

Totally Nonnegative Matrices, Graphs, and the Temperley-Lieb Algebra

REU Spring/Summer Research
May – August 2002

Randolph Pistor
Conducted Under Professor Mark Skandera
The University of Michigan

ABSTRACT. We explore the poset of products of two minors in totally nonnegative matrices. In doing so, we employ the n th Temperley-Lieb algebra and set partitions to describe the elements of this poset.

1. Introduction

Let A be an $(n \times n)$ matrix and let I and I' be subsets of $[n] = \{1, \dots, n\}$. We define $\Delta_{I, I'}$, the (I, I') minor of A , to be the determinant of the submatrix of A corresponding to rows I and columns I' . A matrix is called *totally nonnegative* if each of its minors is nonnegative. For instance, the matrix

$$\begin{bmatrix} 3 & 3 & 2 & 2 & 1 \\ 3 & 3 & 2 & 2 & 1 \\ 5 & 5 & 4 & 4 & 3 \\ 5 & 5 & 4 & 4 & 3 \\ 5 & 5 & 4 & 4 & 3 \end{bmatrix}$$

Figure 1.1.

is totally nonnegative, a fact that can be verified computationally.

One of the most important examples of a totally nonnegative matrix arises in the study of directed graphs. We define a *planar network of order n* to be a planar acyclic directed graph G with $2n$ distinguished boundary vertices. The vertices of a planar network are labeled counterclockwise as $s_1, \dots, s_n, t_n, \dots, t_1$. The vertices s_1, \dots, s_n are referred to as *sources*, and the vertices t_1, \dots, t_n are referred to as *sinks*. An example of a planar network, taken from an introduction of Mark Skandera [13, Figure 1.2], is given in Figure 1.2.

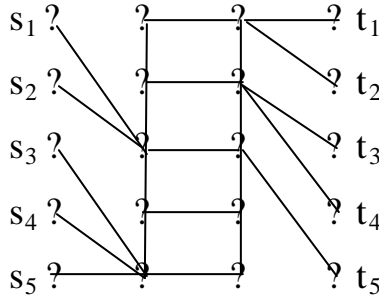


Figure 1.2. A planar network of order 5

In this example, the vertical edges are oriented from bottom to top, and other edges are oriented to the right.

Given a planar network G of order n , we define the *path matrix* of G to be the $(n \times n)$ matrix $A = [a_{ij}]$, in which the element a_{ij} is the number of paths from s_i to t_j . More generally, suppose each edge e in G is labeled by a real positive *weight* w_e . From this, we define the weight of a *path* from s_i to t_j to be the product of weights of edges along this path. We can then define the *weighted path matrix* of G to be the $(n \times n)$ matrix $A = [a_{ij}]$, where a_{ij} is the sum of weights of all paths from s_i to t_j . Thus, a path matrix is a weighted path matrix where $w_e = 1$ for all edges e . Such a weighted path matrix is always totally nonnegative, as stated in the following result.

Theorem 1.1. *Let A be the weighted path matrix of a planar network G . The minor $\Delta_{I,I'}$ of A is equal to the sum of weights of all nonintersecting path families which connect the sources indexed by I to the sinks indexed by I' . In particular, A is totally nonnegative.*

The first proofs of this fact were given by Karlin and MacGregor [7], and also by Lindström [8]. Since then, Theorem 1.1 has been used to prove that matrices arising in various situations are totally nonnegative. For example, observe that the matrix in Figure 1.1 is the path matrix for the planar network in Figure 1.2. This again verifies that the matrix in Figure 1.1 is totally nonnegative. (For further examples of applications see [5] and [6]. See also [4] for tests used to determine if a matrix is totally nonnegative.)

The converse of Theorem 1.1 is true as well, and was proven combining the results of Whitney [15], Loewner [9], Cryer [2] and Brenti [1]. This result was first stated in [1].

Theorem 1.2. *Every $(n \times n)$ totally nonnegative matrix A is the weighted path matrix of a planar network G of order n .*

The concept of a totally nonnegative matrix can be extended to include polynomials. We say that a polynomial in n^2 variables $\{x_{ij} \mid i, j \in [n]\}$ is *totally nonnegative* if the polynomial evaluates to a nonnegative number whenever we set $x_{ij} = a_{ij}$ for a totally nonnegative $(n \times n)$ matrix A . In relation to totally nonnegative polynomials, a result by Lusztig [10] has shown that the elements of the dual canonical basis of the coordinate

ring GL_n are totally nonnegative. However, this basis currently has no simple description. To better understand this basis, it would be especially helpful to characterize all totally nonnegative polynomials. A special case of interest is to characterize the totally nonnegative polynomials of the form

$$\Delta_{J, J'} \Delta_{\bar{I}, \bar{J}'} - \Delta_{I, I'} \Delta_{\bar{I}, \bar{J}'}$$

where I is an n -element subset of $[2n]$ and $I' = [2n] \setminus I$, i.e. I' is unique complement of I . See the paper by Fallat, Gekhtman and Johnson [3] for the history of these recently discovered polynomials.

Next, we define a *partially ordered set*, or a *poset*, as a set with a transitive order relation $<_p$ such that any two distinct elements x and y are related as

$$\begin{aligned} &x <_p y, \\ &y <_p x, \\ &x \text{ and } y \text{ are incomparable.} \end{aligned}$$

As such, a poset can also be viewed as a set P of points connected by lines or arrows. For elements x and y in P , we say that $x <_p y$ if there is a path from x to y , and we say that x and y are incomparable if no such path exists. An example of a poset is given in Figure 1.3.

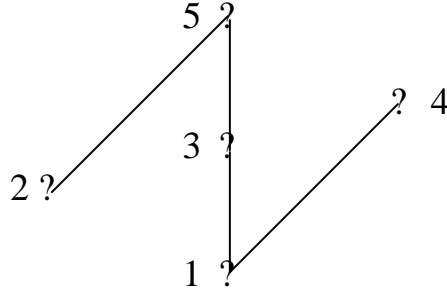


Figure 1.3. A poset of 5 elements

Here we have: $1 <_p 2$, $1 <_p 3$, $1 <_p 4$, $2 <_p 5$, $3 <_p 5$.

In relation to posets, the products of minors of totally nonnegative matrices form a partial order P . We say that $\Delta_{I, I'} \Delta_{\bar{I}, \bar{J}'} =_{\mathbb{P}} \Delta_{J, J'} \Delta_{\bar{I}, \bar{J}'}$ if $\Delta_{I, I'} \Delta_{\bar{I}, \bar{J}'} = \Delta_{J, J'} \Delta_{\bar{I}, \bar{J}'}$ for all totally nonnegative matrices. Based on this definition, we arrive at the following question.

Question 1.1. What does this partial order P look like?

In attempting to describe the answer to Question 1.1, which is explored further in Section 5, we employ the Temperley-Lieb algebra. We define the *n*th *Temperley-Lieb*

algebra, denoted by T_n , as the set of all pictures on $2n$ vertices, n vertical + n vertical, and n non-crossing curves. T_3 is drawn in Figure 1.4.

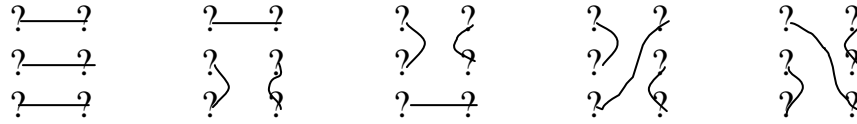


Figure 1.4. The elements of T_3

For a given positive integer n , the number of elements in T_n is the n th term of the sequence known as the Catalan numbers, where a_n is given by the formula

$$\frac{1}{n+1} \binom{2n}{n}$$

The operation defined on T_n is concatenation, expressed with the operation symbol $?$. An example of concatenation between two elements of T_3 is illustrated in Figure 1.5.

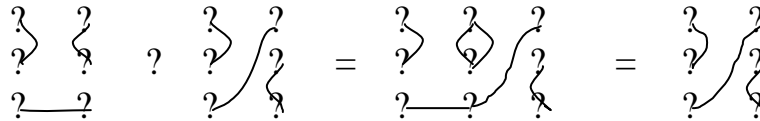


Figure 1.5.

With concatenation, loops are ignored and long curves are shortened to normal length.

It is also possible to draw the elements of T_n as *circle diagrams*, where the elements of T_n are drawn as $2n$ dots on a circle connected by line segments. When drawn, the dots form the vertices of a regular $2n$ -gon. T_3 is drawn as circle diagrams in Figure 1.6.

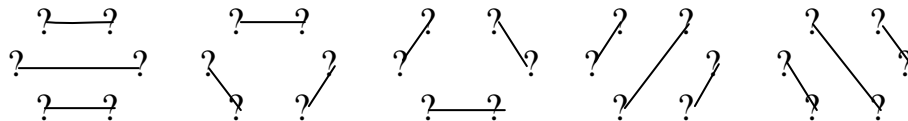


Figure 1.6.

The main results of this paper explore the relationship between graph theory and elements of T_n suggested by the examples in Figure 1.6.

2. Special Subsets of the Temperley-Lieb Algebra

In this section we examine subsets of T_n that relate to planar networks and totally nonnegative matrices. In particular, we consider the subsets of T_n corresponding to the coloring rule (I, \bar{I}) , where I is an n -element subset of $[2n] = \{1, \dots, 2n\}$, $\bar{I} = [2n] \setminus I$, and

$$\begin{aligned} I &= \text{the set of all 'black' dots,} \\ \bar{I} &= \text{the set of all 'white' dots.} \end{aligned}$$

See Figure 2.1. We refer to this selection of n black dots and n white dots as a *dot diagram* D , where the dots are labeled counterclockwise.

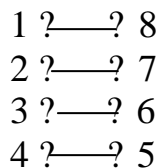


Figure 2.1. A dot diagram D with $I = \{1, 2, 4, 6\}$ and $\bar{I} = \{3, 5, 7, 8\}$

We say that an element τ of T_n is *compatible* with D if no edge of τ connects two vertices of the same color. Such an element is said to lie in the *special subset* S of T_n corresponding to D . Thus, every special subset of T_n corresponds to a selection of n black dots and n white dots and all elements of T_n such that each curve has a white endpoint and a black endpoint. The special subset corresponding to D in Figure 2.1 is given in Figure 2.2.

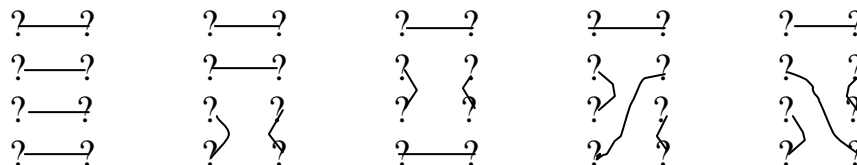


Figure 2.2.

In addition, the special subset S corresponding to D , that is, the coloring rule (I, \bar{I}) , is also written as $U(I)$, where U is a function that maps the rule (I, \bar{I}) to a special subset of T_n .

A result by Mark Skandera [14, Obs. 2.3, Cor. 3.4] demonstrates that a direct relation exists between totally nonnegative matrices and special subsets of T_n .

Theorem 2.1. $\Delta_{I, J} \Delta_{\bar{I}, \bar{J}} = \Delta_{J, I} \Delta_{\bar{J}, \bar{I}}$ if and only if the subset $U(I)$ of T_n corresponding to coloring dots labeled by I is strictly contained in the subset $U(J)$ of T_n corresponding to coloring dots labeled by J .

Example 2.1. We use the symbol $\bar{?}$ to denote strictly contained. Let $I = \{1, 2, 3, 4\}$ and let $J = \{1, 2, 4, 6\}$. Then we have

$$\begin{array}{l}
 U(J) = \begin{array}{ccccc}
 \bar{?} & \bar{?} & \bar{?} & \bar{?} & \bar{?} \\
 \bar{?} & \bar{?} & \bar{?} & \bar{?} & \bar{?} \\
 \bar{?} & \bar{?} & \bar{?} & \bar{?} & \bar{?} \\
 \bar{?} & \bar{?} & \bar{?} & \bar{?} & \bar{?}
 \end{array} \\
 \\
 U(I) = \begin{array}{l}
 \bar{?} \\
 \bar{?} \\
 \bar{?} \\
 \bar{?}
 \end{array}
 \end{array}$$

Thus, $U(I) \bar{?} U(J)$. From this we can conclude that $\Delta_{I, I'} \Delta_{\bar{I}, \bar{I}'} = \Delta_{I, I'} \Delta_{\bar{I}, \bar{I}'}$.

It follows from Theorem 1.1, Theorem 1.2 and Theorem 2.1 that the following three problems are in fact equivalent:

- Characterizing totally nonnegative polynomials of the form $\Delta_{I, I'} \Delta_{\bar{I}, \bar{I}'} - \Delta_{I, I'} \Delta_{\bar{I}, \bar{I}'}$.
- Counting path families in planar networks obeying the rule (I, \bar{I}) versus the rule (J, \bar{J}) .
- Comparing subsets of T_n corresponding to (I, \bar{I}) versus (J, \bar{J}) .

From this equivalence one arrives at the following logical question.

Question 2.1. Is there an easy way of naming all of the special subsets created by the (I, \bar{I}) coloring rule?

In Section 3 we examine a way to represent the special subsets of T_n as graphs, and in Section 4 arrive at our main theorem that establishes a direct relation between these graphs and their corresponding special subsets.

3. The Left-Right Algorithm

Given a dot diagram D it can be a difficult and often tedious task to find all of the elements in T_n that are compatible with D . There is a simple way, however, to find two of them. Furthermore, these two elements can be combined in such a way that they yield a graph G that is useful in determining which of the elements in T_n are compatible with D .

As in Figure 1.6, it is possible to draw D as a circle diagram, which we refer to as simply the $2n$ -gon. As such, D is a null graph of n black vertices and n white vertices. From this observation, we define the *Left-Right algorithm* as follows:

Step 1. Moving along the vertices of the $2n$ -gon counterclockwise, for all white vertices v_i , connect v_i to v_{i+1} if v_{i+1} is a black vertex. Continue with this procedure, ignoring the edges already formed, until all black and white vertices are used. Label this diagram x .

Step 2. Moving along the vertices of the $2n$ -gon clockwise, for all white vertices v_i , connect v_i to v_{i+1} if v_{i+1} is a black vertex. Continue with this procedure, ignoring the edges already formed, until all black and white vertices are used. Label this diagram y .

Step 3. Superimpose the diagrams x and y to obtain the graph G .

Example 3.1. Let D be the dot diagram in Figure 2.1. We represent ‘superimpose x and y ’ as $x \ ? \ y$.

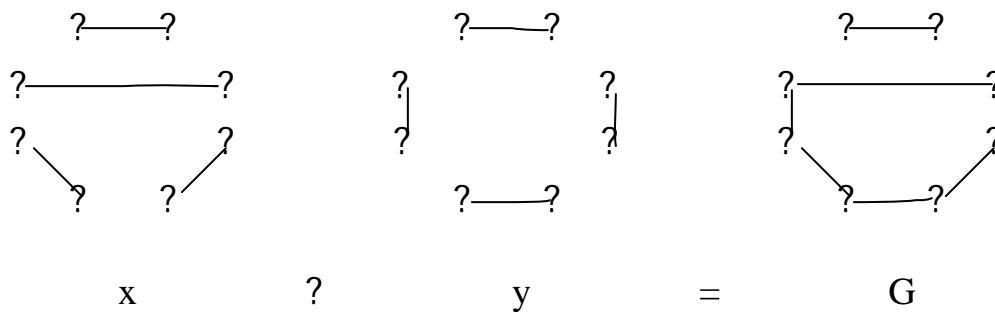


Figure 3.1. The Left-Right algorithm

For all future results, assume that D is a diagram of the $2n$ vertices (v_1, \dots, v_{2n}) of a regular $2n$ -gon, n colored black and n colored white. Consider the $2n$ -gon to be rotated such that two vertices are on top, and label the vertex on the left v_1 . Label the other vertices counterclockwise. In the following results, consider all arithmetic on these subscripts to be done modulo $2n$. Let G be the graph whose vertex set is D and whose $2n$ edges are created by applying the Left-Right algorithm to D . (For each component with two vertices and one edge, consider that edge to be double.)

Within G , we define the *degree of a vertex* to be the number of edges incident upon that vertex. For a given vertex v_i in G we denote this by $\mathbf{deg} \ v_i$. In addition, suppose that $P = (v_i, \dots, v_k)$ is a path in G . Then we say that P is *odd* if it consists of an odd number of vertices and *even* if it consists of an even number of vertices.

Observation 3.1. *Let G be the graph resulting from applying the Left-Right algorithm to a dot diagram D on vertices v_1, \dots, v_{2n} . Suppose (v_i, \dots, v_k) is an odd path in G , where $k = (2n - 1)$. Then G cannot contain an odd number of these paths.*

Proof: We use proof by contradiction. Suppose that G contains p odd paths, where p is an odd integer, and q even paths, where q is an integer. Recall that single vertices count as odd paths. Let $\{a_1, \dots, a_p\}$ be the set of the number of vertices in each odd path, and let $\{\beta_1, \dots, \beta_q\}$ be the set of the number of vertices in each even path. Then

$$\sum_{i=1}^p a_i + \sum_{j=1}^q \beta_j = 2n$$

must hold, since all vertices are used. However, since p is odd and each element a_i has an odd number of vertices, we see that the first sum is odd. The second sum must be even, because each path has an even number of vertices. Therefore, the total sum is odd, and we have a contradiction.

?

Lemma 3.2. *Let G be the graph resulting from applying the Left-Right algorithm to a dot diagram D on vertices v_1, \dots, v_{2n} . If the consecutive vertices v_i, \dots, v_k alternate in color, then (v_i, \dots, v_k) is a path in G .*

Proof: Let v_i, \dots, v_k be an alternating sequence of consecutive white and black vertices not forming a path in G . Suppose that for some j , $i < j = k$, v_{j-1} and v_j are not connected. If v_j is black, then v_{j-1} must be white and the Left-Right algorithm will connect them; thus, a contradiction. If v_j is white, then v_{j-1} must be black and the Left-Right algorithm will connect them; thus, another contradiction. Therefore, v_{j-1} and v_j must be connected, and (v_i, \dots, v_k) forms a path in G .

?

Corollary 3.3. *Let G be the graph resulting from applying the Left-Right algorithm to a dot diagram D on vertices v_1, \dots, v_{2n} . Then for a given vertex v_i in G , $\deg v_i = 2$.*

Proof: This follows from the fact that the Left-Right algorithm connects every white vertex to two black vertices and every black vertex to two white vertices. Thus, for a vertex v_i in G , $\deg v_i = 2$. In the case of a single boundary edge (v_i, v_{i+1}) , we define this edge to be double, and the claim still holds.

?

Lemma 3.4. *Let G be the graph resulting from applying the Left-Right algorithm to a dot diagram D on vertices v_1, \dots, v_{2n} . Suppose the vertices v_i, \dots, v_{i+2k-1} alternate in color. If v_{i-1} and v_i are colored equally and v_{i+2k-1} and v_{i+2k} are colored equally, then v_{i-1} and v_{i+2k} are adjacent in G .*

Proof: By assumption (v_i, \dots, v_{i+2k-1}) is an even path in G and we see that v_i and v_{i+2k-1} must be of opposite color. This implies that v_{i-1} and v_{i+2k} are opposite color as well, and can be connected by the Left-Right algorithm. To see that they are, suppose that v_{i-1} and v_i are white vertices and that v_{i+2k-1} and v_{i+2k} are black vertices. Also, suppose that i is an odd integer. Because the sequence alternates in color, we see that all the elements with odd index are white and all the elements with even index are black. Thus, in the right diagram x of the Left-Right algorithm, v_i connects with v_{i+1} , v_{i+2} connects with v_{i+3} , and so on until v_{i+2k-2} connects with v_{i+2k-1} , completing the first iteration. In the second iteration, all two-element paths are ignored. This will eliminate the paths mentioned above, and only v_{i-1} and v_{i+2k} will remain. Since v_{i+2k} is the opposite color of v_{i-1} and lies to the right, v_{i-1} and v_{i+2k} will be connected. The same argument must hold if i is even or the colors are reversed, only the diagram will be the left diagram y .

?

Within G , every component takes on the appearance of a polygon. We define the *boundary* of a polygon to be the part of a polygon made from non-boundary edges and edges of the form (v_i, v_{i+1}) . To distinguish between the two possible types of polygons, we define an *empty polygon* as a component of G with no edges intersecting its boundary. See Figure 3.2. All other polygons are referred to as *filled polygons*.

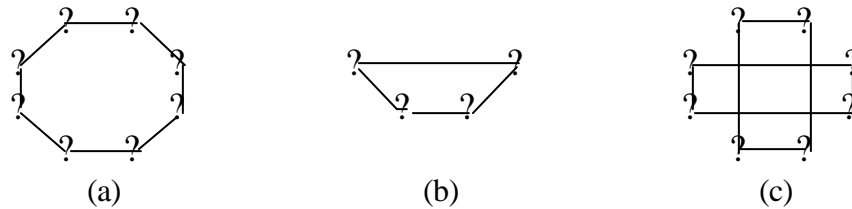


Figure 3.2. Empty and filled polygons found in graphs of T_4

Here, both (a) and (b) are empty polygons, while (c) is a filled polygon constructed from two intersecting empty polygons.

Proposition 3.5. *Let G be the graph resulting from applying the Left-Right algorithm to a dot diagram D on vertices v_1, \dots, v_{2n} . Then G is a collection of polygons, each having an even number of vertices and no two sharing a vertex.*

Proof: We use proof by contradiction. Assume that G is a collection of polygons, at least two having an odd number of vertices. Then it is possible to form a polygon P with k sides, where k is an odd integer, and a polygon Q with m sides, where m is an odd integer. Since P is a polygon, by Lemma 3.2 its k vertices must form an alternating sequence of vertices (v_i, \dots, v_j) in G , where v_i is connected to v_j . We label this alternating sequence as C . By assumption, C contains an odd number of vertices, and this implies that the endpoints of C , v_i and v_j , are the same color. Because the Left-Right algorithm only connects vertices of opposite color, it is impossible to connect v_i and v_j to form P , and we have a contradiction. Therefore, each polygon in G must have an even number of vertices.

Now, suppose that P and Q share a vertex v_p . Then we see that v_p has at least degree three. From Corollary 3.3 we know that a vertex can have at most degree two. Thus, we have a contradiction, and P and Q cannot share a vertex.

?

From Lemma 3.2 and Proposition 3.5 it follows that if (v_i, \dots, v_j) is an alternating sequence of vertices in D , then $\{v_i, \dots, v_j\}$ belong to the same component in G . The converse of this must also be true.

Moving on, we say that a graph G is *valid* if the following two conditions are satisfied:

1. G contains an equal number of black and white vertices
2. G can be created by the Left-Right algorithm.

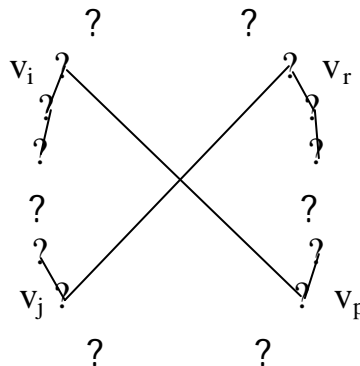
These same requirements must be met for a subgraph H of G to be valid. A result by Zac Pavlov [12, Theorem 3.2] discovered this summer in conjunction with this research paper proves the following proposition.

Proposition 3.6. *Let G be the graph resulting from applying the Left-Right algorithm to a dot diagram D on vertices v_1, \dots, v_{2n} . Then any component in G can be removed such that the resulting subgraph H and the removed component are valid.*

The above proofs supply the majority of the tools that will be needed for proving the remaining results of the paper. The rest of this section characterizes some of the properties that G must have. For further results, see the summer research papers by Mike Ostrowski [11] and Zac Pavlov [12].

Proposition 3.7. *Let G be the graph resulting from applying the Left-Right algorithm to a dot diagram D on vertices v_1, \dots, v_{2n} . Then every cycle in G is a convex polygon.*

Proof: Assume that G is a collection of cycles, with at least one cycle that is not a convex polygon. Label this cycle C , and observe that by Proposition 3.3 C can be described by the two paths $P = (v_i, \dots, v_j)$ and $Q = (v_p, \dots, v_r)$, labeled counterclockwise and located on opposite sides of G . See the figure below. (The color of the vertices is not illustrated.)



Since C is not convex, it follows from Corollary 3.3 that v_i connects to v_p and v_j connects to v_r , else some of the vertices in P or Q would have degree three. Thus, because G is assumed to be valid, C must be valid as well. To show that C cannot exist, it suffices to show that the Left-Right algorithm does not connect C in such a way that C is not convex. By Proposition 3.5 both P and Q must be odd paths or even paths. Thus, we have two cases.

Assume that P and Q are even paths within C . Assume that v_i is a black vertex. (The proof is equally valid if v_i is a white vertex.) Then v_r must also be black, and v_j and v_p must be white vertices. By Proposition 3.6 we may remove the components between P and Q , i.e. the cycles consisting of the vertices v_{j+1}, \dots, v_{p-1} and v_{r+1}, \dots, v_{i-1} , and only the paths P and Q remain. However, by Lemma 3.4, v_i connects to v_j and v_p connects to v_r , yielding a contradiction.

Assume that P and Q are odd paths within C . Assume that v_i is a white vertex. Then v_j is also white, and v_p and v_r must be black vertices. By Proposition 3.6 we may again remove the components between P and Q , and only the paths P and Q remain. However, by Lemma 3.2 v_i connects to v_r and v_j connects to v_p , and we have a contradiction. Therefore, C cannot exist, and every cycle in G must be convex.

?

Proposition 3.8. *Let G be the graph resulting from applying the Left-Right algorithm to a dot diagram D on vertices v_1, \dots, v_{2n} . Then for each edge in G , the indices of the two endpoints differ by an odd number.*

Proof: We use proof by contradiction. Assume that for each edge in G the indices of the two endpoints differ by an even number. Suppose that (v_i, v_k) is one such edge. Since v_i and v_k are adjacent we see that they must be of opposite color. Also, because the indices of v_i and v_k differ by an even number, an odd number of vertices must lie between them. If these odd vertices form a single path, we have a contradiction, since this implies that v_i and v_k are the same color. Thus, a combination of even paths and single vertices must lie between v_i and v_k . Suppose v_j is such a single vertex. By Lemma 3.4 we know that v_j would have connected to either v_i or v_k , depending on the color of v_i and v_k , and we would have a contradiction, since v_i and v_k would not be adjacent. Therefore, the indices of v_i and v_k must differ by an odd number.

?

Proposition 3.9. *If X and Y are two components of G which do not cross, then the first vertex of X and the last vertex of Y must have the same color.*

Proof: To begin, the first vertex of X is found by moving counterclockwise from the vertices of Y . Let v_i be the first vertex of X and v_j be the last vertex of Y . If no other components lie between X and Y , then the proof is complete; we immediately see that v_i and v_j must be the same color, else the Left-Right algorithm would have connected X and Y to form a single, larger component.

Thus, we use proof by contradiction, and assume that v_i and v_j are of opposite color and that G is still valid. Then we see that components must lie between X and Y , else X and Y would be connected. By Proposition 3.6 we may remove the components between X and Y , and only X and Y remain as two distinct components of H , the subgraph of G . However, because v_i and v_j are of opposite color and no components lie between them, by Lemma 3.2 we see that X and Y would be connected. Thus, H is not valid and we have a contradiction. Therefore, v_i and v_j must be the same color.

?

4. Set Partitions of $[2n]$

Suppose that G_1 and G_2 are two graphs created by the Left-Right algorithm. Then we say that the set partition corresponding to G_1 *refines* the set partition corresponding to G_2 if for each component X of G_1 , the vertices of X are all contained within a single component of G_2 . See Figure 4.1.

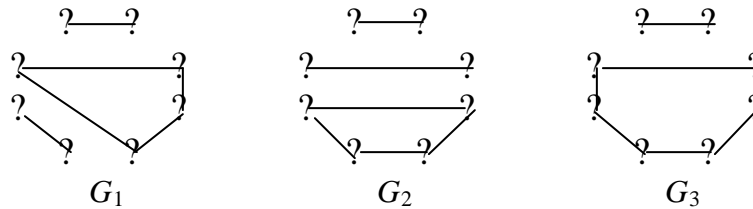


Figure 4.1. Graphs of T_4

Here, both G_1 and G_2 refine G_3 , while G_1 does not refine G_2 and vice versa.

To prove the main result of the paper, we utilize a result obtained this summer by Zac Pavlov [12, Lemma 4.1]. The result assumes all of the definitions from Section 3.

Proposition 4.1. *Let X and Y be two distinct components of G . Then no element $?$ in T_n which is compatible with D contains an edge connecting a vertex from X to a vertex of Y .*

This result will be used to prove our main theorem, as well as the following result.

Proposition 4.2. *Let Q be a component of G with $2m$ vertices. Then it is always possible to obtain m non-crossing edges from the vertices of Q such that the m edges refine Q and form an element of T_m .*

Proof: Let k be the number of edges that can be obtained from Q . It is clear that $k \leq m$, because Q only contains $2m$ vertices, and each edge connects two of these vertices. Thus, we see that $k = m$. Because Q has $2m$ vertices, it is always possible that $k = m$. Based on the claim, we need only show that if $k = m$, then it is always possible that the m edges do not cross. Recall from Proposition 3.5 that within Q , m vertices are black and m vertices

are white. Thus, the $2m$ vertices of Q are identical to a dot diagram for a special subset of T_m . Because the smallest special subset of T_m has a cardinality of one, it is always possible to obtain at least one combination of m non-crossing edges from the vertices of Q .

?

Let f be the map which associates a special subset of T_n to a dot diagram D ,

$$f(D) = \{\tau \in T_n \mid \tau \text{ is compatible with } D\}.$$

Let D_1 and D_2 be two dot diagrams, and let G_1 and G_2 be the two corresponding graphs created by the Left-Right algorithm.

Theorem 4.3. *Let τ be an element of the n th Temperley-Lieb algebra. Let D be a dot diagram, and assume that all elements in $f(D)$ are drawn as circle diagrams. Then $\tau \in f(D)$ if and only if τ refines G .*

Proof: We begin with the if statement and use proof by contradiction. Let $\tau \in f(D)$, and assume that τ does not refine G . By definition, each edge in τ is a double edge, connecting only a black and a white vertex. Thus, τ itself is a graph whose components consist of exactly two vertices and an edge. Because τ does not refine G , the vertices of at least one edge in τ must lie in two separate components within G . Thus, we see that if τ were superimposed onto G , one or more of the edges in τ would connect two distinct components of G . Recall from Proposition 4.1 that no element of T_n which is compatible with D contains such an edge. This implies that τ is not compatible with D , and we have a contradiction. Therefore, τ refines G .

We now consider the only-if statement and again use proof by contradiction. Assume $\tau \notin f(D)$. By our initial assumption, τ is an element of T_n that refines G . Suppose X is a component of G that consists of exactly two vertices and an edge. Then it is clear that τ must also contain this component and all others consisting of two vertices, else τ would not refine G . Thus, we see that all edges in τ not contained within a single edge such as X must be contained within components of G consisting of four or more vertices. Assume that these components consist of a total of $2m$ vertices, where $m = n$. By Proposition 3.5 m are black and m are white. By Proposition 4.2 we know that from these $2m$ vertices m non-crossing edges can be obtained. Thus, it is always possible that τ connects n black vertices to n white vertices without any crossings, with m of the edges from components consisting of four or more vertices and $(n - m)$ of the edges from components consisting of only two vertices. Therefore, by definition $\tau \in f(D)$, and we have a contradiction.

?

Corollary 4.4. *If a component X of G_1 consists of exactly two vertices and a (double) edge, then every Temperley-Lieb element $\tau \in f(D_1)$ contains that edge.*

Proof: Suppose τ is an element of T_n . From Theorem 4.3 we know that $\tau \in f(D_1)$ if and only if τ refines G_1 . Thus, we see that if a component X of G_1 consists of exactly two

vertices and an edge, then every element in $f(D_1)$ must also contain that edge, else that element would not refine G_1 .

?

An example of this surprising result can be seen by examining the graph G created by the Left-Right algorithm in Figure 3.1 and the corresponding special subset of D in Figure 2.2.

We can now establish the relationship between the graph G created by the Left-Right algorithm and totally nonnegative matrices.

Proposition 4.5. *$f(D_1)$ is contained in $f(D_2)$ if and only if the set partition corresponding to G_1 refines the set partition corresponding to G_2 .*

Proof: We begin with the if statement and use proof by contradiction. By assumption, $f(D_1)$ is contained in $f(D_2)$, and we assume that G_1 does not refine the set partition corresponding to G_2 . From Theorem 4.3 we know that every element in $f(D_1)$ refines G_1 , and every element in $f(D_2)$ refines G_2 . Also, by assumption every element in $f(D_1)$ refines G_2 . However, because G_1 does not refine G_2 , we see that there is at least one element in $f(D_1)$ that does not refine G_2 , and we have a contradiction. Therefore, G_1 must refine the set partition corresponding to G_2 .

We now consider the only-if statement, and again use proof by contradiction. By assumption, G_1 refines the set partition corresponding to G_2 , and we assume that $f(D_1)$ is contained in $f(D_2)$ except for a single element $?$. By Theorem 4.3, every element in $f(D_1)$ refines G_1 , and every element in $f(D_2)$ refines G_2 . Since $? \in f(D_1)$, we see that $?$ refines G_1 , and by assumption, G_2 as well. This implies that $? \in f(D_2)$, and we have a contradiction. Therefore, if G_1 refines the set partition corresponding to G_2 , then $f(D_1)$ must be contained in $f(D_2)$.

?

By definition, the special subsets $f(D_1)$ and $f(D_2)$ are equal to the subsets $U(I)$ and $U(J)$, where D_1 corresponds to (I, \bar{I}) and D_2 corresponds to (J, \bar{J}) . Thus, we have the following result.

Corollary 4.6. *$\Delta_{I, I'} \Delta_{\bar{I}, \bar{I}'} = \Delta_{J, J'} \Delta_{\bar{J}, \bar{J}'}$ if and only if the set partition corresponding to G_1 refines the set partition corresponding to G_2 .*

Proof: By Theorem 2.1 we know that $\Delta_{I, I'} \Delta_{\bar{I}, \bar{I}'} = \Delta_{J, J'} \Delta_{\bar{J}, \bar{J}'}$ if and only if $f(D_1)$ is contained in $f(D_2)$. By Proposition 4.5 this is true if and only if the set partition corresponding to G_1 refines the set partition corresponding to G_2 . From this the claim follows.

?

5. The Poset of Special Subsets

A poset is said to be a *lattice* if for any pair of elements x and y , there is a unique least upper bound called ‘ x join y ’ written as $x \vee y$, and a unique greatest lower bound called ‘ x meet y ’ written as $x \wedge y$. Thus, a lattice is apparently an algebra with operations \vee and \wedge . In addition, a poset is said to be a *join-semilattice* if only the join operation is defined and a *meet-semilattice* if only the meet operation is defined.

Let P be a poset. We consider the elements of P to be the graphs G corresponding to all possible dot diagrams, ordered by refinement. Each graph G in P may be viewed as a set partition of $[2n] = \{1, \dots, 2n\}$. That is, the indices of the vertices of each connected component of G form a block of the partition $[2n]$. For example, suppose G is the graph in Figure 5.2.

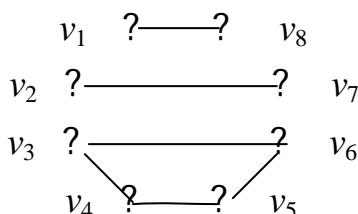


Figure 5.2. A graph G with vertices labeled counterclockwise

If we label the vertices counterclockwise as $\{v_1, \dots, v_8\}$, we see that the components of G are the cycles $\{v_1, v_8\}$, $\{v_2, v_7\}$ and $\{v_3, v_4, v_5, v_6\}$. Thus, G can be viewed as the set-partition

$$[8] = \{1, 8\} \cup \{2, 7\} \cup \{3, 4, 5, 6\}.$$

This can also be written as

$$[8] = 18 / 27 / 3456$$

if we interchange the union symbol (\cup) for the division symbol ($/$), which we refer to as a *partition divider*. Suppose that G_1 and G_2 are two graphs in P , with corresponding set partitions

$$\begin{aligned} s_1 &= a_1 / \dots / a_m \\ s_2 &= \beta_1 / \dots / \beta_n, \end{aligned}$$

where a_i and β_j are cycles in s_1 and s_2 , respectively. We say that s_1 *refines* s_2 if for each cycle a_i in s_1 , the vertices in a_i are contained within a single cycle β_j in s_2 . From this, we define the Join Algorithm.

The Join Algorithm

Let G_1 and G_2 be two graphs in P created by the Left-Right algorithm. Let s_1 and s_2 be the set partitions corresponding to G_1 and G_2 , where

$$\begin{aligned} s_1 &= a_1 / \dots / a_m \\ s_2 &= \beta_1 / \dots / \beta_n. \end{aligned}$$

At step i:

Let $(v_{j_1}, \dots, v_{j_r})$ be the vertices of cycle a_i . For each edge (w_1, w_2) in a_i , **do:**

- (i) **If** (w_1, w_2) refines a single cycle β_j in s_2 , **end.**
- (ii) **Else** remove the appropriate partition divider from s_2 such that w_1 and w_2 are included within the same cycle in s_2 .

After all cycles in s_1 have been considered, the resulting set partition s_3 will correspond to the graph G_3 .

Lemma 5.1. *For the cycle a_i in s_1 , the Join Algorithm removes partition dividers from s_2 in such a way that the resulting cycle β_j in s_2 is unique.*

Proof: We use proof by contradiction. Assume that β_j is not unique. Then for some edge (v_p, v_{p+1}) in a_i , where $k = p < r$, there are at least two ways to remove a partition divider from s_2 such that β_j is created. This implies that v_p or v_{p+1} is contained in more than one cycle, i.e. v_p or v_{p+1} is used more than once. By Lemma 3.2 and Proposition 3.5 this is impossible, and we have a contradiction. Therefore, β_j is unique.

?

Conjecture 5.2. *The Join Algorithm produces the graph G_3 such that $G_1 \vee G_2 = G_3$.*

To prove this conjecture, it must not only be shown that G_3 is a least upper bound for G_1 and G_2 , but also that G_3 is a valid graph. Lemma 5.1 would be especially useful for demonstrating the uniqueness of G_3 , because it implies that the Join Algorithm yields the same graph regardless of the order in which the cycles in s_2 are considered.

If Conjecture 5.2 were true, we would have as a consequence that P is a join-semilattice, because a method for finding $G_1 \vee G_2$ exists. In addition, if we were to let P be the poset of special subsets of T_n , ordered by inclusion, (i.e. define $S_1 <_p S_2$ if S_1 is a subset of S_2 .) we would also have as a consequence of Conjecture 5.2 and Theorem 4.3 that P is a join-semilattice. This follows from the fact that these two posets, one of graphs G and one of the corresponding special subsets S , are isomorphic.

6. Open Problems

Problem 6.1. Find a way to describe $f(D_1)$ in terms of the two Temperley-Lieb elements (x, y) created by the Left-Right algorithm applied to D_1 .

Theorem 4.3 illustrates the direct relation between $f(D_1)$ and the graph G_1 created from D_1 by superimposing (x, y) . It should be possible to develop a removal algorithm that extracts all elements of T_n that refine G_1 . Such an algorithm would have to consider empty and filled polygons separately, taking into account the intersecting components that create a filled polygon.

Problem 6.2. Given two graphs G_1 and G_2 constructed using the Left-Right algorithm, find a formula for $G_1 \vee G_2$ in P .

Assuming from Conjecture 5.2 that the Join Algorithm yields a valid least upper bound for G_1 and G_2 , it is still cumbersome to work with. It would be convenient to have a simple formula for $G_1 \vee G_2$ in P , assuming that such a formula is even possible.

Problem 6.3. Given two special subsets S_1 and S_2 of T_n , find a formula for $S_1 \vee S_2$ in P .

Assuming that Conjecture 5.2 is true, P must be a join-semilattice. It would be very useful to have a formula for $S_1 \vee S_2$ in P , particularly one that utilizes the elements in S_1 and S_2 .

Problem 6.4. What is the significance of each maximal chain in P ?

A *maximal chain* is a chain such that another element cannot be added to make it longer. It would be interesting to know if the maximal chains in P have any significance.

Problem 6.5. Find a formula for the number of distinct maximal chains in P .

7. Acknowledgements

The author is grateful to Zac Pavlov, Mike Ostrowski and Nina Palmo for many helpful conversations, and especially to Mark Skandera for encouragement, insight, and a terrific summer research project.

References

- [1] F. Brenti. Combinatorics and total positivity. *J. Combin. Theory Ser. A*, **71** (1995) pp. 175-218.
- [2] C. W. Cryer. Some properties of totally positive matrices. *Lin. Alg. Its Appl.*, **15** (1976) pp. 1-25.
- [3] S. M. Fallat, M. I. Gekhtman, and C. R. Johnson. Multiplicative principal-minor inequalities for totally nonnegative matrices, 2001. Preprint.
- [4] S. Fomin and A. Zelevinsky. Total positivity: Tests and parametrizations. *Math. Intelligencer*, (2001) pp. 23-33.
- [5] I. Gessel and G. Viennot. Binomial determinants, paths, and hook length formulae. *Advances in Mathematics*, **58** (1985) pp. 300-321.
- [6] I. Gessel and G. Viennot. Determinants and plane partitions, 1989. Preprint.

- [7] S. Karlin and G. MacGregor. Coincidence probabilities. *Pacific J. Math.*, **9** (1959) pp. 1141-1164.
- [8] B. Lindström. On the vector representations of induced matroids. *Bull. London Math. Soc.*, **5** (1973) pp. 85-90.
- [9] C. Loewner. On totally positive matrices. *Math. Z.*, **63** (1955) pp. 338-340.
- [10] G. Lusztig. Total positivity in reductive groups. In *Lie Theory and Geometry: in Honor of Bertram Kostant*, vol. 123 of *Progress in Mathematics*. Birkhäuser, Boston, 1994 pp. 531-568.
- [11] M. Ostrowski. The relation of graphs to totally nonnegative matrices and the Temperley-Lieb algebra. . Summer REU Research, The University of Michigan, 2002.
- [12] Z. Pavlov. Geometric findings with relation to the Temperley-Lieb algebra. Summer REU Research, The University of Michigan, 2002.
- [13] B. Reed and Mark Skandera. Total nonnegativity and $(3 + 1)$ -free posets. *J. Combin. Theory Ser. A*. (to appear).
- [14] M. Skandera. Inequalities in products of minors of totally nonnegative matrices. *J. Alg. Combinatorics*. (to appear).
- [15] A. Whitney. A reduction theorem for totally positive matrices. *J. d'Analyse Math.*, **2** (1952) pp. 88-92.

E-mail address: rpistor@umich.edu