaluating their wieghted sum is easy, and this sum equals the sum over cycle covers. However, there are several clow sequences which we can drop from consideration without sacrificing ease of computation. One such set arises from the following consideration:

In a cycle cover, all vertices are covered exactly once. Suppose we enumerate the vertices in the order in which they are visited in the cycle cover (following the order imposed by the cycle heads). If vertex h becomes the head of a cycle, then all vertices in this and subsequent cycles are larger than h. So all the lower numbered vertices must have been already visited. So at least $h-1$ vertices, and hence $h-1$ edges, must have been covered.

We can require our clow sequences also to satisfy this property. We formalize the prefix property: a clow sequence $\mathcal{C} = \langle C_1, \ldots, C_k \rangle$ has the prefix property if for $1 \leq r \leq k$, the total lengths of the clows $C_1, \ldots, C_{r-1}$ is at least $h(C_r) - 1$. A similar prefix property can be formalized for partial cycle covers. Formally,

Definition 3. *An l-clow sequence $\mathcal{C} = \langle C_1, \ldots, C_k \rangle$ is said to have the prefix property if it satisfies the following condition:*

$$\forall r \in [k], \quad \sum_{t=1}^{r-1} |C_t| \geq h(C_r) - 1 - (n - l)$$

The interesting fact is that the involution constructed in the previous subsection for clow sequences works even over this restricted set!

Theorem 2 (Theorem 2 in [Val92]).

$$c_l = (-1)^l \sum_{\substack{\mathcal{C} \text{ is an } l\text{-clow sequence} \\ \text{with the prefix property}}} sgn(\mathcal{C})wt(\mathcal{C})$$

In [Val92], Valiant observes that prefix clow sequences are the terms computed by Samuelson's method for evaluating $\chi_\lambda(A)$. Hence the correctness of the theorem follows from the correctness of Samuelson's method. And the correctness of Samuelson's method is traditionally shown using linear algebra. A simple alternative combinatorial proof of this theorem is shown in the appendix.

Algorithm using prefix clow sequences To compute c_l using this characterization, we must sum up the contribution of all l-clow sequences with the prefix property. One way is to modify the dynamic programming approach used in the previous sub-section for clow sequences. This can be easily done. Let us instead do things differently; the reason will become clear later.

Adopt the convention that there can be clows of length 0. Then each l-clow sequence $\mathcal{C}$ has exactly one clow C_i with head i, for $i = 1$ to n. So we write $\mathcal{C} = \langle C_1, \ldots, C_n \rangle$.

Define the signed weight of a clow C as $sw(C) = -w(C)$ if C has non-zero length, and $sw(C) = 1$ otherwise. And define the signed weight of an l-clow sequence as $sw(\mathcal{C}) = \prod_{i=1}^{n} sw(C)$. Then $sgn(\mathcal{C})w(\mathcal{C}) = (-1)^l sw(\mathcal{C})$. So from the above theorem,

$$c_l = \sum_{\substack{\mathcal{C} \text{ is an } l\text{-clow sequence} \\ \text{with the prefix property}}} sw(\mathcal{C})$$

We say that a sequence of non-negative integers $l_1, \ldots, l_n$ satisfies property prefix(l) if
(1). $\sum_{t=1}^{n} l_t = l$, and
(2). For $r \in [n]$, $\sum_{t=1}^{r-1} l_t \geq r - 1 - (n - l)$. Alternatively $\sum_{t=r}^{l} l_t \leq n - r + 1$.
Such sequences are "allowed" as lengths of clows in the clow sequences we construct; no other sequences are allowed.

We group the clow sequences with prefix property based on the lengths of the individual clows. In a clow sequence with prefix property $\mathcal{C}$, if the length of clow C_i (the possibly empty clow with head i) is l_i, then any clow with head i and length l_i can replace C_i in $\mathcal{C}$ and still give a clow sequence satisfying the prefix property. Thus, if $z(i,p)$ denotes the total signed weight of all clows which have vertex i as head and length p, then

$$c_l = \sum_{l_1,\ldots,l_n:\text{prefix}(l)} \prod_{i=1}^{n} z(i, l_i)$$

To compute c_l efficiently, we place the values $z(i,p)$ appropriately in a series of matrices $B_1, \ldots, B_n$. The matrix B_k has entries $z(k,p)$. Since we only consider sequences satisfying prefix(l), it suffices to consider $z(k,p)$ for $p \leq n - k + 1$. Matrix B_k is of dimension $(n-k+2) \times (n-k+1)$ and has $z(k,p)$ on the pth lower diagonal as shown below.

$$B_k = \begin{bmatrix} z(k,0) & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & 0 \\ z(k,n-k+1) & z(k,n-k) & z(k,n-k-1) & \cdots & z(k,2) & z(k,1) \end{bmatrix}$$

Now from the equation for c_l, it is clear that

$$[\ c_0\ c_1\ c_2\ c_3\ \cdots\ \cdots\ c_n\ ]^T = \prod_{k=1}^{n} B_k$$

It remains now to compute $z(i,p)$, the entries in the B matrices. We know that $z(i,0) = 1$ and $z(i,1) = -a_{ii}$. For $p \geq 2$, a clow of length p with head i must first visit a vertex $u > i$, then perform a walk of length $p - 2$ via vertices greater than i to some vertex $v > i$, and then return to i. To construct the path, we exploit the fact that the (j,k)th entry in a matrix A^p gives the sum of the weights of all paths in G_A of length exactly p from j to k. So we must consider the induced subgraph with vertices $i+1, \ldots, n$. This has an adjacency matrix A_{i+1} obtained by removing the first i rows and the first i columns of A. So $A_1 = A$. Consider the submatrices of A_i as shown below.

$$A_i = \begin{pmatrix} a_{ii} & R_i \\ S_i & A_{i+1} \end{pmatrix}$$

Then the clows contributing to $z(i,p)$ must use an edge in R_i, perform a walk corresponding to A_{i+1}^{p-2}, and then return to i via an edge in S_i. In other words,

$$z(i,p) = -R_i\ A_{i+1}^{p-2}\ S_i$$

So the matrices B_k look like this:

$$B_k = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ -a_{kk} & 1 & & \cdots & 0 & 0 \\ -R_kS_k & -a_{kk} & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & 0 \\ \vdots & \vdots & \vdots & & \vdots & 0 \\ -R_kA_{k+1}^{n-k-2}S_k & -R_kA_{k+1}^{n-k-3}S_k & -R_kA_{k+1}^{n-k-4}S_k & \cdots & -a_{kk} & 1 \\ -R_kA_{k+1}^{n-k-1}S_k & -R_kA_{k+1}^{n-k-4}S_k & -R_kA_{k+1}^{n-k-3}S_1 & \cdots & -R_kS_k & -a_{kk} \end{bmatrix}$$

This method of computing $\chi_A(\lambda)$ is precisely Samuelson's method [Sam42] (see [FF63,Ber84,Val92]). Samuelson arrived at this formulation using Laplace's Theorem on the matrix $\lambda I - A$, whereas we have arrived at it via clow sequences with the prefix property. This interpretation of the Samuelson-Berkowitz algorithm is due to Valiant[Val92]; the combinatorial proof of correctness (proof of Theorem 2) is new. (It is mentioned, without details, in [MV97].)

3.3 From Clows to Tour Sequences Tables: Getting to Chistov's Algorithm

We now move in the other direction – generalize further beyond clow sequences. Firstly, we relax the condition that the head of a clow may be visited only once. This gives us more generalized closed walks which we call tours. To fix a canonical representation, we do require the edges of the tour to be listed beginning from an occurrence of the head. Since there could be multiple such occurrences, we get different tours with the same multiset of edges. eg the tour corresponding to the vertex sequence 253246 is different from the tour corresponding to the vertex sequence 246253. Secondly, we deal with not just sequences but ordered lists, or tables, of sequences. Within a sequence the tours are ordered by their heads (and all heads are distinct), but there is no ordering amongst the different sequences. And the parity of a tour sequence table depends on the number of sequences in it, not the number of tours in it. A clow sequence is thus a tour sequence table where each sequence contains a single tour which is a clow and the sequences are ordered by their tour heads. Formally,

Definition 4. *A tour is an ordered sequence of edges $C = \langle e_1, e_2, \ldots, e_p\rangle$ such that $e_i = \langle u_i, u_{i+1}\rangle$ for $i \in [p-1]$ and $e_p = \langle u_p, u_1\rangle$ and $u_1 = min\{u_1, \ldots, u_m\}$. The vertex u_1 is called the head of the tour and denoted $h(C)$. The length of the tour is $|T| = p$, and the weight of the tour is $wt(T) = \prod_{i=1}^{m} w(e_i)$.*

A j-tour sequence $\mathcal{T}$ is an ordered sequence of tours $\mathcal{T} = \langle T_1, \ldots, T_k\rangle$ such that $h(T_1) < \ldots < h(T_k)$ and $|T_1|+\ldots+|T_k| = j$. The weight of the tour sequence is $wt(\mathcal{T}) = \prod_{j=1}^{k} wt(T_j)$, and the length is $|\mathcal{T}| = j$.

An l-tour sequence table TST is an ordered sequence of tour sequences $\mathcal{F} = \langle \mathcal{T}_1, \ldots, \mathcal{T}_r \rangle$ such that $|\mathcal{T}_1| + \cdots + |\mathcal{T}_r| = l$. The weight of the TST is $wt(\mathcal{F}) = \prod_{j=1}^{r} wt(\mathcal{T}_j)$, and the sign is $(-1)^{l+r}$.

The following theorem shows that TSTs can be used to compute the characteristic polynomial.

Theorem 3.

$$c_l = (-1)^l \sum_{\mathcal{F} \text{ is an } l\text{-}TST} sgn(\mathcal{F}) wt(\mathcal{F})$$

Proof. We demonstrate an involution on the set of l-TSTs with all l-clow sequences being fixed points, and all other l-TSTs being mapped to TSTs of the same weight but opposing sign. Since l-clow sequences which are not cycle covers also yield a net contribution of zero (Theorem 1), the sum over all l-TSTs is precisely c_l.

Given an l-TST $\mathcal{F} = \langle \mathcal{T}_1, \ldots, \mathcal{T}_r \rangle$, let H be the set of all vertices which occur as heads of some tour in the table. For $S \subseteq H$, we say that S has the *clow sequence property* if the following holds: There is an $i \leq r$ such that
(1). The tour sequences $\mathcal{T}_{i+1}, \ldots, \mathcal{T}_r$ are all single-tour sequences
(say tour sequence $\mathcal{T}_j$ is the tour T_j),
(2). No tour in any of the tour sequences $\mathcal{T}_1, \ldots, \mathcal{T}_i$ has a head vertex in S,
(3). Each vertex in S is the head of a tour T_j for some $i+1 \leq j \leq r$. i.e. $\{h(T_j) \mid j = i+1, \ldots, r\} = S$,
(4). The tour sequence table $\langle \mathcal{T}_{i+1}, \ldots, \mathcal{T}_r \rangle$ actually forms a clow sequence.
i.e. the tours T_j for $i+1 \leq j \leq r$ are clows, and $h(T_{i+1}) < \ldots < h(T_r)$.
In other words, all tours in $\mathcal{F}$ whose heads are in S are actually clows which occur in a contiguous block of single-tour sequences, arranged in strictly increasing order of heads, and this block is not followed by any other tour sequences in $\mathcal{F}$.

Now, in H, find the smallest vertex v such that $H_{>v} = \{h \in H \mid h > v\}$ has the clow sequence property but $H_{\geq v} = \{h \in H \mid h \geq v\}$ does not.

If no such v exists, then H satisfies the clow sequence property and hence $\mathcal{F}$ is an l-clow sequence. In this case, map it to itself.

If such a v exists, then locate the first tour sequence $\mathcal{T}_i = \langle T_1, \ldots, T_k \rangle$ where v appears (as a head). Then v is the head of the last tour T_k, because all tours with larger heads occur in a contiguous block of single-tour sequences at the end. The tour T_k can be uniquely decomposed as TC, where T is a tour and C a clow, both with head v.

Case 1: $T \neq \phi$. Map this l-TST to an l-TST where $\mathcal{T}_i$ is replaced, at the same position, by the following two tour sequences: $\langle C \rangle, \langle T_1, \ldots, T_{k-1}, T \rangle$. This preserves weight but inverts the sign. In the modified l-TST, the newly introduced sequence containing only C will be chosen for modification as in Case 3.

Case 2: $T = \phi$, and $k > 1$. Map this l-TST to an l-TST where $\mathcal{T}_i$ is replaced, at the same position, by the following two tour sequences: $\langle C \rangle, \langle T_1, \ldots, T_{k-1} \rangle$. This too preserves weight but inverts the sign. In the modified l-TST, the newly introduced sequence containing only C will be chosen for modification as in Case 3.

Case 3: $T = \phi$ **and** $k = 1$. Then a tour sequence $\mathcal{T}_{i+1}$ must exist, since otherwise $H_{\geq v}$ would satisfy the clow sequence property. Now, if $\mathcal{T}_{i+1}$ has a tour with head greater than v, then, since $H_{>v}$ satisfies the clow sequence property, the TST $\mathcal{T}_{i+1}, \ldots, \mathcal{T}_r$ must be a clow sequence. But recall that T has the first occurrence of v as a head and is itself a clow, so then $\mathcal{T}_i, \ldots, \mathcal{T}_r$ must also be a clow sequence, and $H_{\geq v}$ also satisfies the clow sequence property, contradicting our choice of v. So $\mathcal{T}_{i+1}$ must have all tours with heads at most v. Let $\mathcal{T}_{i+1} = \langle P_1, \ldots, P_s \rangle$. Now there are two sub-cases depending on the head of the last tour P_s.
Case 3(a). $h(P_s) = v$. Form the tour $P'_s = P_s C$. Map this l-TST to a new l-TST where the tour sequences $\mathcal{T}_i$ and $\mathcal{T}_{i+1}$ are replaced, at the same position, by a single tour sequence $\langle P_1, \ldots, P_{s-1}, P'_s \rangle$. The weight is preserved and the sign inverted, and in the modified l-TST, the tour P'_s in this new tour sequence will be chosen for modification as in Case 1.
Case 3(b). $h(P_s) \neq v$. Map this l-TST to a new l-TST where the tour sequences $\mathcal{T}_i$ and $\mathcal{T}_{i+1}$ are replaced, at the same position, by a single tour sequence $\langle P_1, \ldots, P_s, C \rangle$. The weight is preserved and the sign inverted, and in the modified l-TST, the tour C in this new tour sequence will be chosen for modification as in Case 2.

Thus l-TSTs which are not l-clow sequences yield a net contribution of zero.

Algorithm using tour sequence tables We show how grouping the l-TSTs in a carefully chosen fashion gives a formulation which is easy to compute.

Define $e_l = (-1)^l c_l$, then

$$e_l = \sum_{\mathcal{F} \text{ is an } l\text{-TST}} sgn(\mathcal{F}) wt(\mathcal{F})$$

To compute c_l and hence e_l using this characterization, we need to compute the contributions of all l-TSTs. This is more easily achieved if we partition these contributions into l groups depending on how many edges are used up in the first tour sequence of the table. Group j contains l-TSTs of the form $\mathcal{F} = \langle \mathcal{T}_1, \ldots, \mathcal{T}_r \rangle$ where $|\mathcal{T}_1| = j$. Then $\mathcal{F}' = \langle \mathcal{T}_2, \ldots, \mathcal{T}_r \rangle$ forms an $(l-j)$-TST, and $sgn(\mathcal{F}) = -sgn(\mathcal{F}')$ and $wt(\mathcal{F}) = wt(\mathcal{T}_1) wt(\mathcal{F}')$. So the net contribution to e_l from this group, say $e_l(j)$, can be factorized as

$$\begin{aligned} e_l(j) &= \sum_{\substack{\mathcal{T}\text{: } j\text{-tour sequence} \\ \mathcal{F}'\text{: } (l-j)\text{-TST}}} -sgn(\mathcal{F}') wt(\mathcal{F}') wt(\mathcal{T}) \\ &= -\left(\sum_{\mathcal{T}\text{: } j\text{-tour sequence}} wt(\mathcal{T})\right) \left(\sum_{\mathcal{F}'\text{: } (l-j)\text{-TST}} sgn(\mathcal{F}') wt(\mathcal{F}')\right) \\ &= -d_j e_{l-j} \end{aligned}$$

where d_j is the sum of the weights of all j-tour sequences.

Now we need to compute d_j.

It is easy to see that $A^l[1,1]$ gives the sum of the weights of all tours of length l with head 1. To find a similar sum over tours with head k, we must consider the induced subgraph with vertices $k, k+1, \ldots, n$. This has an adjacency matrix A_k obtained by removing the first $k-1$ rows and the first $k-1$ columns of A.

(We have already exploited these properties in Section 3.2.) Let $y(l, k)$ denote the sum of the weights of all l-tours with head k. Then $y(l, k) = A_k^l[1, 1]$.

The weight of a j-tour sequence $\mathcal{T}$ can be split into n factors: the kth factor is 1 if $\mathcal{T}$ has no tour with head k, and is the weight of this (unique) tour otherwise. So

$$\begin{aligned} d_j &= \textstyle\sum_{0 \le l_i \le j:\ l_1 + \ldots + l_n = j} \ \prod_{i=1}^n y(l_i, i) \\ &= \textstyle\sum_{0 \le l_i \le j:\ l_1 + \ldots + l_n = j} \ \prod_{i=1}^n A_i^{l_i}[1, 1] \end{aligned}$$

Let us define a power series $D(x) = \sum_{j=0}^{\infty} d_j x^j$. Then, using the above expression for d_j, we can write

$$D(x) = \left(\sum_{l=0}^{\infty} x^l A_1^l[1,1]\right) \left(\sum_{l=0}^{\infty} x^l A_2^l[1,1]\right) \cdots \left(\sum_{l=0}^{\infty} x^l A_n^l[1,1]\right)$$

Since we are interested in d_j only for $j \le n$, we can ignore monomials of degree greater than n. This allows us to evaluate the first $n + 1$ coefficients of $D(x)$ using matrix powering and polynomial arithmetic. And now e_l can be computed inductively using the following expression:

$$e_l = \sum_{j=1}^{l} e_l(j) = \sum_{j=1}^{l} -d_j e_{l-j}$$

But this closely matches Chistov's algorithm [Chi85]! The only difference is that Chistov started off with various algebraic entities, manipulated them using polynomial arithmetic, and derived the above formulation, whereas we started off with TSTs which are combinatorial entities, grouped them suitably, and arrived at the same formulation. And at the end, Chistov uses polynomial arithmetic to combine the computation of $D(x)$ and e_l.The full paper has a self-contained algebraic exposition of Chistov's algorithm to expose the contrast of the two proofs; the reader may also consult [Koz92] for an algebraic treatment.

Acknowledgment

We thank Ashok Subramanian for showing how our combinatorial interpretations and construction in the clow sequences approach extend to Chistov's algorithm as well.

References

[Ber84] S. J. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Information Processing Letters*, 18:147–150, 1984.

[Cha82] S. Chaiken. A combinatorial proof of the all minors matrix theorem. *SIAM J. Algebraic Discrete Methods*, 3:319–329, 1982.

[Chi85] A. L. Chistov. Fast parallel calculation of the rank of matrices over a field of arbitrary characteristic. In *Proc Int. Conf. Foundations of Computation Theory, LNCS 199*, pages 63–69. Springer, 1985.

[Csa76] L. Csanky. Fast parallel inversion algorithm. *SIAM J of Computing*, 5:818–823, 1976.

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

[FF63] D. Fadeev and V. Fadeeva. *Computational Methods in Linear Algebra.* Freeman, San Francisco, 1963.

[Foa65] D. Foata. Etude algébrique de certains problèmes d'analyse combinatoire et du calcul des probabilités. *Publ. Inst. Statist. Univ. Paris*, 14:81–241, 1965.

[Foa80] D. Foata. A combinatorial proof of Jacobi's identity. *Ann. Discrete Math.*, 6:125–135, 1980.

[GE] A. Garsia and Ö. Eğecioğlu. Combinatorial foundations of computer science. unpublished collection.

[Ges79] I. Gessel. Tournaments and Vandermonde's determinant. *J Graph Theory*, 3:305–307, 1979.

[Koz92] D. Kozen. *The Design and Analysis of Algorithms.* Springer-Verlag, New York, 1992.

[Lei92] T. F. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes.* Morgan Kaufmann Publishers Inc., San Mateo, 1992.

[Min97] M. Minoux. Bideterminants, arborescences and extension of the matrix-tree theorem to semirings. *Discrete Mathematics*, 171:191–200, 1997.

[MV97] M. Mahajan and V Vinay. Determinant: combinatorics, algorithms, complexity. *Chicago Journal of Theoretical Computer Science* `http://cs-www.uchicago.edu/publications/cjtcs`, 1997:5, 1997. A preliminary version appeared as "A combinatorial algorithm for the determinant" in Proceedings of the *Eighth Annual ACM-SIAM Symposium on Discrete Algorithms* SODA97.

[Orl78] J. B. Orlin. Line-digraphs, arborescences, and theorems of Tutte and Knuth. *J. Combin. Theory Ser. B*, 25:187–198, 1978.

[Sam42] P. A. Samuelson. A method of determining explicitly the coefficients of the characteristic polynomial. *Ann. Math. Stat.*, 13:424–429, 1942.

[Str73] V. Strassen. Vermeidung von divisionen. *Journal of Reine U. Angew Math*, 264:182–202, 1973.

[Str83] H. Straubing. A combinatorial proof of the Cayley-Hamilton theorem. *Discrete Maths.*, 43:273–279, 1983.

[SW86] D. Stanton and D. White. *Constructive Combinatorics.* Springer-Verlag, 1986.

[Tem81] H. N. V. Tempereley. *Graph Theory and Applications.* Ellis Horwood, Chichester, 1981.

[Tod91] S. Toda. Counting problems computationally equivalent to the determinant. manuscript, 1991.

[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.

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

[Zei85] D. Zeilberger. A combinatorial approach to matrix algebra. *Discrete Mathematics*, 56:61–72, 1985.