ISSN 1855-3966 (printed edn.), ISSN 1855-3974 (electronic edn.)
ARS MATHEMATICA CONTEMPORANEA 15 (2018) 267–295
https://doi.org/10.26493/1855-3974.939.77d
(Also available at http://amc-journal.eu)

Calculating genus polynomials via string operations and matrices

Jonathan L. Gross [*]
Dept. of Computer Science, Columbia University, New York, NY 10027, USA

Imran F. Khan
PUCIT, University of the Punjab, Lahore 54000, Pakistan

Toufik Mansour
Department of Mathematics, University of Haifa, 3498838 Haifa, Israel

Thomas W. Tucker [†]
Dept. of Mathematics, Colgate University, Hamilton, NY 13346, USA

Received 21 September 2015, accepted 27 November 2017, published online 20 June 2018

Abstract

To calculate the genus polynomials for a recursively specifiable sequence of graphs, the set of cellular imbeddings in oriented surfaces for each of the graphs is usually partitioned into *imbedding-types*. The effects of a recursively applied graph operation τ on each imbedding-type are represented by a *production matrix*. When the operation τ amounts to constructing the next member of the sequence by attaching a copy of a fixed graph H to the previous member, Stahl called the resulting sequence of graphs an *H-linear family*. We demonstrate herein how representing the imbedding types by strings and the operation τ by *string operations* enables us to automate the calculation of the production matrices, a task requiring time proportional to the square of the number of imbedding-types.

Keywords: Graph imbedding, genus polynomial, production matrix, transfer matrix method.

Math. Subj. Class.: 05A15, 05A20, 05C10

[*] J. L. Gross is supported by Simons Foundation Grant #315001.
[†] T. W. Tucker is supported by Simons Foundation Grant #317689.

E-mail addresses: gross@cs.columbia.edu (Jonathan L. Gross), imran.farid@pucit.edu.pk (Imran F. Khan), tmansour@univ.haifa.ac.il (Toufik Mansour), ttucker@colgate.edu (Thomas W. Tucker)

1 Introduction

The ***genus polynomial*** of a graph G is defined to be the generating function

$$\Gamma_G(z) = \sum_{i \geq 0} g_i(G) z^i,$$

where $g_i(G)$ counts the cellular imbeddings of G in the closed oriented surface S_i of genus i. Following their introduction by [12] in 1987, and starting with the work of [6], the genus polynomials for a recursively constructed sequence of graphs have most frequently been calculated, as in [8, 9, 13], by partitioning the imbeddings according to the cyclic orderings of occurrences of root-vertices on the face-boundary walks (abbr. ***fb-walks***) of the imbeddings. In this paper, we describe how to expedite such calculations.

1.1 Rotation systems

All our graphs come with a labeling of the edges. All graph imbeddings in this paper are assumed to be cellular, that is, each component of the complement of the imbedded graph is homeomorphic to the interior of the unit disk. All surfaces are assumed to be closed and oriented.

To describe the imbeddings of a graph G, we assign $+$ and $-$ orientations to the edges, including self-loops. Then any imbedding defines, for each vertex, a cyclic order of the signed edge-ends initiating at that vertex, which is called the ***rotation*** at that vertex. The rotations collectively form a ***rotation system*** (e.g., see [19]), which acts as a permutation ρ on the oriented edge set. If λ is the involution that reverses the orientation of each edge, then the face boundary walks of the imbedding are the orbits of the permutation $\rho\lambda$.

A rotation system for a graph has also been called a "ribbon graph" or a "fat graph", especially in the context of algebraic geometry, Riemann surfaces, and the theory of dessins ([3, 20, 24]). We use the Euler polyhedral formula

$$|V| - |E| + |F| \;=\; 2 - 2\gamma(S)$$

to compute the genus $\gamma(S)$ of the imbedding surface S.

Two imbeddings $\iota_1, \iota_2 \colon G \to S$ determine the same rotation system if and only if there is a homeomorphism of the surface S taking $\iota_1(G)$ to $\iota_2(G)$ that acts as the identity isomorphism on the graph G (i.e., respects the labeling of edges). Accordingly, there is a bijection from the set of imbeddings of G to the set of rotation systems.

A problem in calculating genus polynomials is that the number of possible cyclic orderings of the edge-ends incident at a d-valent vertex is $(d-1)!$. Thus, the number of imbeddings of a graph G is the product $\Pi(d_v - 1)!$, taken over all vertices v of G, where d_v is the valence of v. It is well-known [30] that the problem of calculating the minimum genus of a graph is NP-hard, even when the graph is 3-regular. It follows that calculating the genus polynomial is at least that hard. For example, the number of rotation systems for the complete graph K_7 is

$$(5!)^7 \approx 3.6 \times 10^{14},$$

and the genus polynomial for K_7 has only recently been computed [2]. Table 1 gives the list of coefficients.

Table 1: Genus distribution of the complete graph K_7.

i	g_i
0	0
1	240
2	3,396,960
3	3,746,107,320
4	594,836,922,960
5	20,761,712,301,960
6	158,500,382,165,280
7	178,457,399,105,280

1.2 Context

Genus polynomials for recursively specified families of graphs have been computed mostly within a general paradigm in which the recursive operation occurs in the vicinity on a small number of vertices or edges designated as *roots*. The set of all imbeddings of each graph in the family is *partitioned* into what we now call *imbedding-types*, according to incidence of the fb-walks on the roots, a technique for calculating genus polynomials that was introduced by [6]. This basic paradigm is exemplified by [8, 13] for root-vertices, and by [25, 26] for root-edges.

This paper integrates several embellishments of the basic paradigm:

- The genus polynomial for a graph is partitioned into a ***pgd-vector***, with one coordinate for each imbedding type, such that each coordinate is a polynomial that gives the number of oriented imbeddings of that imbedding type in every orientable surface.
- The recursively applied topological operation is represented by a *production system*, as developed by Gross, Khan, and Poshni, in a series of papers [8, 13, 25, 26], that transforms the pgd-vector for a given graph into the pgd-vector for the graph resulting from an application of the recursive operation used to specify the graph family. In those papers, the productions were calculated with the aid of a multiplicity of drawings of rotation projections.
- The representation of production systems by matrices was introduced by Stahl [27], for application to pgd-vectors of some graphs in what he called *H-linear families*. Such matrices are now called ***production matrices***, and the graph sequences are now called *H-linear sequences*, or simply *linear sequences*. Stahl used what he called *permutation-partition pairs* to derive production matrices.
- The representation of *imbedding-types* by strings of root-vertices, as developed by Gross [11].
- Using string operations directly to calculate the production matrices, as suggested subsequently by Mohar [23].

The general idea of a ***linear sequence*** is that a copy of a graph H is attached to each graph in the sequence to form the next graph in the sequence. It is necessary to attach each copy of H in the same way, as described precisely by [4].

1.3 Outline of this paper

Our main focus in this paper is the calculation of production matrices. Since the size of the matrix increases with the number of imbedding types, and since the number of imbedding-types grows exponentially with the number of roots and with the valences of the roots, most of the calculations of genus polynomials have been for sequences of graphs with at most two roots and valences no larger than 4.

The string notation by which we concisely represent imbedding types allows us to automate the bookkeeping used in tracking the way imbedding types are changed by the addition of paths between root vertices. The advantages of this system are many. It allows us to derive in a few lines (see Subsection 4.3) the computation of production matrices that formerly involved many figures [10] or detailed paper-and-pencil applications of what Stahl [28] called the "Walkup reduction" for permutation-partition pairs. String notation facilitates the computer calculation of production matrices whose derivation would be unfeasibly tedious by hand (see the 12×12 matrix in Section 5). Finally, it reveals ways of combining different imbedding types to get smaller matrices (see Subsection 5.1).

Following a review in Section 2 of the representation of imbedding-types by strings, Section 3 introduces the representation of topological and vertex-labeling operations on imbeddings by string operations. Section 3 also introduces the concept of grouping two or more i-types into a "super-type". As an illustration of how the string operations are used in calculations of genus distributions, Section 4 applies these representations to two previously published examples, one of which (the iterated claw) we have adapted here to give a detailed example of grouping. Also, we explain in Section 4 how our use of productions to calculate pgd-vectors is interpretable as an embellishment of the *transfer matrix method*, along the lines described by [29].

Section 5 explores issues related to computation. It uses the theory developed in the previous sections to calculate genus polynomials for a vertex-amalgamation path of copies of K_4 and for an edge-amalgamated path of copies of K_4. Without string operations, both derivations would be long and tedious. We used two computational aids while preparing this paper.

- The computational system Maple®.
- A computer program, based on string operations, that calculates production matrices.

Such kinds of aids are what we have in mind in various comments here, rather than a state-of-the-art computer. Section 5 includes an additional example of the grouping of i-types into a super-type.

In Section 6, we use Burnside's Lemma to derive a formula for the maximum number of imbedding types for a graph with two roots of any possible combination of valences. We generalize the formula to more that two roots. From the rapid growth rate of the number of imbedding-types, as valences and the number of roots of the graphs at issue increases, it becomes clear that programmable computation tools are a virtual necessity when seeking to calculate genus polynomials.

2 Representing imbedding-types by strings

In this section, we develop a system of notation that uses strings of root-labels, so that representing the addition of an edge to a graph becomes a simple matter of applying a few string-processing rules.

2.1 Face-boundary-walks

We assign labels $0, 1, 2, \ldots$ to the roots of a graph G. Given an imbedding of G, we represent a face as a string of roots, in the order they are encountered in a traversal of its fb-walk following the orientation of the surface. If an fb-walk does not contain any roots, we call its string *empty*. Two strings are ***equivalent representations of an fb-walk*** if one is a cyclic shift of the other. We denote an entire equivalence class of strings by putting a representative string of labels inside parentheses. The ***canonical*** representative for the equivalence class of fb-walks is the one with minimum lexicographic order with respect to the labels $0, 1, \ldots$.

Remark 2.1. Vertices that are not roots do not appear in the string representing a face. Accordingly, the appearance of consecutive labels $\ldots 12 \ldots$ within a string would not imply that there is an edge between vertices 1 and 2. Also, since any labeled vertex may appear more than once around an fb-walk, the corresponding cyclic list of root-labels is not a permutation.

2.2 Imbedding types

The collection of non-empty strings for all the fb-walks of an oriented imbedding of a rooted graph G is called an ***imbedding-type*** of G (abbr. ***i-type***). The collection of all imbedding types over all imbeddings of G is called the ***full*** collection of imbedding types for G.

In order to compare imbedding types for the same rooted graph, we usually use the shortlex order [31] on canonical representatives to make a list of fb-walks (rather than a set): shorter faces are listed before longer ones, and if two faces have the same length, the one with shortlexically least canonical representative is listed first. We call such a list the ***canonical form*** for the i-type.

Example 2.1. Figure 1 shows an imbedding of K_4 in the sphere with roots $0, 1, 2,$ and 3. If the "interior" fb-walks are oriented counterclockwise (which forces the "exterior" fb-walk to appear as clockwise, from the perspective of vertex 0), then the i-type (in canonical form) is

$$(012)(023)(031)(132).$$

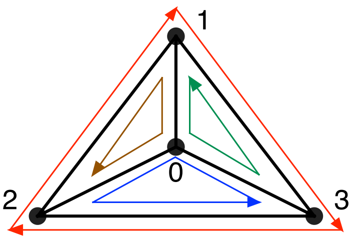

Figure 1: An imbedding of K_4 in S_0.

Notice that each face is represented by its canonical form (cyclic shift with least lexicographic order) and that the faces are listed in shortlex order. Since for this example every vertex is a root, it follows that two consecutive vertices (with respect to cyclic order) in the representation of a face actually does represent a directed edge. For any two roots i and j, the directed edge ij appears exactly once. If $i = 0$ and $j = 1$, we could suppress the labels $2, 3$ to obtain the i-type

$$(01)(0)(01)(1) = (0)(1)(01)(01)$$

for the imbedding of Figure 1. If the only root is 0, then the imbedding type would be $(0)(0)(0)$. Notice in the last imbedding type, the number of strings is less than the number of faces, because the fb-walk (132) contains no instances of vertex 0, and we do not list empty faces. If we reverse the orientation of the sphere and have all four vertices $0, 1, 2, 3$ as roots, then the i-type in canonical form would be

$$(021)(032)(013)(123) = (013)(021)(032)(123).$$

Observe that the shortlex order for the faces differs from the previous orientation. However, the i-type for roots $0, 1$ is the same as before, as is the i-type for root 0, when labels $1, 2,$ and 3 are suppressed.

Example 2.2. Considering all 2^4 rotation systems for K_4, we get the following census of i-types for roots $0, 1$, given in shortlex order:

- 2 of i-type $(0)(1)(01)(01)$
- 2 of i-type $(0)(01011)$
- 2 of i-type $(1)(00101)$
- 2 of i-type $(01)(0011)$
- 8 of i-type $(01)(0101)$

Notice that since there is only one edge 01, only one of the substrings 01 in an i-type, for example $(01)(0101)$, comes from an edge. The other juxtapositions of 0 and 1 come from suppressing incidences of the roots 2 and 3. We conclude that

$$\{(0)(1)(01)(01), (0)(01011), (1)(00101), (01)(0011), (01)(0101)\}$$

is a full set of i-types for K_4 with roots 0 and 1. In Section 6 of this paper, we shall see that the maximum number of i-types for a pair of 3-valent roots is 38.

Remark 2.2. We observe that within the string representation of any i-type, each root-vertex appears as many times as its valence. If there is an edge between roots i and j, then both ij and ji must appear at least once in every i-type. On the other hand, as we have noted, the appearance of ij in a string does not imply that there is an edge between i and j.

Remark 2.3. Suppose that G has no multi-edges or self-loops, and suppose that every vertex is a root. Then each rotation system for the graph G determines a unique i-type, since each i-type determines a rotation system for the dual graph. In this circumstance, the number of i-types would be the same as the number of rotation systems. At the opposite extreme, the set of imbeddings for a tree with one root-vertex has only one i-type.

Remark 2.4. When there are multi-edges or loops and every vertex is a root, it happens that different rotation systems can determine the same i-type. For example, the bouquet B_n has only one vertex 0 and has n loops at that vertex. Then an i-type is simply a partition of $2n$ into k parts, where k is the opposite parity of n (k is the number of faces, so the Euler characteristic $1 - n + k$ must be even). Thus, the number of i-types for imbeddings of B_n with k faces is at most the Stirling subset number $\left\{ {2n \atop k} \right\}$ (i.e., the Stirling number of the second kind), where k and n have opposite parities.

2.3 String notational conventions

We adopt two notational conventions for strings:

- The concatenation of a string S with a string T is denoted by ST.
- The reverse string for a string S is denoted by S^{-1}.

We emphasize that SS^{-1} is not the empty string, but rather the concatenation of S with its reverse (which forms a palindrome). This notation does satisfy the relations

$$\begin{aligned}(ST)^{-1} &= T^{-1}S^{-1} \quad \text{and} \\ (S^{-1})^{-1} &= S\end{aligned}$$

as if in a group, even though our strings are not permutations (since roots can repeat), and even though they do not form a group.

2.4 Pgd-vectors

Given an i-type t, we write its ***partial genus polynomial*** in the form

$$\sum a_i z^i$$

where a_i is the number of type-t imbeddings of G of genus i.

If the i-types are listed in shortlex order, then we can associate the set of partitioned genus polynomials for G with a column vector whose r^{th} coordinate is the partial genus polynomial for the r^{th} i-type. This is called a ***pgd-vector*** for the graph G. For instance, the partitioned genus distribution for the complete graph K_4 given by Example 2.2 corresponds to the vector

$$\begin{bmatrix} 2 & 2z & 2z & 2z & 8z \end{bmatrix}^T$$

where the superscript T denotes the transpose.

3 Operations on imbedding-types

In this section, we describe how a path-adding operation affects the i-types. We also describe the relabeling of root-vertices, and the suppression of some root-labels, which are used, for instance, when there are no more paths to be added at a root-vertex.

3.1 Adding a path within a face and between faces

Let G be a rooted graph and let iUj be a path whose endpoints i, j are roots of G but all other vertices of U are not in G. If U is empty, we have simply the edge ij. The effect of adding iUj into a face with fb-walk $(iSjT)$ is given by the following operation:

$$(iSjT) + iUj \;\to\; (iSjU^{-1})(iUjT). \tag{3.1}$$

In calculations, we may denote the right-hand side by $\mathrm{Add}_{iUj}[iSjT]$. If the i-type in which the fb-walk $(iSjT)$ occurs is of the form

$$(iSjT)W_1W_2\ldots W_k,$$

which includes other fb-walks, then applying Operation (3.1) to that i-type yields the i-type

$$(iSjU^{-1})(iUjT)W_1W_2\dots W_k.$$

That is, the other fb-walks of the i-type are simply recopied.

The effect of adding the path iUj between two faces (iS) and (jT) is given by this operation:

$$[(iS),(jT)] + iUj \;\to\; z(iSiUjTjU^{-1}). \tag{3.2}$$

The right-hand side may be expressed as $\mathrm{Add}_{iUj}[(iS),(jT)]$. When applying Operation (3.2) to an i-type with fb-walks (iS) and (jT), any other fb-walks of the i-type are simply recopied, the same as for Operation (3.1). The multiplier z indicates that the genus of the imbedding rises by 1 when a handle is added to the surface.

For the circumstance in which the faces (iS) and (jT) lie within (disjoint) imbeddings ι and ι' of separate graphs G and G', the effect of joining the imbeddings by adding the path iUj between the two faces (iS) and (jT) is given by this operation:

$$[(iS),(jT)] + iUj \;\to\; (iSiUjTjU^{-1}). \tag{3.3}$$

The non-presence of the multiplier z signifies the fact that the genus of the surface in which the resulting graph is imbedded is simply the sum of the genera of the imbeddings ι and ι'.

Example 3.1. Consider an imbedding of the 4-cycle 0213 in the sphere. There are two faces, one with fb-walk (0213) and the other with fb-walk (0312). Thus, the initial i-type is (0213)(0312). There are four ways to add a path 0451 to such an imbedding, one within the face (0213), one within the face (0312) and two between the faces (0213) and (0312). Figure 2 shows the four possible ways to add the path $0U1$ and the resulting i-type for each.

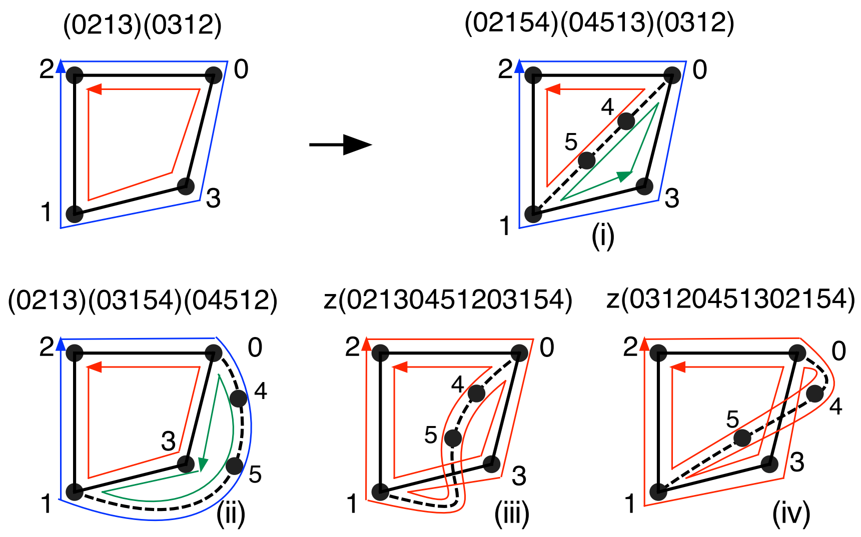

Figure 2: Adding the path 0451 to an imbedding of a 4-cycle in the sphere.

(i) Inserting path 0451 into the face (0213) yields the imbedding type

$$(02154)(04513)(0312),$$

as per Operation (3.1). We now have three faces. Root-vertices 0 and 1 now have valence 3, so they now appear three times in this representation of the i-type.

(ii) Inserting the path 0451 instead into the face (0312) yields i-type

$$(0213)(03154)(04512).$$

(iii) If we join the two faces, from endpoint 0 inside the face (0213), to endpoint 2 inside the face (0312), then the resulting string expression is

$$z(02130451203154).$$

(iv) If we add the path 0451 with edge-end 0 now inside the face (0312) and edge-end 1 inside the face (0213), we get the string expression

$$z(02154031204513).$$

It follows that the net result of adding the path 0451 to the i-type $(0213)(0312)$ is the following linear combination of i-types taken over the ring $\mathbb{Z}[z]$ of polynomials with integer coefficients:

$$(02154)(04513)(0312) + (0213)(03154)(04512) \\ + z(0213045120354) + z(03120451302154).$$

Remark 3.1. The path ii for adding a self-loop is simply a special case. As a variation on Operation (3.1), we have

$$(iS) + ii \rightarrow (i)(iSi)$$

As a variation on Operation (3.2), we have

$$[(iS), (iT)] \rightarrow z(iSiiTi)$$

Remark 3.2. If a graph already has an edge ij, then adding the path $P = ij$ creates a multiple adjacency.

3.2 Suppressing roots and relabeling roots

Given a subset of roots $\{i, j, \dots\}$, the ***root-suppression operator*** $\mathrm{Sup}_{i,j,\dots}$ acts to suppress every occurrence of the root-labels $i, j, \dots$ within an i-type t. For example,

$$\mathrm{Sup}_{1,2}[(1)(12)(0212)(0231303)] = (0)(03303).$$

Observe that we delete empty pairs of parentheses as a final step in suppressing roots.

Example 3.1, continued. Suppressing roots 2 and 3 as well as any roots along U transforms the i-type $(021U^{-1})(0U13)(0312)$ into the i-type $(01)(01)(01)$. Similarly,

$$\mathrm{Sup}_{1,2,U}[z(021U^{-1})(0U13)(0312)] = z(010101).$$

Moreover, when root-suppression is applied to a linear combination of i-types, it can reduce the number of terms. For instance,

$$\begin{aligned}\mathrm{Sup}_{2,3,U}&[(021U^{-1})(0U13)(0312) + (0213)(031U^{-1})(0U12) \\ &\quad + z(02130U1203U^{-1}) + z(03120U13021U^{-1})] \\ &= 2(01)(01)(01) + 2z(010101).\end{aligned}$$

We can also relabel roots, by using the ***root-relabeling operator***. Suppose that the label i appears in i-type t and label j does not. Then $\mathrm{Lab}_{ij}[t]$ is the i-type obtained by replacing in t all occurrences of i by j. Thus,

$$\mathrm{Lab}_{24}[(1)(2)(22)(1323)] = (1)(4)(44)(1343).$$

We denote by $\mathrm{Lab}_{ii',jj',\ldots}[t]$ the result of relabeling i by i', j by j' etc.

3.3 Reversing orientation

If the orientation of a graph imbedding is reversed, the effect on i-types is as follows:

- the cyclic order of each fb-walk is reversed;
- the genus of the imbedding stays the same.

We call this the ***i-type reversal operator***. Given an i-type t, we denote by t^{-1} the i-type in which each fb-walk string is reversed. Note that if (ST) is an fb-walk within i-type t, then the corresponding fb-walk in t^{-1} is $(T^{-1}S^{-1})$, for which a cyclic shift gives $(S^{-1}T^{-1})$. On the other hand, the i-type $(R^{-1}S^{-1}T^{-1})$ is not a cyclic shift of the i-type $(RST)^{-1} = (T^{-1}S^{-1}R^{-1})$.

Proposition 3.3. *The i-type reversal operator commutes with the operators* Add, Sup, *and* Lab.

Proof. Clearly, we can reverse lists either before of after suppressing or relabeling vertices, and the result is the same. Using Rule (3.1) for adding a path within a face, we have

$$\mathrm{Add}_P[(iSjT)]^{-1} = [(SP^{-1})(PT)] = (T^{-1}P^{-1})(S^{-1}P) \quad \text{and} \tag{3.4}$$

$$\mathrm{Add}_P[(iSjT)^{-1}] = \mathrm{Add}_P[iT^{-1}jS^{-1}] = (T^{-1}P^{-1})(S^{-1}P) \tag{3.5}$$

Using Rule (3.2) for adding an edge between two faces, we have

$$\mathrm{Add}_P[(iS),(jT)]^{-1} = z(PTP^{-1}S)^{-1} = z(S^{-1}PT^{-1}P^{-1}) \quad \text{and} \tag{3.6}$$

$$\mathrm{Add}_P[(iS)^{-1},(jT)^{-1}] = \mathrm{Add}_P[(iS^{-1}),(jT^{-1})] = z(PT^{-1}P^{-1}S^{-1}) \tag{3.7}$$

□

3.4 Combining i-types into super-types

As we have observed, the number of i-types grows exponentially with the valence and the number of roots, so any way of reducing the number of i-types is welcome. For example, in building a graph by path-addition, we can always group an i-type with its reverse, since i-type reversal commutes with edge path-adding. Indeed, root-suppression is also a way of grouping many i-types together.

Suppose that the rooted graph H is obtained from the rooted graph G by a sequence Op of the following kinds of operations:

path-additions, root-suppression, and root-relabeling.

Let $\mathcal{T}$ be the full collection of i-types for G, and let $\mathcal{S}$ be the full collection of i-types for H, both in shortlex order. Then for any i-type $t \in \mathcal{T}$, we see that

> The expression $Op(t)$ is a linear combination of elements of $\mathcal{S}$, with coefficients taken from the ring $\mathbb{Z}[z]$ of polynomials in z.

We represent Op, therefore, as a matrix M whose columns are labeled by i-types in $\mathcal{S}$, and whose rows are labeled by i-types in $\mathcal{T}$, where $M_{s,t}$ is the coefficient of i-type s in the expression $Op(t)$.

Let $\mathcal{P}$ and $\mathcal{Q}$ be partitions of $\mathcal{S}$ and $\mathcal{T}$, respectively. Suppose that we order the i-types within $\mathcal{S}$ and the i-types within $\mathcal{T}$ so that the i-types within each cell of $\mathcal{P}$ and within each cell of $\mathcal{Q}$ are contiguous in the respective orderings, inducing a partitioning of the production matrix M into blocks that satisfy this criterion:

> Within each block, the column sums are the same. (This requirement applies also to the blocks that span only a single row of the matrix M, which implies that the entries in such a row are identical.)

Then we call the partitions $\mathcal{P}$ and $\mathcal{Q}$ ***compatible*** with M. Moreover, we call each part of $\mathcal{P}$ and $\mathcal{Q}$ a ***super-type*** for the operation Op. We can then condense the matrix M to a smaller one whose columns are indexed by $\mathcal{P}$ and rows by $\mathcal{Q}$, and whose entries are the constant column sum of the block of M determined by the respective parts.

We have already encountered super-types in two contexts: type-reversal and root-suppression. For type-reversal, we partition a full collection of i-types into parts by grouping together an i-type and its reverse. Since type-reversal commutes with path-adding, root-suppression, and root-relabeling, it is compatible with any sequence of those operations. We can also view root-suppression $\mathrm{Sup}_{i,j,\ldots}$ itself as creating super-types. In this case, we have $\mathcal{S} = \mathcal{T}$. The parts of $\mathcal{P}$ are just singletons; i-types s, t are in the same part of $\mathcal{Q}$ if and only if $\mathrm{Sup}_{i,j,\ldots}(s) = \mathrm{Sup}_{i,j,\ldots}(t)$. Notice in this case, the matrix M is just the identity matrix and each block is a part of a single column of M. The condensed matrix has a single 1 in each column.

Another way to create super-types is to exploit any symmetry between roots. With $H, G, \mathcal{S}, \mathcal{T}$ as before, suppose there is a graph automorphism f of H that permutes the roots of H and G. Then f also induces a permutation of the $\mathcal{S}$ and $\mathcal{T}$. We can then use orbits of that permutation as super-types.

Grouping types into super-types by graph automorphisms and reversal is illustrated particularly well in the family of iterated claws in Subsection 4.3, where 12 i-types are reduced to three super-types. For now we consider an example that provides a clear illustration of the theory underlying the reduction.

Example 3.2. Suppose that $G = K_4$, as in Example 2.2, with roots 0 and 1, and that the graph H is obtained from G by the operation of adding a second edge between 0 and 1. Since there is an automorphism of the graph G interchanging 0 and 1, we have the partition given in Table 2 for the full set $\mathcal{T}$ of i-types of the graph G, under the partition $\mathcal{Q}$ (induced by this automorphism), with the parts of $\mathcal{Q}$ indicated by square brackets.

Table 2: Partitioning the i-types for $(K_4, \{0, 1\})$.

$\mathcal{T}$		$\mathcal{T}/\mathcal{Q}$
(0)(1)(01)(01)		(0)(1)(01)(01)
(0)(01011)		[(0)(01011), (1)(00101)]
(1)(00101)	$\longrightarrow$	(01)(0011)
(01)(0011)		(01)(0101)
(01)(0101)		

We can construct the full set $\mathcal{S}$ of 13 i-types for the graph H, by adding the path 01 to the i-types in $\mathcal{T}$ for the graph G. In Table 3, we again use square brackets to enclose the parts of the partition $\mathcal{P}$.

Table 3: Partitioning the i-types for $(K_4 + 01, \{0, 1\})$.

$\mathcal{S}$		$\mathcal{S}/\mathcal{P}$
(0)(1)(01)(01)(01)(0)(1)(010101)		(0)(1)(01)(01)(01)
(0)(01)(01011)		(0)(1)(010101)
(1)(01)(00101)		[(0)(01)(01011), (1)(01)(00101)]
(0)(011)(0101)		[(0)(011)(0101), (1)(001)(0101)]
(1)(001)(0101)		(01)(01)(0011)
(01)(01)(0011)		(01)(01)(0101)
(01)(01)(0101)	$\longrightarrow$	(01)(001)(011)
(01)(001)(011)		[(00101011), (00110101)]
(00101011)		(00101101)
(00110101)		(01010101)
(00101101)		
(01010101)		

Applying the string operation for adding the edge 01 to K_4, we obtain the matrix M, which maps the pgd-vector for K_4 to the pgd-vector for the graph $K_4 + 01$, as follows:

$$\begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 2z & 0 & 0 & 0 & 0 \\ 2z & 4 & 0 & 0 & 0 \\ 2z & 0 & 4 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ z & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & z & z & 2z & 0 \\ 0 & z & z & 2z & 0 \\ 0 & z & z & 0 & 0 \\ 0 & 0 & 0 & 0 & 4z \end{bmatrix} \begin{bmatrix} 2 \\ 2z \\ 2z \\ 2z \\ 8z \end{bmatrix} = \begin{bmatrix} 4 \\ 4z \\ 12z \\ 12z \\ 4z \\ 4z \\ 8z \\ 40z \\ 4z \\ 8z^2 \\ 8z^2 \\ 4z^2 \\ 32z^2 \end{bmatrix} \tag{3.8}$$

Our partition $\mathcal{Q}$ of $\mathcal{T}$ groups columns 2 and 3 and represents combining the i-types $(0)(01011)$ and $(1)(00101)$, which corresponds to the automorphism on K_4 that swaps roots 0 and 1. Our partition $\mathcal{P}$ of $\mathcal{S}$ involves three pairings: rows 3 and 4, rows 5 and 6, and rows 10 and 11, which correspond to the automorphism on $K_4 + 01$ that swaps roots 0 and 1. This compresses the 13×5 matrix of (3.8) down to the 10×4 matrix on the left side of Equation (3.9).

$$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 2z & 0 & 0 & 0 \\ 4z & 4 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ z & 0 & 3 & 0 \\ 0 & 0 & 0 & 5 \\ 0 & 0 & 2 & 0 \\ 0 & 2z & 4z & 0 \\ 0 & z & 0 & 0 \\ 0 & 0 & 0 & 4z \end{bmatrix} \begin{bmatrix} 2 \\ 4z \\ 2z \\ 8z \end{bmatrix} = \begin{bmatrix} 4 \\ 4z \\ 24z \\ 8z \\ 8z \\ 40z \\ 4z \\ 16z^2 \\ 4z^2 \\ 32z^2 \end{bmatrix} \tag{3.9}$$

4 Two examples of linear families

In this section, we examine the application of the string operations Sup, Add, and Lab to two linear sequences previously studied elsewhere.

4.1 Production matrices

Given a linear family $\{G_n : n = 0, 1, \ldots\}$ of graphs, constructed by recursive application of the topological operator $\tau \colon G_n \to G_{n+1}$, and with the pgd-vector $V_n(z)$ for G_n, for $n = 0, 1, \ldots$, the associated ***production matrix*** $M_\tau(z)$ is a matrix such that we have the recursion

$$V_n(z) = M_\tau(z) V_{n-1}(z), \text{ for } n = 1, 2, \ldots \tag{4.1}$$

and, consequently, the equation

$$V_n(z) = M_\tau(z)^n V_0(z), \text{ for } n = 1, 2, \ldots \tag{4.2}$$

Here, as in some previous papers (e.g., [14, 17]), our production matrices record a system of rules that computer scientists might call *productions*.

4.2 X-ladders

An ***X-ladder*** is envisioned as a ladder with evenly many rungs, such that the rungs are paired, and such that within a pair, they cross each other in a planar drawing, as illustrated in Figure 3. This example was first given by [28].

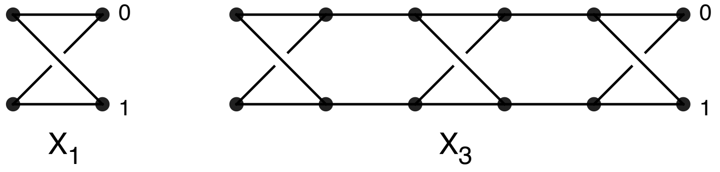

Figure 3: The X-ladders X_1 and X_3.

To represent the construction of X_n from X_{n-1}, we use the following procedure (a sequence of i-type operations) to add the next X:

Procedure 4.1. Add the next X to an X-ladder.

$$\mathrm{Sup}_{0,1} \circ \mathrm{Add}_{02431} \tag{4.3}$$
$$\mathrm{Sup}_{2,3} \circ \mathrm{Add}_{253} \tag{4.4}$$
$$\mathrm{Lab}_{40,51} \tag{4.5}$$

We denote the composition of the steps of Procedure 4.1 by Rec_X.

Since the X-ladder X_1 is simply a 4-cycle with labeled vertices 0 and 1, its one and only i-type is $(01)(01)$. To obtain the pgd-vector for X_2 from the pgd-vector for X_1, we apply Procedure 4.1. In this non-machine calculation, we separate Step (4.4) into two parts.

$$\begin{aligned}
\mathrm{Sup}_{0,1}[\mathrm{Add}_{02431}[(01)(01)]] &= 2(234)(243) + 2z(224334)\\
\mathrm{Sup}_{2,3}[\mathrm{Add}_{253}[2(234)(243)]] &= 4(4)(5)(45) + 4z(4545)\\
\mathrm{Sup}_{2,3}[\mathrm{Add}_{253}[2z(224334)]] &= 8z(45)(45)
\end{aligned}$$

By then applying $\mathrm{Lab}_{40,51}$, we obtain the production

$$\mathrm{Rec}_X[(01)(01)] = 4(0)(1)(01) + 8z(01)(01) + 4z(0101). \tag{4.6}$$

In general, a ***production*** for an i-type associates to it a linear combination of all the i-types, taken over the ring of polynomials in the indeterminate z.

Thus, there are three possible i-types for imbeddings of the X-ladder X_2. Since two of them are not i-types for X_1, we need to continue with the i-types of X_3, to see whether there are any additional i-types, before we write the production matrix.

To compute the effect of Rec_X on X_2, we need to compute its effect on the three imbedding-types $(0)(1)(01)$, $(01)(01)$, and (0101). We already know the production (4.6) for the imbedding-type $(01)(10)$.

We begin with i-type $(0)(1)(01)$.

$$\begin{aligned}\mathrm{Sup}_{0,1}[\mathrm{Add}_{02431}[(0)(1)(01)] &= (234)(243)+3z(224334)\\ \mathrm{Sup}_{23}[\mathrm{Add}_{253}[(234)(243)] &= 2(4)(5)(45)+2z(4545)\\ \mathrm{Sup}_{2,3}[\mathrm{Add}_{253}[3z(224334)]] &= \{12z(45)(45)\}\end{aligned}$$

By relabeling with $\mathrm{Lab}_{40,51}$, we obtain the production

$$\mathrm{Rec}_X[(0)(1)(01)] \;=\; 2(0)(1)(01)+12z(01)(01)+2z(0101). \tag{4.7}$$

We continue with the effect of Rec_X on i-type (0101).

$$\begin{aligned}\mathrm{Sup}_{0,1}[\mathrm{Add}_{02431}[(0101)]] &= 4(243)(342)\\ \mathrm{Sup}_{2,3}[\mathrm{Add}_{253}[4(234)(243)] &= 8(45)(4)(5)+8z(4545)\end{aligned}$$

Applying $\mathrm{Lab}_{40,51}$, we obtain the production

$$\mathrm{Rec}_X[(0101)] \;=\; 8(0)(1)(01)+8z(0101). \tag{4.8}$$

We see that no new types arise when applying Rec_X to X_2. Thus, the only possible i-types for any X-ladder X_n arising from application of Rec_X are

$$(0)(1)(01),\ (01)(01),\ \text{and } (0101).$$

Accordingly, we may write the pgd-vectors of X_1, X_2, and X_3 as

$$V_{X_1}=\begin{bmatrix}0\\1\\0\end{bmatrix}\quad V_{X_2}=\begin{bmatrix}4\\8z\\4z\end{bmatrix}\quad V_{X_3}=\begin{bmatrix}8+64z\\48z+64z^2\\8z+64z^2\end{bmatrix}$$

By recording the coefficients of the i-types in productions (4.6), (4.7), and (4.8) as columns of the production matrix $M_X(z)$ for Rec_X we have

$$M_X(z)=\begin{bmatrix}2&4&8\\12z&8z&0\\2z&4z&8z\end{bmatrix}$$

We see that $M_X(z)V_{X_1}(z)=V_{X_2}(z)$ and that $M_X(z)V_{X_2}(z)=V_{X_3}(z)$.

Proposition 4.1 enables us to check for possible errors.

Proposition 4.1. *Suppose that $\{G_n : n=0,1,\dots\}$ is a linear family with production matrix $M(z)$. Then substituting $z=1$ gives a matrix whose column sums are the same constant s, where the number of imbeddings of G_{n+1} equals s times the number of imbeddings of G_n.*

Proof. Substituting $z=1$ in any column of $M(z)$ counts the number s of ways that the extra paths can be added between the roots of G_n and the roots of G_{n+1}. This number is the same for each imbedding-type and hence for each column of $M(z)$. Clearly, s also tells us the growth factor in the number of imbeddings from G_n to G_{n+1}. □

As Proposition 4.1 indicates, the substitution $z=1$ in $M_X(z)$ gives column sums of $s=16$, implying that any imbedding of X_n of a given type generates 16 imbeddings of X_{n+1}. This makes sense since X_{n+1} has four more 3-valent vertices than L_n, so it should have $(2!)^4=16$ times as many imbeddings.

4.3 Iterated claws

This example is adapted from [14] and [17].

The iterated claw Y_1 is obtained from the complete bipartite graph $K_{3,3}$ as follows:

1. Choose one vertex of $K_{3,3}$ to be the root-vertex 0.
2. Subdivide each of the edges incident with 0.
3. Assign labels 1, 2, and 3 to the resulting three 2-valent vertices.

To obtain the graph $(Y_n, 0)$ from the graph $(Y_{n-1}, 0)$, we join a new 3-valent vertex 7 to the vertices 1, 2, and 3 by paths 741, 752 and 763. We then suppress labels 1, 2, 3, and 0 and relabel vertex 4 as 1, vertex 5 as 2, vertex 6 as 3, and vertex 7 as 0.

Figure 4 illustrates the graph Y_3. We observe that the graph Y_1 is homeomorphic to $K_{3,3}$.

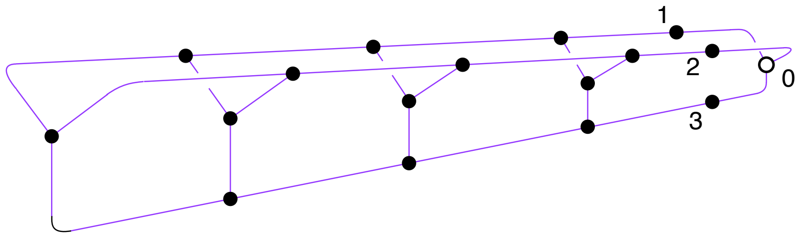

Figure 4: The iterated claw Y_3.

To obtain the pgd-vector of Y_n from the pgd-vector of Y_{n-1}, we now describe how to construct Y_n from Y_{n-1} with this procedure.

Procedure 4.2. Add the next claw to an iterated claw.

$$\mathrm{Sup}_{0,1,2} \circ \mathrm{Add}_{14752} \tag{4.9}$$

$$\mathrm{Sup}_{3} \circ \mathrm{Add}_{367} \tag{4.10}$$

$$\mathrm{Lab}_{41,52,63,70} \tag{4.11}$$

We denote the composition of the steps of Procedure 4.2 by Rec_Y.

We note that at the root vertex 0, there must be face corners 102, 203, and 301. We partition the set of i-types according to the number of faces incident with the root-vertex 0:

(a) three faces: the i-type must be $(013)(021)(032)$ or its reverse;

(b) two faces: the imbedding must be of one of the types $(013)(022031)$, $(021)(012033)$, $(032)(011023)$, or of their reverses;

(c) one face: the imbedding must be of types (011022033), (012031023), or of their reverses.

Thus, we have 12 i-types in all.

Grouping each i-type with its inverse yields six “super-types”. To reduce from six to three, we notice that the dihedral $\mathbb{D}_3$ symmetry of the claw is visible within the notation for the types. For instance, from the one group (b) i-type $(013)(022031)$, we could obtain any of the other the other i-types by a permutation of $1, 2, 3$ and a possible reversal. Thus, we need to consider only how path-adding affects the i-type $(013)(022031)$. On the other hand, the two one-face i-types are not related by a permutation of $1, 2, 3$. Nevertheless, we will see that grouping the two together does provide a compatible partition for the production matrix. We denote the three super-types simply by listing the face structure at 0:

(a) three faces: $(0)(0)(0)$;

(b) two faces: $(0)(00)$;

(c) one face: (000).

We now calculate $\mathrm{Rec}_Y[t]$ for one representative t from each of the three super-types. For i-type $t = (013)(021)(032)$ from super-type $(0)(0)(0)$, we obtain

$$\begin{aligned}\mathrm{Sup}_{0,1,2}[\mathrm{Add}_{14752}[t]] \quad = \quad & (475)(574)(3)(3) + z(4753574)(3) \\ & + z(3475574)(3) + z(34753574)\end{aligned}$$

By applying $\mathrm{Sup}_{3456} \circ \mathrm{Add}_{367}$ to the right side, we obtain

$$4z(77)(7) + z[2(77)(7) + 2z(777)] + z[2(77)(7) + 2z(777)] + z[4(77)(7)].$$

Collecting terms, we obtain

$$12z(7)(77) + 4z^2(777).$$

Relabeling 7 by 0 then yields the production

$$\mathrm{Rec}_Y[(102)(203)(301)] \quad = \quad 0(0)(0)(0) + 12z(0)(00) + 4z^2(000) \tag{4.12}$$

For i-type $t = (013)(022031)$ from super-type $(0)(00)$, we have:

$$\mathrm{Sup}_{012}[\mathrm{Add}_{14752}[t]] = 2(475)(3574)(3) + 2z(34753574).$$

Applying $\mathrm{Sup}_{3456} \circ \mathrm{Add}_{367}$ to the right side, we obtain:

$$2[(7)(7)(7) + 2z(7)(77) + z(777)] + 2z[4(7)(77)].$$

Then relabeling 7 by 0 yields the production

$$\mathrm{Rec}_Y[(013)(022031)] \quad = \quad 2(0)(0)(0) + 12z(00)(0) + 2z(000) \tag{4.13}$$

It is easily verified we would get the same result beginning instead with alternative representatives $t = (021)(012033)$ or $t = (032)(011023)$.

For i-type $t = (011022033)$ from super-type (000), we have

$$\mathrm{Sup}_{012}[\mathrm{Add}_{14752}[t]] \quad = \quad 4(475)(33547)$$

Applying $\mathrm{Sup}_{3456} \circ \mathrm{Add}_{367}$ to the right side yields

$$4[2(7)(7)(7) + 2z(777)].$$

Then relabeling 7 by 0, we obtain the production

$$\mathrm{Rec}_Y[(102203301] \quad = \quad 8(0)(0)(0) + 0(0)(00) + 8z(000) \tag{4.14}$$

It is easily verified that we get the same result when we begin with type $t = (012031023)$. That is what enables us to group them together in a super-type, even though they are not related by a permutation of $1, 2, 3$ or by reversal.

We copy the coefficients from (4.12), (4.13), and (4.14) into the columns of the production matrix $M_Y(z)$ for Rec_Y, with input and output basis $\{(0)(0)(0), (00)(0), (000)\}$.

$$M_Y(z) = \begin{bmatrix} 0 & 2 & 8 \\ 12z & 12z & 0 \\ 4z^2 & 2z & 8z \end{bmatrix}$$

We note that the column sums with $z = 1$ are $16 = 2^4$ and that Y_{n+1} has four extra vertices of valence 3. We observe that the power of string notation for i-types has allowed us to compute the recursion matrix for this family in only a page, while the original calculation [14] requires many pages and many figures. As in [14], we obtain the pgd-vectors

$$V_{Y_1} = \begin{bmatrix} 16z \\ 24z \\ 24z^2 \end{bmatrix} \quad V_{Y_2} = \begin{bmatrix} 48z + 192z^2 \\ 480z^2 \\ 48z^2 + 256z^3 \end{bmatrix} \quad V_{Y_3} = \begin{bmatrix} 1344z^2 + 2048z^3 \\ 576z^2 + 8064z^3 \\ 1536z^3 + 2816z^4 \end{bmatrix}$$

Of course, since $\mathbb{Z}[z]$ is a ring, rather than a field, a "pgd-vector" is more accurately described as an r-tuple than as a vector, where r is the number of i-types.

The functor relating a string operation $\tau\colon G \to H$ to the corresponding production matrix $M_\tau(z)\colon V_G(z) \to V_H(z)$, is represented by the commutative diagram in Figure 5.

$$\begin{tikzcd} G \arrow[r, "\tau"] \arrow[d] & H \arrow[d] \\ V_G(z) \arrow[r, "M_\tau(z)"] & V_H(z) \end{tikzcd}$$

Figure 5: Functor from the category of graphs and string operations to the category of ring modules and matrices with integer polynomial coefficients.

4.4 Polynomial matrix and transfer matrix methods

There are models in the physical sciences where the computational process uses polynomial matrix entries, like our production matrices. Some such models in chemistry were explored in [21, 22], which uses the terminology *polynomial matrix method.* This method was adapted by [1] for application to matching polynomials of *polygraphs.*

As described by [7], the *transfer matrix method* for various mathematical contexts concerns the transformation of a given problem into a matter of counting walks in a digraph. We observe that if A is the adjacency matrix of a digraph, then the ij entry of the matrix A^k counts the numbers of paths from vertex v_i to vertex v_j.

A generalization of this problem (see [29]) is concerned with a digraph in which the arc from vertex i to vertex j, for all i and j, is labeled with the element $m_{i,j}$ of a commutative ring, with $M = (m_{i,j})$. Instead of counting the paths of length k, we are calculating the sum of the products of all length-k paths from v_i to v_j. Of course, the ij entry of the matrix M^k gives this sum for v_i and v_j. In [5] and [23], the matrix M is called a "transfer matrix".

When calculating pgd-vectors for a graph sequence $\{G_n : n = 0, 1, \ldots\}$ that is specified by recursive application of a topological operation τ, we take the imbedding types as vertices of the digraph. We label the arc from type-i to type-j by the coefficient of type-j in the production for type-i.

5 Machine computation of production matrices

In this section, we give two examples of linear sequences whose production matrices have been calculated with the aid of a computer program. It should be clear that calculating these production matrices by hand would be daunting. Heretofore, such calculations have been done mostly by hand, which has limited us to calculating the genus polynomials only for relatively few graph families. As a consequence, we have very little data to study deep issues, such as the log-concavity conjecture, that the genus distribution of every graph is a log-concave polynomial (see [18, 16]).

5.1 Vertex-amalgamation path of copies of K_4

We define the graph T_1 to be the complete graph on four vertices, with a single root, labeled 0. The graph T_n is obtained from T_{n-1} by vertex-amalgamating a new copy of K_4 to T_{n-1}. The graphs T_2 and T_3 are illustrated in Figure 6.

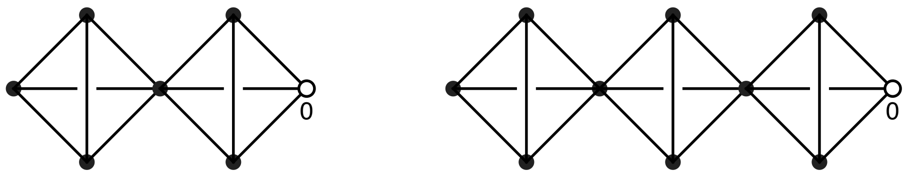

Figure 6: The graphs T_2 and T_3.

Following the paradigm of [13], we could obtain T_n from T_{n-1} by vertex-amalgamating a doubly rooted copy of K_4 to a singly rooted copy of T_{n-1}. However, whereas a pair of 2-valent root-vertices involves at most 10 i-types, it can be seen in Table 5 that for two 3-valent root-vertices, the number of i-types could be as large as 38. Moreover, the potential number of productions for amalgamating two graphs with 38 i-types could be as large as $38^2 = 1444$. In what follows, we see that using the string-operation paradigm enables us to reduce the number of i-types from 38 to 3.

The topological operation of vertex-amalgamating an additional copy of K_4 to the rooted graph $(T_{n-1}, 0)$ can be represented by the following sequence of string operations.

Procedure 5.1. Add the next copy of K_4 by vertex-amalgamation.

$$\mathrm{Add}_{01230} \tag{5.1}$$

$$\mathrm{Add}_{02} \tag{5.2}$$

$$\mathrm{Add}_{13} \tag{5.3}$$

$$\mathrm{Sup}_{0,1,3} \tag{5.4}$$

$$\mathrm{Lab}_{20} \tag{5.5}$$

We see that the i-types for a graph with a single 3-valent root-vertex named 0 are

$$(0)(0)(0) \quad (0)(00) \quad (000)$$

More generally, the number of i-types for a graph with a single k-valent root-vertex equals at most the number of partitions of the integer k. Nonetheless, even though only three productions would be needed, deriving them with pencil-and-paper calculations would be tedious work. Just for a start, there are 12 ways to insert the path 01230 into an imbedding of T_{n-1}, two ways between each of the three pairs of distinct corners at root-vertex 0 and two ways at each corner. The total number of imbeddings of T_n that are consistent with each imbedding of T_{n-1} is 480.

Theorem 5.1. *The pdg-vector of the graph T_n is $M^{n-1}\mathbf{V_1}$, where the initial pgd-vector $\mathbf{V_1}$ is $\begin{bmatrix}2 & 12z & 2z\end{bmatrix}^T$ and the production matrix is*

$$M_T(z) \;=\; \begin{bmatrix} 96z+18 & 80z+30 & 60 \\ 48z^2+156z & 220z & 360z \\ 144z^2+18z & 120z^2+30z & 60z \end{bmatrix} \tag{5.6}$$

Proof. The initial pgd-vector $\mathbf{V_1}$ for $(K_4, 0)$ and the production matrix are best calculated by a computer program. □

5.2 Edge-amalgamation path of copies of K_4

Here we define $\overline{T}_1$ to be the complete graph K_4 with a single root-edge 01. The graph $\overline{T}_n$ is obtained from $\overline{T}_n$ by edge-amalgamating a copy of K_4. The new root-edge is the edge in the new copy that is independent of the edge amalgamated to the previous root-edge. The graphs $\overline{T}_2$ and $\overline{T}_3$ are illustrated in Figure 7.

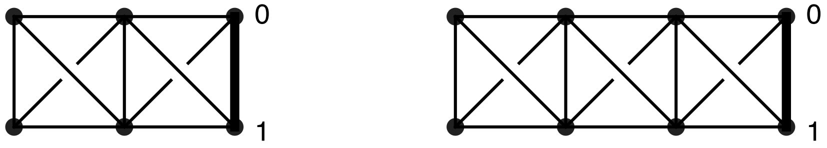

Figure 7: The graphs $\overline{T}_2$ and $\overline{T}_3$.

The topological operation of extending the graph $\overline{T}_{n-1}$ by edge-amalgamating an additional copy of K_4 can be represented by the following sequence of string operations.

Procedure 5.2. Add the next copy of K_4 by edge-amalgamation.

$$\mathrm{Add}_{0231} \tag{5.7}$$
$$\mathrm{Add}_{03} \tag{5.8}$$
$$\mathrm{Add}_{12} \tag{5.9}$$
$$\mathrm{Sup}_{0,1} \tag{5.10}$$
$$\mathrm{Lab}_{20,31} \tag{5.11}$$

We determine that the i-types for the graphs $\overline{T}_n$ are as follows, grouped by classes under the automorphism interchanging 0 and 1 and listed in shortlex order:

1. $(0)(1)(01)(01)$
2. $(0)(1)(0011)$
3. $(0)(01)(011), (1)(01)(001)$
4. $(0)(00111), (1)(00011)$
5. $(0)(01011), (1)(00101)$
6. $(01)(01)(01)$
7. $(01)(0011)$
8. $(01)(0101)$
9. $(001)(011)$
10. (000111)
11. $(001011), (001101)$
12. (010101)

Each imbedding of $\overline{T}_{n-1}$ in each of these 12 super-types has 576 possible extensions to an imbedding of $\overline{T}_n$.

Theorem 5.2. *The pdg-vector of the graph $\overline{T}_n$ is $\overline{M}^{n-1}(z)\mathbf{V}(z)$, where the production matrix is*

$$\begin{bmatrix}
4 & 18 & 8 & 36 & 40 & 6 & 20 & 22 & 12 & 72 & 80 & 84 \\
8z & 0 & 16z & 0 & 0 & 24z & 32z & 32z & 32z & 0 & 0 & 0 \\
64z & 96z & 96z & 96z & 96z & 96z & 128z & 128z & 128z & 0 & 0 & 0 \\
48z^2 & 32z^2 & 32z^2 & 0 & 0 & 48z^2 & 0 & 0 & 0 & 0 & 0 & 0 \\
8z & 36z & 16z & 72z & 80z & 12z & 40z & 44z & 24z & 144z & 160z & 168z \\
60z & 56z & 72z & 48z & 48z & 60z & 64z & 64z & 96z & 0 & 0 & 0 \\
104z^2+4z & 48z^2+18z & 64z^2+8z & 36z & 40z & 72z^2+6z & 20z & 22z & 12z & 72z & 80z & 84z \\
16z & 72z & 32z & 144z & 128z & 24z & 80z & 72z & 48z & 288z & 256z & 240z \\
104z^2 & 48z^2 & 64z^2 & 0 & 0 & 72z^2 & 0 & 0 & 0 & 0 & 0 & 0 \\
32z^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
64z^2 & 96z^2 & 96z^2 & 96z^2 & 96z^2 & 96z^2 & 128z^2 & 128z^2 & 128z^2 & 0 & 0 & 0 \\
60z^2 & 56z^2 & 72z^2 & 48z^2 & 48z^2 & 60z^2 & 64z^2 & 64z^2 & 96z^2 & 0 & 0 & 0
\end{bmatrix}$$

The initial graph $(\overline{T}_1, 0)$ has the pgd-vector

$$\overline{\mathbf{V}}(z) = \begin{bmatrix}2 & 0 & 0 & 0 & 4z & 0 & 2z & 8z & 0 & 0 & 0 & 0\end{bmatrix}^T.$$

Proof. The initial pgd-vector and the production matrix were calculated by our computer program. □

If follows that

$$\overline{\mathbf{T}}_\mathbf{2} = \begin{bmatrix}
8+376z \\
16z+320z^2 \\
128z+1664z^2 \\
96z^2 \\
16z+752z^2 \\
120z+832z^2 \\
584z^2+8z \\
32z+1248z^2 \\
208z^2 \\
64z^3 \\
128z^2+1664z^3 \\
120z^2+832z^3
\end{bmatrix}
\quad \text{and} \quad
\overline{\mathbf{T}}_\mathbf{3} = \begin{bmatrix}
32+5040z+119552z^2+207616z^3 \\
64z+9216z^2+111872z^3 \\
512z+56064z^2+612864z^3 \\
384z^2+28416z^3+103424z^4 \\
64z+10080z^2+239104z^3+415232z^4 \\
480z+43200z^2+365568z^3 \\
5872z^2+32z+176256z^3+389376z^4 \\
128z+19136z^2+414464z^3+644096z^4 \\
832z^2+56704z^3+181760z^4 \\
256z^3+12032z^4 \\
512z^2+56064z^3+612864z^4 \\
480z^2+43200z^3+365568z^4
\end{bmatrix}.$$

6 Enumerating possible imbedding types

Various previously published genus polynomial calculations have involved recursive constructions of families of graphs with two 2-valent root-vertices, for which ten i-types are sufficient. As we progress toward more general results, most especially in regard to the LCGD conjecture, we are encountering recursive graph constructions for which we use arbitrarily many vertex roots, of arbitrary degrees.

In this section, we first use Burnside's Lemma to calculate the number of i-types that can occur for two 2-valent roots. Then we generalize to obtain lower and upper bounds on the number of i-types for arbitrarily many root-vertices or arbitrary valences. Interestingly, our method provides a formula for calculating the number of possible cyclic partitions of a multi-set. Thus, it is a generalization of Stirling numbers of the first kind.

6.1 Two 2-valent roots

Early papers on genus polynomial calculations via pgd-vectors used ten mnemonics for the i-types for graphs with two 2-valent roots. The following table lists the ten mnemonics and their corresponding type-names:

dd^0	dd'	dd''	ds^0	ds'
$(0)(0)(1)(1)$	$(0)(01)(1)$	$(01)(01)$	$(0)(0)(11)$	$(0)(011)$
sd^0	sd'	ss^0	ss^1	ss^2
$(00)(1)(1)$	$(001)(1)$	$(00)(11)$	(0101)	(0011)

An ad hoc examination confirms that the ten type-names contain all the possible partitions of the multi-set $\{0, 0, 1, 1\}$ into cyclic cells. We now undertake a reconfirmation of this calculation of ten possible i-types, using Burnside's Lemma.

Our set of objects is the set of disjoint cycle decompositions of the 24 permutations in the symmetric group Σ_4, with domain $\{0, 1, 2, 3\}$. Our permutation group on them has the permutations

$$\epsilon \text{ (identity)} \quad (0\ 2) \quad (1\ 3) \quad (0\ 2)(1\ 3) \tag{6.1}$$

where we regard the numbers 2 and 3 as second copies of the numbers 0 and 1, respectively. Under the action of this permutation group, the orbit of the permutation $(0\ 1)(2)(3)$ is

$$(0)(1)(2\ 3) \quad (0)(3)(1\ 2) \quad (1)(2)(0\ 3) \quad (2)(3)(0\ 1)$$

This orbit corresponds to the imbedding-type $(0)(1)(01)$.

The identity permutation ϵ fixes all 24 disjoint cycle representations of Σ_4. The permutation $(0\ 2)$ fixes the subgroup of disjoint cycle representations in which both 0 and 2 are fixed or transposed, whose cardinality is 4. The permutation $(1\ 3)$ fixes the same subgroup of cardinality 4. The permutation $(0\ 2)(1\ 3)$ fixes that same subgroup, plus the set

$$(0\ 1)(2\ 3) \quad (0\ 3)(1\ 2) \quad (0\ 1\ 2\ 3) \quad (0\ 3\ 2\ 1)$$

for a total of 8 fixed points. Applying Burnside's Lemma, we divide the sum of the sizes of the fixed-point sets by the cardinality of the permutation group (6.1) to obtain

$$\frac{24 + 4 + 4 + 8}{4} = \frac{40}{4} = 10$$

as the maximum number of i-types for two 2-valent roots.

6.2 Two roots, 2-valent and 3-valent

Suppose that root 0 is 2-valent and root 1 is 3-valent. Then there are 18 imbedding-types, as in Table 4.

Table 4: Table of the i-types for two roots, one 2-valent and one 3-valent.

structure	imbedding types		
1^5	$(0)(0)(1)(1)(1)$		
$1^3\,2$	$(0)(0)(1)(11)$	$(0)(1)(1)(01)$	$(1)(1)(1)(00)$
$1\,2^2$	$(0)(01)(11)$	$(1)(00)(11)$	$(1)(01)(01)$
$1^2\,3$	$(0)(0)(111)$	$(0)(1)(011)$	$(1)(1)(001)$
$2\,3$	$(00)(111)$	$(01)(011)$	$(11)(001)$
$1\,4$	$(0)(0111)$	$(1)(0011)$	$(1)(0101)$
5	(00111)	(01011)	

The action of the permutation group $\Sigma_{\{0,2\}} \times \Sigma_{\{1,3,4\}}$ on the elements of $\Sigma_{\{0,1,2,3,4\}}$ has the cycle index

$$\frac{1}{12}\left[t_1^5 + 4t_1^3t_2 + 3t_1t_2^2 + 2t_2t_3\right].$$

We now consider the number of fixed points for each of the four permutation types.

Type t_1^5. The identity permutation fixes all 120 elements of $\Sigma_{\{0,1,2,3,4\}}$.

Type $t_1^3t_2$. Each permutation of structure $t_1^3t_2$ fixes 12 elements of $\Sigma_{\{0,1,2,3,4\}}$. For instance, $(0\ 2)$ fixes each of the six elements with the 1-cycles (0) and (2) and each of the six with the 2-cycle (02), for a total of 12. The sum of the sized of the fixed-point sets of the four permutations of structure $t_1^3t_2$ is 48.

Type $t_1t_2^2$. Each permutation of structure $t_1t_2^2$ fixes 8 elements of $\Sigma_{\{0,1,2,3,4\}}$. For instance, $(0\ 2)(1\ 3)$ fixes both of the elements with the 1-cycles (0), (2), and (4), both with the 2-cycle (02) and the 1-cycle (4), and also the four elements

$$(0\ 1)(2\ 3),\ (0\ 3)(1\ 2),\ (0\ 1\ 2\ 3),\ \text{and } (0\ 3\ 2\ 1)$$

for a total of 8. The sum of the sized of the fixed-point sets of the four permutations of structure $t_1t_2^2$ is 24.

Type $t_1^2t_3$. Each permutation of structure $t_1^2t_3$ fixes 6 elements of $\Sigma_{\{0,1,2,3,4\}}$. In particular, $(0)(2)(134)$ fixes $\mathbb{Z}_{\{0,2\}} \times \mathbb{Z}_{\{1,3,4\}}$, as does $(0)(2)(1\ 4\ 3)$. Together, they make a contribution of 12 to the sum of the sizes of the fixed point sets.

Type t_2t_3. These two permutations each fix the same 6 elements of $\Sigma_{\{0,1,2,3,4\}}$ as in the preceding case, for a net contribution of 12.

Applying Burnside's Lemma, we infer that the number of orbits is

$$\frac{120+48+24+12+12}{12} = \frac{216}{12} = 18.$$

6.3 Several roots of arbitrary degrees

We now calculate lower and upper bounds on the number of i-types.

Theorem 6.1. *For a class of graphs with roots* $0, 1, \ldots, k-1$ *of respective degrees* $d_0, d_1, \ldots, d_{k-1}$*, the number of i-types is at least*

$$\frac{(d_0 + d_1 + \cdots + d_{k-1})!}{d_0! d_1! \cdots d_{k-1}!} \tag{6.2}$$

Proof. In addition to their respective primary names $0, 1, \ldots, k-1$, each root j has $d_j - 1$ aliases chosen from among the numbers

$$k,\ k+1,\ \ldots,\ d_0 + d_1 + \cdots + d_{k-1}$$

with no two different primary names having any aliases in common. Accordingly, our set of objects is the set of disjoint cycle representations of the symmetric group Σ_K, where $K = d_0 + d_1 + \cdots + d_{k-1}$. The permutation group that acts on them is isomorphic to

$$\Sigma_{d_0} \times \Sigma_{d_1} \times \cdots \times \Sigma_{d_{k-1}}$$

Since the identity permutation fixes all the cycle forms of Σ_K, the sum of the sizes of the sets of fixed points is at least $K!$. The cardinality of the permutation group is $d_1! d_2! \cdots d_k!$. Thus, by Burnside's Lemma, a lower bound on the number of i-types is given by (6.2). □

Theorem 6.2. *For a class of graphs with roots* 0 *and* 1*, of respective degrees* a *and* b*, the number of i-types is at most*

$$\sum_c \prod_{k=1}^{n} k^{c_k} c_k! \sum_{\forall i, p_i + q_i = c_i} \ \sum_{(1^{p_1} 2^{p_2} \cdots a^{p_a}) \in P_a} \ \sum_{(1^{q_1} 2^{q_2} \cdots b^{q_b}) \in P_b} \frac{1}{\prod_{i=1}^{a} i^{p_i} p_i! \prod_{j=1}^{b} j^{q_j} q_j!},$$

where the sum $\sum_c$ *is over all partitions* $1^{c_1} 2^{c_2} \cdots n^{c_n} \in P_n$ *and* P_n *is the set of all partitions of the number* n*.*

Proof. The action of the permutation group

$$\Sigma_{\{1,3,4,\ldots,a+1\}} \times \Sigma_{\{2,a+2,a+3,\ldots,a+b\}}$$

on the elements of $\Sigma_{\{1,2,\ldots,n\}}$, where $n = a + b$, has the cycle index

$$C_{a,b} = \sum_{(1^{p_1} 2^{p_2} \cdots a^{p_a}) \in P_a} \ \sum_{(1^{q_1} 2^{q_2} \cdots b^{q_b}) \in P_b} \frac{\prod_{i=1}^{a} t_i^{p_i} \prod_{j=1}^{b} t_j^{q_j}}{\prod_{i=1}^{a} i^{p_i} p_i! \prod_{j=1}^{b} j^{q_j} q_j!},$$

where P_m is the set of all partitions of m. The number of fixed points for a permutation of cycle type $1^{c_1} 2^{c_2} \cdots n^{c_n}$ is given by

$$a! b! C_{a,b}(1^{c_1} 2^{c_2} \cdots n^{c_n}) \prod_{k=1}^{n} k^{c_k} c_k!,$$

where $C_{a,b}(1^{c_1}2^{c_2}\cdots n^{c_n})$ is the coefficient of $t_1^{c_1}t_2^{c_2}\cdots t_n^{c_n}$ in the polynomial $C_{a,b}$. Thus, each permutation of structure $t_1^{c_1}t_2^{c_2}\cdots t_n^{c_n}$ fixes

$$\prod_{k=1}^{n} k^{c_k}c_k! \sum_{\forall i, p_i+q_i=c_i} \sum_{(1^{p_1}2^{p_2}\cdots a^{p_a})\in P_a} \sum_{(1^{q_1}2^{q_2}\cdots b^{q_b})\in P_b} \frac{a!b!}{\prod_{i=1}^{a} i^{p_i}p_i! \prod_{j=1}^{b} j^{q_j}q_j!}.$$

elements of $\Sigma_{\{1,2,\ldots,n\}}$.

Applying Burnside's Lemma, we conclude that the number of orbits is given by

$$\sum_c \frac{1}{a!b!}\prod_{k=1}^{n} k^{c_k}c_k! \sum_{\forall i, p_i+q_i=c_i} \sum_{(1^{p_1}2^{p_2}\cdots a^{p_a})\in P_a} \sum_{(1^{q_1}2^{q_2}\cdots b^{q_b})\in P_b} \frac{a!b!}{\prod_{i=1}^{a} i^{p_i}p_i! \prod_{j=1}^{b} j^{q_j}q_j!}$$

which equals

$$\sum_c \prod_{k=1}^{n} k^{c_k}c_k! \sum_{\forall i, p_i+q_i=c_i} \sum_{(1^{p_1}2^{p_2}\cdots a^{p_a})\in P_a} \sum_{(1^{q_1}2^{q_2}\cdots b^{q_b})\in P_b} \frac{1}{\prod_{i=1}^{a} i^{p_i}p_i! \prod_{j=1}^{b} j^{q_j}q_j!},$$

where the sum $\sum_c$ is over all partitions $1^{c_1}2^{c_2}\cdots n^{c_n} \in P_n$. □

Applying our formula for $a, b \leq 10$, we obtain Table 5.

Table 5: The maximum number of i-types for two root-vertices, of valences a and b.

$a\backslash b$	1	2	3	4	5	6	7	8	9	10
1	2	4	7	12	19	30	45	67	97	139
2	4	10	18	34	56	94	146	228	340	506
3	7	18	38	74	133	233	385	623	977	1501
4	12	34	74	158	297	550	951	1614	2627	4202
5	19	56	133	297	602	1166	2133	3775	6437	10692
6	30	94	233	550	1166	2382	4551	8424	14953	25835
7	45	146	385	951	2133	4551	9142	17639	32680	58659
8	67	228	623	1614	3775	8424	17639	35492	68356	127443
9	97	340	977	2627	6437	14953	32680	68356	136936	264747
10	139	506	1501	4202	10692	25835	58659	127443	264747	530404

Theorem 6.3. *The formula corresponding to that of Theorem 6.2 for m roots of degrees $(a_1, a_2, \ldots, a_m)$ is given by*

$$\sum_c \prod_{k=1}^{n} k^{c_k}c_k! \sum_{\forall i, p_{1i}+p_{2i}+\cdots+p_{di}=c_i} \sum_{\substack{\forall d=1,2,\ldots,m,\\ \left(1^{p_{d1}}2^{p_{d2}}\cdots a_d^{p_{da_d}}\right)\in P_{a_d}}} \frac{1}{\prod_{d=1}^{m}\prod_{i=1}^{a_d} i^{p_{di}}p_{di}!},$$

where the sum $\sum_c$ is over all partitions $1^{c_1}2^{c_2}\cdots n^{c_n} \in P_n$.

Proof. This proof uses the same arguments as for Theorem 6.2. □

Using the formula from Theorem 6.3 for the calculations, we present in Table 6 the maximum number of imbedding-types for triply rooted graphs with root-vertices of valences $1 \leq i, j, k \leq 5$.

Table 6: The maximum number of imbedding-types for three roots, of valences i, j, k for $i = 1, 2, 3, 4, 5$.

$i = 1$

$j\backslash k$	1	2	3	4	5
1	6	14	28	52	90
2	14	38	84	170	316
3	28	84	206	450	899
4	52	170	450	1058	2254
5	90	316	899	2254	5110

$i = 2$

$j\backslash k$	1	2	3	4	5
1	14	38	84	170	316
2	38	120	290	644	1284
3	84	290	788	1886	4074
4	170	644	1886	4868	11214
5	316	1284	4074	11214	27556

$i = 3$

$j\backslash k$	1	2	3	4	5
1	28	84	206	450	899
2	84	290	788	1886	4074
3	206	788	2370	6146	14302
4	450	1886	6146	17170	42696
5	899	4074	14302	42696	112966

$i = 4$

$j\backslash k$	1	2	3	4	5
1	52	170	450	1058	2254
2	170	644	1886	4868	11214
3	450	1886	6146	17170	42696
4	1058	4868	17170	51630	137070
5	2254	11214	42696	137070	387146

$i = 5$

$j\backslash k$	1	2	3	4	5
1	90	316	899	2254	5110
2	316	1284	4074	11214	27556
3	899	4074	14302	42696	112966
4	2254	11214	42696	137070	387146
5	5110	27556	112966	387146	1161498

7 Conclusions

We have focused here primarily on the computational aspects involved in applying string operations toward the determination of genus polynomials of graphs. We recognize the following two immediate benefits of the string-operations paradigm:

1. It enables us to reduce the number of partial genus polynomials (one for each imbedding-type) into which a genus polynomial must be partitioned.
2. The imbedding-types, the production matrix, and the partial genus polynomials (which are the coordinates of a pgd-vector) can be calculated by a computer program, which enables us to generate a much larger set of experimental data.

Beyond using string operations in new calculations of enumerative results on graph imbeddings, some new theoretical insights may arise from them. One may reasonably consider how the paradigm of string operations relates to the log-concavity conjecture, that every genus polynomial is log-concave (see [16, 18]). We observe that using Theorem 4.7.2 of [29] could give generating functions for the individual entries of a power of a production matrix.

In a sequel [15], we regard a linear family of graphs as a Markov process is which the states are i-types and a slightly modified form of the production matrix is the transition matrix. We explore the properties of such Markov processes.

The methods described here seem amenable to extension. Suppose that instead of a fixed production matrix $M(z)$ for a graph sequence $\{G_n : n = 0, 1, \ldots\}$, with pgd-vectors $V_n(z)$ we had a sequence of production matrices $M_n(z)$, such that Recursion (4.1) was generalized to

$$M_n(z)v_n(z) = V_{n+1}(z),$$

and Equation (4.2) to

$$V_n(z) = M_{n-1}(z)M_{n-2}(z) \cdots M_0(z)V_0(z).$$

A tractable recursion or a closed formula for $M_n(z)$ would enable us to calculate the pgd-vector $V_n(z)$ reasonably rapidly. Of course, such a sequence of production matrices corresponds to a non-stationary Markov process.

References

[1] D. Babić, A. Graovac, B. Mohar and T. Pisanski, The matching polynomial of a polygraph, *Discrete Appl. Math.* **15** (1986), 11–24, doi:10.1016/0166-218x(86)90014-4.

[2] S. Beyer, M. Chimani, I. Hedtke and M. Kotrbčík, A practical method for the minimum genus of a graph: models and experiments, in: A. V. Goldberg and A. S. Kulikov (eds.), *Experimental Algorithms*, Springer, volume 9685 of *Lecture Notes in Computer Science*, pp. 75–88, 2016, doi:10.1007/978-3-319-38851-9_6, proceedings of the 15th International Symposium (SEA 2016) held in St. Petersburg, June 5 – 8, 2016.

[3] B. Bollobás and O. M. Riordan, A polynomial invariant of graphs on orientable surfaces, *Proc. London Math. Soc.* **83** (2001), 513–531, doi:10.1112/plms/83.3.513.

[4] Y. Chen, J. L. Gross, T. Mansour and T. W. Tucker, Recurrences for the genus polynomials of linear sequences of graphs, 2016, manuscript, 26 pages.

[5] T. Y. Chow and J. West, Forbidden subsequences and Chebyshev polynomials, *Discrete Math.* **204** (1999), 119–128, doi:10.1016/s0012-365x(98)00384-7.

[6] M. L. Furst, J. L. Gross and R. Statman, Genus distributions for two classes of graphs, *J. Comb. Theory Ser. B* **46** (1989), 22–36, doi:10.1016/0095-8956(89)90004-x.

[7] I. M. Gessel and R. P. Stanley, Algebraic enumeration, in: R. L. Graham, M. Grötschel and L. Lovász (eds.), *Handbook of Combinatorics, Volume II*, Elsevier, Amsterdam & MIT Press, Cambridge, Massachusetts, pp. 1021–1061, 1995.

[8] J. L. Gross, Genus distribution of graph amalgamations: self-pasting at root-vertices, *Australas. J. Comb.* **49** (2011), 19–38, https://ajc.maths.uq.edu.au/pdf/49/ajc_v49_p019.pdf.

[9] J. L. Gross, Genus distributions of cubic outerplanar graphs, *J. Graph Algorithms Appl.* **15** (2011), 295–316, doi:10.7155/jgaa.00227.

[10] J. L. Gross, Embeddings of cubic Halin graphs: genus distributions, *Ars Math. Contemp.* **6** (2013), 37–56, doi:10.26493/1855-3974.217.440.

[11] J. L. Gross, Embeddings of graphs of fixed treewidth and bounded degree, *Ars Math. Contemp.* **7** (2014), 379–403, doi:10.26493/1855-3974.366.dd1.

[12] J. L. Gross and M. L. Furst, Hierarchy for imbedding-distribution invariants of a graph, *J. Graph Theory* **11** (1987), 205–220, doi:10.1002/jgt.3190110211.

[13] J. L. Gross, I. F. Khan and M. I. Poshni, Genus distribution of graph amalgamations: pasting at root-vertices, *Ars Combin.* **94** (2010), 33–53.

[14] J. L. Gross, I. F. Khan and M. I. Poshni, Genus distributions for iterated claws, *Electron. J. Combin.* **21** (2014), #P1.12, http://www.combinatorics.org/ojs/index.php/eljc/article/view/v21i1p12.

[15] J. L. Gross, T. Mansour and T. W. Tucker, Markovian analysis of production matrices for genus polynomials, in preparation.

[16] J. L. Gross, T. Mansour, T. W. Tucker and D. G. L. Wang, Log-concavity of combinations of sequences and applications to genus distributions, *SIAM J. Discrete Math.* **29** (2015), 1002–1029, doi:10.1137/140978867.

[17] J. L. Gross, T. Mansour, T. W. Tucker and D. G. L. Wang, Iterated claws have real-rooted genus polynomials, *Ars Math. Contemp.* **10** (2016), 255–268, doi:10.26493/1855-3974.538.86e.

[18] J. L. Gross, D. P. Robbins and T. W. Tucker, Genus distributions for bouquets of circles, *J. Comb. Theory Ser. B* **47** (1989), 292–306, doi:10.1016/0095-8956(89)90030-0.

[19] J. L. Gross and T. W. Tucker, *Topological Graph Theory*, Dover Publications, Mineola, New York, 2001, reprint of the 1987 original [Wiley, New York] with a new preface and supplementary bibliography.

[20] G. A. Jones and J. Wolfart, *Dessins d'enfants on Riemann surfaces*, Springer Monographs in Mathematics, Springer, Cham, 2016, doi:10.1007/978-3-319-24711-3.

[21] M. V. Kaulgud and V. H. Chitgopkar, Polynomial matrix-method for calculation of π-electron energies for linear conjugated polymers, *J. Chem. Soc. Faraday Trans. II* **73** (1977), 1385–1395, doi:10.1039/f29777301385.

[22] M. V. Kaulgud and V. H. Chitgopkar, Polynomial matrix method for the calculation of charge densities and bond orders in linear conjugated π-electron systems, *J. Chem. Soc. Faraday Trans. II* **74** (1978), 951–957, doi:10.1039/f29787400951.

[23] B. Mohar, Genus distribution of path-like and ring-like graphs, oral presentation at SIAM DM'12 at Halifax, Nova Scotia, June 2012.

[24] M. Mulase and M. Penkava, Ribbon graphs, quadratic differentials on Riemann surfaces, and algebraic curves defined over $\overline{\mathbb{Q}}^*$, *Asian J. Math.* **2** (1998), 875–919, doi:10.4310/ajm.1998.v2.n4.a11.

[25] M. I. Poshni, I. F. Khan and J. L. Gross, Genus distributions of graphs under edge-amalgamations, *Ars Math. Contemp.* **3** (2010), 69–86, doi:10.26493/1855-3974.110.6b6.

[26] M. I. Poshni, I. F. Khan and J. L. Gross, Genus distributions of graphs under self-edge-amalgamations, *Ars Math. Contemp.* **5** (2012), 127–148, doi:10.26493/1855-3974.166.63e.

[27] S. Stahl, Permutation-partition pairs III: Embedding distributions of linear families of graphs, *J. Comb. Theory Ser. B* **52** (1991), 191–218, doi:10.1016/0095-8956(91)90062-O.

[28] S. Stahl, On the zeros of some genus polynomials, *Canad. J. Math.* **49** (1997), 617–640, doi:10.4153/cjm-1997-029-5.

[29] R. P. Stanley, *Enumerative combinatorics, Volume I*, The Wadsworth & Brooks/Cole Mathematics Series, Wadsworth & Brooks/Cole Advanced Books & Software, Monterey, California, 1986, doi:10.1007/978-1-4615-9763-6.

[30] C. Thomassen, The genus problem for cubic graphs, *J. Comb. Theory Ser. B* **69** (1997), 52–58, doi:10.1006/jctb.1996.1721.

[31] Wikipedia contributors, Shortlex order — Wikipedia, The Free Encyclopedia, 2015, `https://en.wikipedia.org/wiki/Shortlex_order`.