CALCULATING GENUS POLYNOMIALS VIA STRING OPERATIONS AND MATRICES

JONATHAN L. GROSS, IMRAN F. KHAN, TOUFIK MANSOUR, AND THOMAS W. TUCKER

Abstract. In calculations of genus polynomials for a recursively specifiable sequence of graphs, the imbeddings of each of the graphs are partitioned into *imbedding-types*. The effects of a recursively applied graph operation τ on each imbedding-type are represented by a *production matrix*. We demonstrate herein how representing 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. It also allows us to reduce the number of imbedding-types, which lets us calculate some genus polynomials that were heretofore computationally infeasible.

1. Introduction

The ***genus polynomial*** of a graph G is the generating function $\Sigma g_i(G)z^i$, where $g_i(G)$ counts the cellular imbeddings of G in the closed oriented surface S_i of genus i. Since their introduction [GF87] in 1987, the genus polynomials for a recursively constructed sequence of graphs has most frequently been calculated, as in [GKP10, Gr11a, Gr11b], by partitioning the imbeddings according to the cyclic orderings of root-vertices on the face-boundary walks (abbr. ***fb-walks***) of the imbedding. In this paper, we describe how to automate such calculations.

1.1. **Rotation systems.** 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 a ***rotation***. The rotations act collectively as a permutation ρ

Date: **September 21, 2015.**

2000 *Mathematics Subject Classification.* 05A15, 05A20, 05C10.

Key words and phrases. graph imbedding, genus polynomial, production matrix, transfer matrix method.

J.L. Gross is supported by Simons Foundation Grant #315001.

T.W. Tucker is supported by Simons Foundation Grant #317689.

on the oriented edge set, as a ***rotation system*** (e.g., see [GT87]). If λ is the involution that reverses the orientation of each edge, then the fb-walks of the imbedding are the orbits of the permutation $\rho\lambda$. We use the Euler polyhedral formula

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

to compute the genus of the imbedding surface S. The problem of calculating genus polynomials is that the number of possible cyclic orderings of edge-ends incident at a d-valent vertex is $(d-1)!$. 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 not previously been published. The following set of coefficients has been obtained by M. Kotrbcik via a program based on the Heffter-Edmonds algorithm that ran for several hundred hours:

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

We have recently obtained the same genus polynomial via a program based on string-operations.

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 are *partitioned* into what we now call *imbedding-types*, according to incidence of the face-boundary walks on the roots. This basic paradigm is exemplified by [GKP10, Gr11a] for root-vertices, and by [PKG10] 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, 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.
- the representation of production systems by matrices, now called ***production matrices***, which was introduced by Stahl [Stah91];
- the representation of *imbedding-types* by strings of root-vertices, as presented by Gross [Gr12] in January 2012; and
- using string operations directly to calculate the production matrices, as suggested by Mohar [Mo12] in June 2012.

We explain in Section 4 how our use of productions to calculate pgd-vectors is a generalization of the *transfer matrix method*, along the lines described by [Stan86].

To this date, the production matrices for recursively defined families have been computed by hand, taking many pages and many figures (e.g., see [Gr13]). An achievement of this paper is to automate the bookkeeping necessary for the computation of a production matrix. All the imbeddings of interest here are in oriented surfaces.

1.3. **Outline of this paper.** Section 2 describes the representation of imbedding-types by strings. Section 3 introduces the representation of topological operations on imbeddings by string operations. Section 4 applies these representations to two previously published examples. Section 5 explores issues related to machine computation. It uses the theory developed 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.

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 automated calculation is a virtual necessity when seeking to derive genus polynomials for such graphs, as well as when seeking concrete coefficients in the genus polynomials.

2. Representing Imbedding-Types by Strings

In this section, we develop a notation using strings of root-labels so that representing addition of an edge to a graph becomes simply a matter of applying a few 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 labels, in the order they are encountered in a traversal of its fb-walk following the orientation of the surface. 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.

Remark 2.1. Unlabeled vertices do not appear in the string representing a face, so the appearance of consecutive labels 12 does not imply that there is an edge between 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.** A list of 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***). Fb-walks containing no labeled vertices would appear as empty strings, and they are not included. A collection C of imbedding types for G is ***full*** if every imbedding of G is represented by some i-type $t \in C$.

The imbeddings $\iota_1 : G \to S$ and $\iota_2 : G \to S$ of a rooted graph $(G, j_1, j_2, \ldots, j_r)$ are ***congruent as rooted-graph imbeddings*** if there exists an auto-homeomorphism $h : S \to S$ that permutes the roots (possibly the identity permutation), and a graph automorphism $\alpha : G \to G$ such that $h \circ \iota_1 = \iota_2 \circ \alpha$, in which case we say the i-types for imbeddings ι_1 and ι_2 are ***congruent i-types***. The congruence relation is illustrated by the following commutative diagram. Intuitively, this means that the two imbeddings "look alike".

$$\begin{tikzcd}
G \arrow[r, "\alpha"] \arrow[d, "\iota_1"'] & G \arrow[d, "\iota_2"] \\
S \arrow[r, "h"] & S
\end{tikzcd}$$

Figure 2.1. The congruence relation for imbeddings ι_1 and ι_2.

Example 2.1. In Figure 2.2, we see six graph imbeddings, which represent the five i-types of K_4. The 16 imbeddings of K_4 are partitioned as follows:

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

Thus, the set

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

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

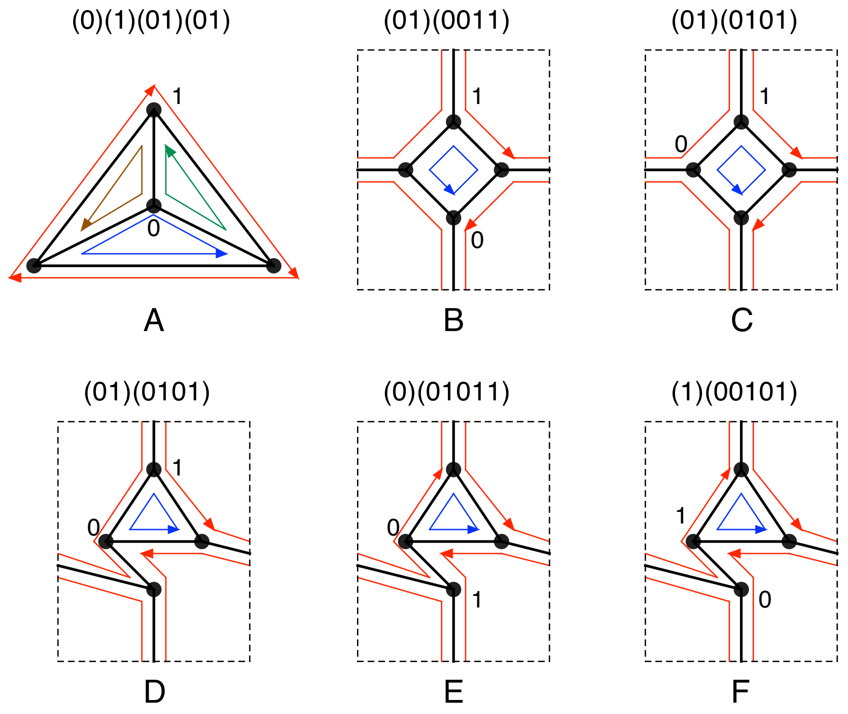

FIGURE 2.2. The five i-types of K_4, plus a duplicate.

Remark 2.2. We observe that although imbeddings B and C are unrooted-congruent, they are not rooted-congruent, and their i-types are different. Similarly, imbeddings D and E are unrooted-congruent, but not rooted-congruent, and they have different i-types. We notice also, that although imbeddings E and F are not of the same i-type, they are congruent as rooted-graph imbeddings, with a swap of roots 0 and 1; we return to this circumstance in §3.5. Finally, we notice that although imbeddings C and D are evidently non-congruent, they have the same i-type.

Remark 2.3. We observe that in each imbedding type, each root-vertex appears as many times as its valence.

Remark 2.4. Suppose that G has no multi-edges or self-loops and that we label every vertex. Then each rotation system for G uniquely determines an i-type, so the number of i-types is the same as the number of rotation systems. At the opposite extreme, suppose that G is a bouquet B_n of n self-loops (a one-vertex graph of valence $2n$). Then the number of possible i-types is the same as the number of partitions of $2n$, which is far less than the number $(2n-1)!$ of rotation systems for G.

2.3. **String notational conventions.**

We introduce 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 strings are not permutations (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 we order the i-types, 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 2.1 corresponds to the vector

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

3. Operations on Imbedding-Types

We now describe how a path-adding operation affects i-types. We also describe the relabeling of root-vertices, and the suppression of some root-labels, 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.**

Figure 3.1 shows the four possible ways to add the path $0U1$ to a fully labeled 4-cycle in the sphere and the resulting imbedding-type for each.

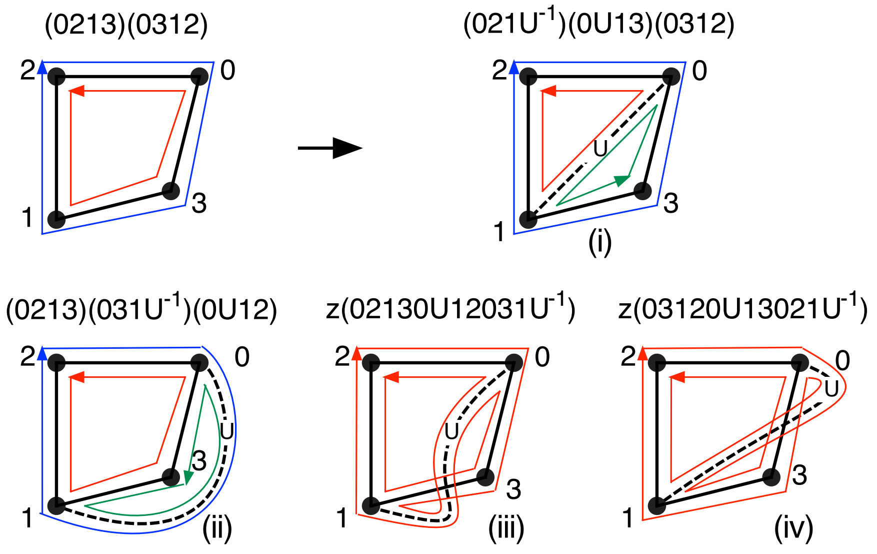

Figure 3.1. Adding a path to a 4-cycle in the sphere.

(i) Inserting path 0U1 into the inner face yields the imbedding type $(021U^{-1})(0U13)(0312)$. 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. Suppressing labels 2 and 3 and the vertices of U yields the i-type (01)(01)(01).

(ii) Inserting the path 0U1 instead into the outer face yields i-type $(0213)(031U^{-1})(0U12)$. Suppressing labels 2 and 3 and the vertices of U yields i-type (01)(01)(01), as in case (i).

(iii) If we join the two faces, from inside the inner face at endpoint 0, to endpoint 2 inside the outer face, then the resulting string expression is $z(02130U12031U^{-1})$. Preceding an imbedding-type by the indeterminate z corresponds to an increase of one in the genus. Label suppression yields the string expression $z(010101)$.

(iv) If we add the path 0U1 with edge-end 0 now inside the outer face and edge-end 1 inside the inner face, we get the string expression $z(03120U13021U^{-1})$. Label suppression yields $z(010101)$, as in case (iii).

Remark 3.1. In hand calculations, we do not need to standardize the representation of an i-type. *Canonical forms* (see §5.1) are necessary for use in machine calculation, but optional for hand calculation.

In general, given occurrences of roots i and j in a single face $(iSjT)$ (possibly $i = j$), we define the operation of ***adding a path*** $P = iUj$ ***within the face*** $(iSjT)$ by the rule

$$\begin{aligned} Add_P[(iSjT)] &= (SjU^{-1}i)(iUjT) = (jU^{-1}iS)(iUjT) \\ &= (SP^{-1})(PT) = (P^{-1}S)(PT) \end{aligned} \tag{3.1}$$

Figures 3.1(i) and (ii) above illustrate the add-within operation. Vertices i and j each appear in the i-type $(SjU^{-1}i)(iUjT)$ one more time than in the antecedent i-type $(iSjT)$, since the valences of i and j both increase by one under path-adding.

If a graph already has an edge ij, then adding the path $P = ij$ creates a multiple adjacency. We also allow the path $P = ii$ for adding a self-loop at vertex i. As a variation on Rule (3.1),we have the rule

$$Add_{ii}[(iS)] = (i)(iSi) \tag{3.2}$$

Given occurrences of i and j in different faces (iS) and (jT) (possibly $i = j$), we also define the operation of ***adding a path*** $P = iUj$ ***between faces***(iS) and (jT) by the rule

$$\begin{aligned} Add_P[(iS),(jT)] &= z(SiUjTjU^{-1}i) = z(iUjTjU^{-1}iS) \\ &= z(SPTP^{-1}) = z(P^{-1}SPT) \end{aligned} \tag{3.3}$$

Figures 3.1(iii) and (iv) above illustrate the add-between operation.

3.2. Adding a path to an imbedding-type.

To add a path $P = iUj$ to an i-type t with many faces, we express the totality of possible occurrences of i and j within and between the faces of i-type t. All faces containing neither i nor j are left alone. Thus, the result of adding a path to an i-type is a linear combination (over the ring $\mathbb{Z}[z]$ of polynomials with integer coefficients) of i-types.

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

$$Sup_{1,2}[(1)(12)(0212)(0231303)] \;=\; (0)(03303).$$

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 $Lab_{ij}[t]$ is the i-type obtained by replacing in t all occurrences of i by j. Thus,

$$Lab_{24}[(1)(2)(22)(1323)] \;=\; (1)(4)(44)(1343).$$

We denote by $Lab_{ii',jj',\dots}[t]$ the result of relabeling i by i', j by j' etc.

3.4. **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 3.1. *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

$$\begin{aligned}(3.4)\; Add_P[(iSjT)]^{-1} \;&=\; [(SP^{-1})(PT)] \;=\; (T^{-1}P^{-1})(S^{-1}P) \text{ and}\\ (3.5)\; Add_P[(iSjT)^{-1}] \;&=\; Add_P[iT^{-1}jS^{-1}] \;=\; (T^{-1}P^{-1})(S^{-1}P)\end{aligned}$$

Using Rule (3.3) for adding an edge between two faces, we have

$$(3.6) \qquad Add_P[(iS),(jT)]^{-1} \;=\; z(PTP^{-1}S)^{-1} = z(S^{-1}PT^{-1}P^{-1})$$

and

$$\begin{aligned}(3.7) \quad Add_P[(iS)^{-1},(jT)^{-1}] \;&=\; Add_P[(iS^{-1}),(jT^{-1})]\\ &=\; z(PT^{-1}P^{-1}S^{-1})\end{aligned}$$

□

3.5. Combining i-types into super-types.

It is sometimes possible to reduce the work needed to calculate a genus polynomial formula for a recursively specified family of graphs by combining i-types. The most frequently encountered application of this reduction is when each i-type is combined with the i-type obtainable by reversing all the fb-walks.

4. Two Examples of Linear Families

For our present purposes, a ***linear family*** is a sequence of graphs $\{G_n : n = 0, 1, \ldots\}$ having the same root-labels (but different roots) and the same full collection of i-types for those roots, where a recursively applied topological operator $\tau : G_n \to G_{n+1}$ is specified as a sequence of path additions between root-vertices. We require that the operator τ is the same for all $n \geq 0$. This definition includes any "H-linear family", in the sense of Stahl [Stah91, Stah97], who described such a family an as a recursively specified sequence of graphs in which the recursive topological operation is attaching an additional copy of some subgraph H.

4.1. Production matrices.

Given a linear family $\{G_n : n = 0, 1, \ldots\}$ of graphs, constructed by recursive application of the topological operator $\tau : G_n \to G_{n+1}$, 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

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

and, consequently, the equation

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

Here, as in some previous papers (e.g., [GKP14, GMTW15b]), our production matrices record a system of rules that computer scientists might call *productions*.

4.2. ***X*-ladders.** This example was first given by [Stah97]. An ***X*-ladder** is envisioned as a ladder with evenly many rungs, such that the rungs are paired and within a pair, they cross each other in a planar drawing, as illustrated in Figure 4.1.

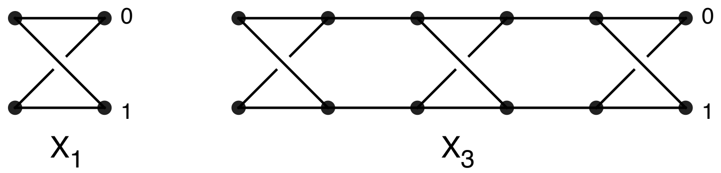

FIGURE 4.1. The X-ladders X_1 and X_3.

To represent the construction of X_n from X_{n-1}, we use the following sequence of i-type operations:

(1) Add a path 02431 from vertex 0 to vertex 1 and suppress vertices 0 and 1.
(2) Add a path 253 from vertex 2 to vertex 3 and suppress vertices 2 and 3.
(3) Relabel vertices 4 and 5 as 0 and 1, respectively.

We denote by Rec_X (for recursion) the composition of these operations.

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)(10). To obtain the pgd-vector for X_2 from the pgd-vector for X_1, we proceed as follows:

$$\begin{aligned} Sup_{01}[Add_{02431}[(01)(10)]] &= 2(342)(243) + 2z(243342) \\ Sup_{23}[Add_{253}[2(342)(243)]] &= 4(45)(4)(5) + 4z(4545) \\ Sup_{23}[Add_{253}[2z(243342)]] &= 8z(54)(45) \end{aligned}$$

By then applying $Lab_{40,51}$, we obtain the production

$$Rec_X[(01)(10)] = 4(0)(1)(01) + 8z(01)(01) + 4z(0101). \tag{4.3}$$

for type (01)(10).

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, the X-ladder X_2 has three imbedding types. Since this is more than the number for X_1, we need to calculate the i-types of X_3, to be sure that we have all the 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 (01)(0)(1), (01)(01), and (0101). We already know the production (4.3) for the imbedding-type (01)(10). We now proceed as follows:

$$\begin{aligned} Sup_{01}[Add_{02431}[(01)(0)(1)]] &= (243)(342) + 3z(243342) \\ Sup_{23}[Add_{253}[(243)(342)] &= 2(45)(4)(5) + 2z(4545) \\ Sup_{23}[Add_{253}[3z(243342)]] &= \{12z(54)(45)\} \end{aligned}$$

By then applying $Lab_{(40),(51)}$, we obtain the production

$$(4.4)\ Rec_X[(01)(0)(1)] \ = \ 2(01)(0)(1) + 12z(01)(01) + 2z(0101)$$

for type (01)(0)(1).

The calculation

$$\begin{aligned} Sup_{01}[Add_{02431}[(0101)]] &= 4(243)(342) \\ Sup_{23}[Add_{253}[4(243)(342)]] &= 8(45)(4)(5) + 8z(4545) \\ \therefore Rec_X[(0101)] &= 8(01)(0)(1) + 8z(0101) \end{aligned} \tag{4.5}$$

gives the production for type (0101).

We see that no new types arise. Thus, the only possible i-types for any X-ladder X_n are

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

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

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

It follows from (4.3), (4.4), and (4.5) that the production matrix $M_X(z)$ for Rec_X is:

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

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* $\{G_n : n = 0, 1, \ldots\}$ *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}*, is* 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 [GKP14] and [GMTW15b].

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.

Figure 4.2 illustrates the graph Y_3. We observe that the graph Y_1 is homeomorphic to $K_{3,3}$.

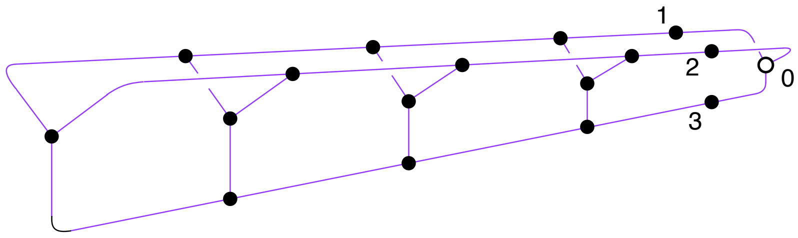

FIGURE 4.2. The iterated claw Y_3.

To obtain the graph $(Y_n, 0)$ from the graph $(Y_{n-1}, 0)$, we join a new 3-valent vertex v to the vertices 1, 2, and 3 by paths $v41$, $v52$ and $v63$. We then suppress labels 1, 2, 3, and 0 and relabel 4 by 1, 5 by 2, 6 by 3, and v by 0. 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}, using only these these type-operations.

(1) Add the path $14v52$ and suppress vertices 1, 2 and 0.
(2) Add the path $36v$ and suppress 3.
(3) Relabel vertices $4, 5, 6, v$ as $1, 2, 3, 0$, respectively.

We call the resulting operator Rec_Y.

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

(a) three faces: the imbedding-type must be (102)(203)(301) or its reverse;

(b) two faces: the imbedding must be of one of the types (102203)(301), (203301)(102), (301102)(203), or their reverses;
(c) one face: the imbedding must be of types (102203301), (102301203), or their reverses.

This gives 12 types in all. We first cut the number in half by grouping a type with its inverse. To further reduce the number of imbedding-types from six to three, we begin by noticing that the dihedral $\mathbb{D}_3$ symmetry of the claw is visible within the notation for the types. For instance, from the one (b)-type (102203)(103), we could obtain the other types by a permutation of $1, 2, 3$. Thus, we need to consider only how path-adding affects the imbedding-type (102203)(13). We denote the three classes simply by listing the face structure at 0:

(a) three faces: (0)(0)(0);
(b) two faces: (00)(0);
(c) one face: (000).

Using these three imbedding classes, we can replace step (3) above by the step:

Suppress $4, 5, 6$ and relabel v as 0.

We now calculate $Rec[t]$ for one representative t from each class.

For $t = (102)(203)(301)$ from class (0)(0)(0), we have

$$\begin{aligned} Sup_{012}[Add_{14v52}[t]] &= (4v5)(5v4)(3)(3) + z(4v535v4)(3) \\ &\quad + z(34v55v4)(3) + z(34v535v4) \end{aligned}$$

When we apply $Sup_{3456} \circ Add_{36v}$ to the right side, we obtain

$$4z(vv)(v) + z[2(vv)(v) + 2z(vvv)] + z[2(vv)(v) + 2z(vvv)] + z[4(vv)(v)].$$

Relabeling v by 0 then yields the production

$$(4.6)\; Rec_Y[(102)(203)(301)] = 0(0)(0)(0) + 12z(00)(0) + 4z^2(000)$$

For type $t = (102203)(301)$ from class (00)(0), we have:

$$Sup_{012}[Add_{14v52}[t]] = 2(4v5)(35v4)(3) + 2z(34v535v4).$$

Applying $Sup_{3456} \circ Add_{36v}$ to the right side, we obtain :

$$2[1(v)(v)(v) + 2z(vv)(v) + z(vvv)] + 2z[4(vv)(v)].$$

Then relabeling v by 0 yields the production

$$(4.7)\; Rec_Y[(102203)(301)] = 2(0)(0)(0) + 12z(00)(0) + 2z(000)$$

It is easily verified we get the same result beginning instead with $t = (203301)(102)$ or $t = (301102)(203)$.

For type $t = (102203301)$ from class (000), we have:

$$Sup_{012}[Add_{14v52}[t]] \;=\; 4(4v5)(3354v)$$

Applying $Sup_{3456} \circ Add_{36v}$ to the right side, we get:

$$4[2(v)(v)(v) + 2(vvv)].$$

Then relabeling v by 0, we get the production

$$Rec_Y[(102203301] \;=\; 8(0)(0)(0) + 0(00)(0) + 8z(000) \tag{4.8}$$

It is easily verified that we get the same result beginning with type $t = (102301203)$; in other words, the i-types (102203301) and (102301203) are equivalent, in the sense defined in §3.5.

We conclude from (4.6), (4.7), and (4.8) that the production matrix for Rec_Y with input and output basis $\{(0)(0)(0), (00)(0), (000)\}$ is given by

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

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 notational power of i-types has allowed us to compute the recursion matrix for this family in only a page, while the original calculation [GKP14] requires many pages and many figures. As in [GKP14], we obtain the pgd-vectors

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

The covariant functor relating a string operation $\tau : G \to H$ to the corresponding production matrix $M_\tau(z) : V_G(z) \to V_H(z)$, is represented by the following commutative diagram:

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

FIGURE 4.3. Functor from the category of graphs and string operations to the category of ring modules and matrices with integer polynomial coefficients.

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.

4.4. **Generalized transfer matrix method.** The *transfer matrix method* (see [GS95]), 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 [Stan86]) 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 [ChWe99] and [Mo12], 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

A major impetus for this paper is to provide the data structures necessary for machine computation of the production matrices for the pgd-vectors for various families of graphs. Heretofore, all such calculations have been done by hand, and we have calculated the genus polynomials only for a relatively few of families. As a consequence, we actually 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 [GRT89, GMTW15a]). For this section, we have used a computer program to calculate recursion matrices for two families of graphs.

5.1. **Lexicographically ordering imbedding-types.** One of several computational benefits of representing topological operations by string-operations is the possibility of reducing the number of i-types, as we saw in Subsection 4.3 for iterated claws. We have already observed that the number of i-types tends to grow rapidly as the number of roots increases, or as their valences increase. Another benefit of string notation for i-types is that they have a natural linear order, thus allowing rapid comparison, search, and storage in any machine computation of production matrices.

The ***canonical form of a string-based i-type*** is obtained from an arbitrary representative of that type in three steps:

(1) The cycles within each imbedding-type are arranged by size in non-descending order.
(2) The string within each cycle is sorted into its earliest lexicographic form.
(3) The cycles of each size are arranged in lexicographically non-descending order.

Example 5.1. Consider the imbedding-type

$$(01120)(11020)(10)(210)(201)(2) \tag{5.1}$$

Reorganizing the cyclic strings of (5.1) by length gives the form

$$(2)(10)(210)(201)(01120)(11020)$$

Writing each fb-walk in earliest lexicographic form yields

$$(2)(01)(021)(012)(00112)(01102)$$

Reordering canonical forms of fb-walks of the same size lexicographically gives the final canonical form for the i-type (5.1)

$$(2)(01)(012)(021)(00112)(01102)$$

The canonical forms of i-types have a ***lexicographical order***.

(1) The primary criterion is the cycle structure, which is determined as if the cyclic partition were a disjoint cycle representation of a permutation. For instance, the imbedding-types (0)(1)(01) and (1)(001) have cycle structures $t_1^2t_2$ and t_1t_3, respectively. This implies that smaller cycle sizes precede larger cycle sizes for each imbedding-type.
(2) The secondary criterion, which is applied within the sublist of imbedding-types corresponding to each fixed cycle structure, is lexicographically non-descending.

We now give two examples of linear families whose production matrices have been calculated by machine. It should be clear that calculating these production matrices by hand would be daunting.

5.2. **Vertex-amalgamation path of copies of K_4.** We define the graph T_1 to be the graph K_4 with a single root, labeled 0. The graph T_n is obtained by vertex-amalgamating a copy of K_4 with two root-vertices to T_{n-1}, with the new root, also called 0, on the newly added copy. The graphs T_2 and T_3 are illustrated in Figure 5.1.

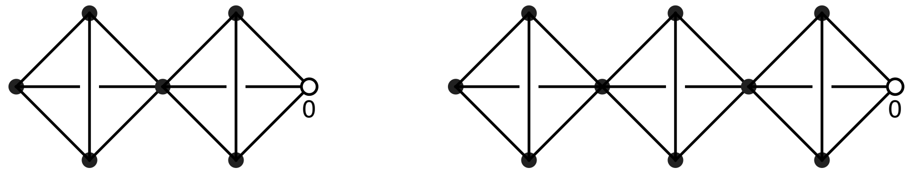

FIGURE 5.1. The graphs T_2 and T_3.

Following the paradigm of [GKP10], we would 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 6.1 that for two 3-valent root-vertices, the number of i-types could be as large as 38. Thus, the potential number of productions 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.

(1) Add the closed path 01230.
(2) Add the path 02.
(3) Add the path 13.
(4) Suppress roots 0, 1, and 3.
(5) Relabel root 2 as root 0.

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 the number of partitions of the integer k.

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 $\begin{pmatrix}2 & 12z & 2z\end{pmatrix}^{tr}$ and the production matrix is*

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

Proof. The initial pgd-vector $\mathbf{V_1}$ for $(K_4, 0)$ and the production matrix can be calculated by hand or by a computer program. $\square$

5.3. **Edge-amalgamation path of copies of K_4.** Here we define $\overline{T}_1$ to be 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 5.2.

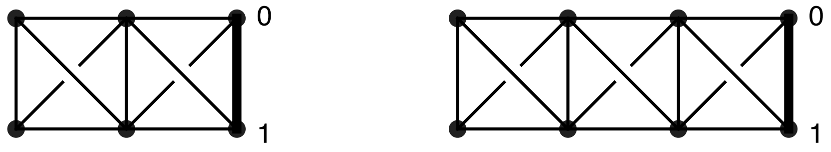

FIGURE 5.2. The graphs $\overline{T}_2$ and $\overline{T}_3$.

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

(1) Add the path 0231.
(2) Add the path 03.
(3) Add the path 12.
(4) Suppress roots 0 and 1.
(5) Relabel root 2 as root 0 and root 3 as root 1.

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 lexicographic 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)

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

$$\overline{\mathbf{V}}(z) \;=\; \begin{pmatrix}2 & 0 & 0 & 0 & 4z & 0 & 2z & 8z & 0 & 0 & 0 & 0\end{pmatrix}^{tr}$$

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*

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

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

If follows that

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

6. Enumerating Possible 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 follows:

structure	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 $\Sigma_{\{1,2\}} \times \Sigma_{\{1,3,4\}}$ 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 (0 2), 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 (0 2) 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)(1 3 4) 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, ..., $k-1$ 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, ..., $k-1$, 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). $\square$

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 $\Sigma_{\{1,2,\dots,n\}}$.

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 the following table:

TABLE 6.1. 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*

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

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 6.2 the maximum number of imbedding-types for triply rooted graphs with root-vertices of valences $1 \le i, j, k \le 5$.

TABLE 6.2. 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

	$j\backslash k$	1	2	3	4	5
	1	90	316	899	2254	5110
	2	316	1284	4074	11214	27556
i=5	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 [GRT89, GMTW15a]). We observe that using Theorem 4.7.2 of [Stan86] could give generating functions for the individual entries of a power of a production matrix.

There are models in the physical sciences to which the transfer matrix method has been applied. Some in chemistry were explored in [KaCh77, KaCh78], where the computational process uses polynomial matrix entries and is called the *polynomial matrix method*. This method was adapted by [BGMP86] for application to matching polynomials of *polygraphs*.

In a sequel [GMT15], 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

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

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

[BGMP86] D. Babic, A. Graovac, B. Mohar, and T. Pisanski, The matching polynomial of a polygraph, *Discrete Appl. Math.* **15** (1986), 11–24.

[ChWe99] T.Y. Chow and J. West, Forbidden subsequences and Chebyshev polynomials, *Discrete Math.* **204** (1999), 119–128.

[GS95] I.M. Gessel and R.P. Stanley, Algebraic enumeration, Chap. 21 of *Handbook of Combinatorics, Vol. 1*, Elsevier and the MIT Press, 1995.

[Gr11a] J.L. Gross, Genus distribution of graph amalgamations: Self-pasting at root-vertices, *Australasian J. Combin.* **49** (2011), 19–38.

[Gr11b] J.L. Gross, Genus distributions of cubic outerplanar graphs, *J. Graph Algorithms and Applications* **15** (2011), 295–316.

[Gr12] J.L. Gross, Embeddings of graphs of fixed treewidth and bounded degree, *Ars Math. Contemporanea* **7** (2014), 423–440. Presented at AMS Annual Meeting at Boston, January 2012.

[Gr13] J.L. Gross, Embeddings of cubic Halin graphs: genus distributions, *Ars Math. Contemporanea* **6** (2013), 37–56.

[GF87] J.L. Gross and M.L. Furst, Hierarchy for imbedding-distribution invariants of a graph, *J. Graph Theory* **11** (1987), 205–220.

[GKP10] 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.

[GKP14] J.L. Gross, I.F. Khan, and M.I. Poshni, Genus distributions for iterated claws, *Electronic J. Combin.* **21** (2014), #P1.12.

[GMT15] J.L. Gross, T. Mansour, and T.W. Tucker, Markovian analysis of production matrices for genus polynomials, in preparation.

[GMTW15a] 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.

[GMTW15b] J.L. Gross, T. Mansour, T.W. Tucker, and D.G.L. Wang, Iterated claws have real-rooted genus polynomials, *Ars Math. Contemporanea*, to appear. Presented at the GEMS Conference, July 2013.

[GRT89] J.L. Gross, D.P. Robbins and T.W. Tucker, Genus distributions for bouquets of circles, *J. Combin. Theory (B)* **47** (1989), 292–306.

[GT87] J.L. Gross and T.W. Tucker, *Topological Graph Theory*, Dover, 2001 (original ed. Wiley, 1987).

[KaCh77] M.V. Kaulgud and V.H. Chitgopkar, Polynomial matrix method for the calculation of π-electron energies for linear conjugated polymers, *J. Chem. Soc. Faraday Trans. II* **73** (1977), 1385–1395.

[KaCh78] 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.

[Mo12] B. Mohar, Genus distribution of path-like and ring-like graphs. Oral presentation at SIAM DM'12 at Halifax, NS, June 2012.

[PKG10] M.I. Poshni, I.F. Khan, and J.L. Gross, Genus distribution of edge-amalgamations, *Ars Math. Contemporanea* **3** (2010), 69–86.

[Stah91] S. Stahl, Permutation-partition pairs III: Embedding distributions of linear families of graphs, *J. Combin. Theory Ser. B* **52** (1991), 191–218.

[Stah97] S. Stahl, On the zeros of some genus polynomials, *Canad. J. Math.* **49** (1997), 617–640.

[Stan86] R.P. Stanley, *Enumerative Combinatorics, Vol. 1*, Wadsworth & Brooks/Cole, 1986.

Dept. of Computer Science, Columbia University, New York, NY 10027, USA;
email: gross@cs.columbia.edu

PUCIT, University of the Punjab, Lahore 54000, Pakistan
email: imran.farid@pucit.edu.pk

Department of Mathematics, University of Haifa, 3498838 Haifa, Israel; email: tmansour@univ.haifa.ac.il

Dept. of Mathematics, Colgate University, Hamilton, NY 13346, USA;
email: ttucker@colgate.edu