Genus Distributions of 4-Regular Outerplanar Graphs

Mehvish I. Poshni

Department of Computer Science
Columbia University, New York, NY 10027, USA

poshni@cs.columbia.edu

Imran F. Khan

Department of Computer Science
Columbia University, New York, NY 10027, USA

imran@cs.columbia.edu

Jonathan L. Gross

Department of Computer Science
Columbia University, New York, NY 10027, USA

gross@cs.columbia.edu

Submitted: Aug 9, 2011; Accepted: Oct 8, 2011; Published: Oct 31, 2011
Mathematics Subject Classification: 05C10

Abstract

We present an $O(n^2)$-time algorithm for calculating the genus distribution of any 4-regular outerplanar graph. We characterize such graphs in terms of what we call *split graphs* and *incidence trees*. The algorithm uses post-order traversal of the incidence tree and *productions* that are adapted from a previous paper that analyzes double-root *vertex-amalgamations* and *self-amalgamations*.

1 Introduction

A graph G is called an ***outerplanar graph*** if it has a planar embedding in which some face-boundary walk contains every vertex of G. We refer to such an embedding as an ***outerplane embedding***, and we denote the face containing all the vertices by f_∞ to indicate that it contains the point at infinity. An outerplane embedding is said to be ***normalized*** if all self-loops of the graph lie on the face-boundary walk of the face f_∞. We designate the edges that constitute the face-boundary walk of f_∞ as ***exterior edges***, in contrast to the usage of ***interior edges*** for the remaining edges. Figure 1.1 shows a 4-regular outerplane embedding before normalization, with the exterior edges shown darker than interior edges.

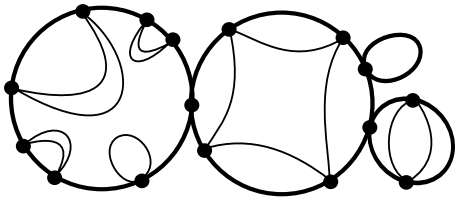

Figure 1.1: An unnormalized outerplane embedding of a 4-regular outerplanar graph.

Calculating genus distributions for families of graphs is a classical problem in topological graph theory. This paper describes a polynomial-time algorithm for calculating the genus distribution of any 4-regular outerplanar graph. One special importance of 4-regular graphs is that they occur as projections of knots and links. Another is that the medial graph of any embedded graph is a 4-regular graph. In the remainder of this section, we review some standard terminology and notation, as well as prior work in this area. In §2 and §3, we lay groundwork for exploiting the structure of 4-regular outerplanar graphs for our present purpose. In §4, we discuss the algorithm, and we do a dry run on a small example. In §5, we discuss the complexity of the algorithm. In §6, we give the proof of correctness. In §7, we make some concluding remarks.

In this paper, we assume some familiarity with topological graph theory (see [GrTu87] or [Wh01]). We permit graphs to have self-loops and multiple adjacencies. When regarding an edge as a topological space homeomorphic to a curve in the space, an ***edge-end*** is the small local part of the edge that begins at its endpoint and ends well short of its midpoint. Each edge of the graph has two edge-ends, even if it is a self-loop. We refer to the edge-ends of exterior and interior edges as ***exterior edge-ends*** and ***interior edge-ends***, respectively. The cyclic permutation of edge-ends at a vertex v is known as a ***rotation*** at v. A ***rotation system*** of graph G is a set containing one rotation for every vertex of G. It is well known that there is a bijective correspondence between the rotations systems of a graph G and the orientable embeddings of G. Moreover, each rotation system of a graph G specifies an embedding of G. We denote the orientable surface of genus i by S_i and the number of cellular embeddings of a graph G on the surface S_i by $g_i(G)$. The sequence $\{g_i(G) : i \geq 0\}$, is known as the ***genus distribution*** of the graph G. An embedding of a graph G on a surface is assumed to be cellular and orientable and is generically denoted by ι_G. We abbreviate face-boundary walk as ***fb-walk***.

Any vertex in a graph may be designated a ***root vertex***. A graph with one or more root vertices is known as a ***rooted graph***. In this paper, we primarily deal with graphs having two roots. We refer to such a graph as a ***double-rooted graph***. For the purpose of this paper, we assume that each root vertex in a double-rooted graph is 2-valent. If a 2-valent root vertex u occurs twice in an fb-walk, it breaks the fb-walk into two ***strands***, which are the maximal subwalks such that u is not an interior point. We refer to these strands as u***-strands***. For a double-rooted graph (G, u, v), the vertex u is referred to as the ***first-root*** of the graph G and the vertex v is referred to as the ***second-root*** of the graph G.

An enumerative concern, related to calculating genus distributions, has been to count the number of embeddings of a graph in a surface of minimum-genus. Some recent papers in this regard include [BGGS00], [GoRiSi07], [GrGr08], and [KoVo02]. Genus distributions were first studied in [GrFu87], [FuGrSt89], and [GrRoTu89]. Prior work on genus distributions has been largely focused on graph families with high symmetries. In this context, prior work on counting embeddings in all orientable surfaces or in all surfaces includes [ChLiWa06], [KwLe93], [KwLe94], [KwSh02], [McG87], [Mu99], [St90], [St91a], [St91b], [Tesa00], [ViWi07], [WaLi06], and [WaLi08]. The well known Heffter-Edmonds algorithm calculates the genus distribution of any graph, but its time-complexity is super-exponential in the size of the graph (see [GrTu87]).

We call attention to a recent series of related papers that provide foundational techniques for calculating genus distributions of graphs that are produced from other more *well-understood* graphs by performing various operations on them. These include the publications [GKP10], [Gr10], [PKG10], [KPG10], [Gr11a], and [PKG11]. An innovative use of these techniques appears in [Gr11b], where a polynomial-time algorithm for calculating the genus distribution of cubic outerplanar graphs is described. The results in this paper also serve to demonstrate the power of these techniques. This paper is predominantly self-contained, notwithstanding the use of results developed in [Gr11a].

2 Split Graphs and Incidence Trees

Given a normalized outerplane embedding of a 4-regular outerplanar graph G, we classify its vertices into two types. A ***Type-I vertex*** has two exterior and two interior incident edge-ends, whereas a ***Type-II vertex*** has four exterior incident edge-ends. Thus, every cut-vertex is a Type-II vertex. Moreover, by requiring that the outerplane embedding be normalized, we ensure that every self-loop lies on the fb-walk of the face f_∞ and, therefore, render its single endpoint a Type-II vertex. All other vertices are Type-I. The general term ***splitting of a vertex*** v_i is used to mean either of the following two operations on the vertex v_i:

- **Type-I Vertices:** In the rotation at a Type-I vertex v_i in an outerplane embedding, the exterior edge-ends e_1 and e_2 incident on v_i are contiguous, as are the interior edge-ends d_1 and d_2. Let the cyclic counter-clockwise order of the edge-ends incident on v_i be (e_1, e_2, d_1, d_2) in the outerplane embedding of graph G. Then splitting the vertex v_i consists of introducing two new vertices v_i' and v_i'', called ***single-primed*** and ***double-primed vertices***, respectively, with the edge-ends d_2 and e_1 incident on v_i' instead of on v_i, and with the edge-ends e_2 and d_1 incident on v_i'' instead of on v_i. The vertex v_i is deleted. This is illustrated in Figure 2.1.

- **Type-II Vertices:** Let the exterior edge-ends of v_i be cyclically ordered as (e_1, e_2, e_3, e_4), where e_1 and e_4 belong to one block and e_2 and e_3 to another. Then splitting the vertex v_i consists of introducing two new vertices $\dot{v}_i$ and $\ddot{v}_i$. We refer to either of these as a ***dotted vertex***. The edge-ends e_1 and e_4 are made incident

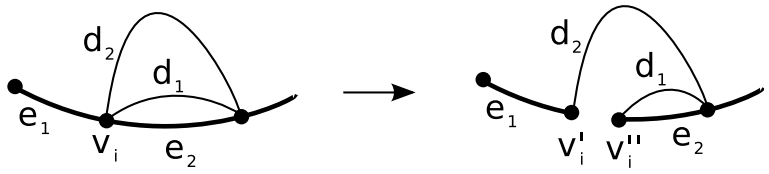

Figure 2.1: Splitting a Type-I vertex v_i.

on $\dot{v}_i$, while the edge-ends e_2 and e_3 are made incident on $\ddot{v}_i$ instead of on v_i. The vertex v_i is deleted.

In this manner, we may split every vertex of the normalized outerplane embedding of a 4-regular outerplanar graph G, thereby obtaining a graph G' where each vertex is 2-valent and, therefore, each component is a cycle C_n for some n. We refer to G' as the ***split graph*** for the graph G, and we refer to each pair of vertices obtained from a split as ***coupled vertices***. The two vertices in a coupled pair belong to different components. These two components are called ***coupled components*** with respect to that pair of vertices. An example of a 4-regular outerplanar graph and its split graph is shown in Figure 2.2.

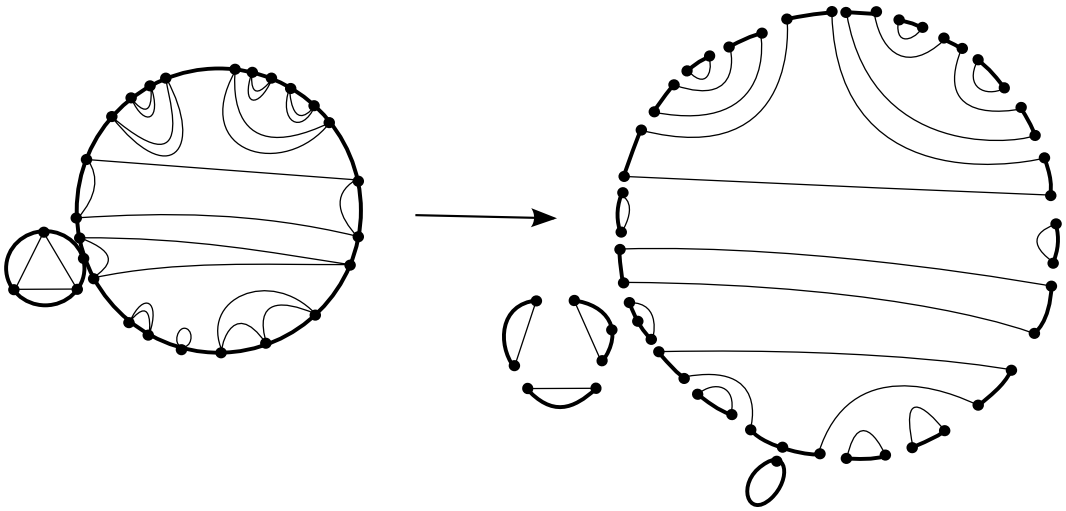

Figure 2.2: A 4-regular outerplanar graph and the split graph obtained from its normalized outerplane embedding.

Remark Each component of a split graph is the boundary of a 2-cell, which is regarded as having a counter-clockwise orientation induced from the orientation of the outerplane embedding.

It is easy to visualize how the original graph G can be reassembled by amalgamating each pair of coupled vertices in G'. As we will see, our algorithm utilizes this reconstructability of a 4-regular outerplanar graph. It calculates genus distribution of the outerplanar graph by simulating its reconstruction, while calculating the genus distributions for the subgraphs assembled at each step of the algorithm.

By the ***component graph of the split graph*** $C(G')$, we mean the graph whose nodes are the components of the split graph, and in which two nodes are adjacent if they are coupled. We describe an algorithm to build an *ordered tree* that can be regarded as a depth-first spanning tree of $C(G')$:

1. Designate an arbitrary component in the component graph $C(G')$ as the ***root node*** of the tree. Represent the root node visually with a round-shaped vertex.

2. Construct a depth-first ordered tree rooted at the root node in the component graph $C(G')$, such that the child components for each tree node $\mathcal{C}$ correspond to the components coupled with it only with respect to its single-primed and dotted vertices.

3. By ordered tree, we mean that the counter-clockwise rotation at each tree node imposes a linear ordering on its children. The order prescribed for the children of each tree node $\mathcal{C}$ is that in which these coupled child components are encountered under the counter-clockwise orientation on $\mathcal{C}$ in G'. The first child node of the root node is chosen arbitrarily since the root node has no parent node. In contrast, for any other tree node $\mathcal{C}$ coupled with parent node $\mathcal{P}$, the first child node of $\mathcal{C}$ is chosen to be the first component coupled with it after $\mathcal{P}$ under the counter-clockwise orientation on $\mathcal{C}$.

4. Each new node added to the tree in step 2 is represented visually by a ***square*** node if it corresponds to a component coupled to its parent with respect to dotted vertices, otherwise it is represented by a ***round*** node.

The ordered tree formed in this manner is unique for a fixed root and a fixed first child of the root, and is referred to as the ***incidence tree*** of the outerplanar graph G with respect to the given outerplane embedding. Depending on the context, we may regard the tree nodes of an incidence tree, interchangeably, either as the components of $C(G')$ or as their more abstract round and square visual representations. For the split graph in Figure 2.2 and a particular choice of the root component and the first child component, the corresponding incidence tree is shown in Figure 2.3. The darker round node 18 shown in Figure 2.3 is the root node. The arbitrarily selected first child of the root is illustrated by the dark directed edge incident on it from the root node.

The post-order traversal of an incidence tree prescribes the order in which coupled vertices are amalgamated when simulating the reconstruction of the outerplanar graph. In this sense, the incidence tree for a 4-regular outerplanar graph fills the same role as the "inner tree" for a 3-regular outerplanar graph in [Gr11b]. However, as we have just seen, its construction involves more subtleties.

Remark The purpose in introducing component graphs is to facilitate the conceptualization of incidence trees. In practice, an incidence tree can be constructed directly from a split graph without recourse to construction of the component graph.

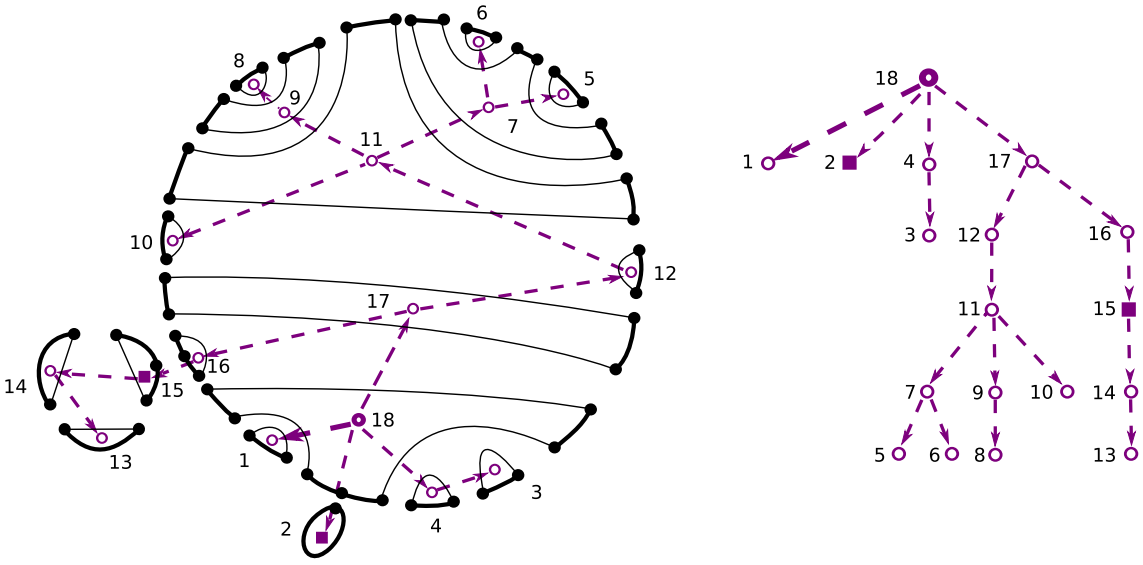

Figure 2.3: An incidence tree for the split graph from Figure 2.2.

3 Amalgamations and Self-Amalgamations

In order to simulate the reconstruction of a 4-regular outerplanar graph, we require two graph operations, known as *amalgamation* and *self-amalgamation*. We employ constructs known as *productions* to model the effect of using these two operations on graphs. In both cases, a production algebraically represents the embeddings of the resulting graph. A production for double-rooted graphs is defined in terms of a concept called a *double-root partial*. Accordingly, we introduce this concept before we proceed to define these two operations and their corresponding productions.

Double-Root Partials

We observe that any given 2-valent vertex appears exactly twice in the set of fb-walks of an embedding. This enables us to partition the embeddings of a double-rooted graph (G, u, v) on a surface S_i into the four basic types: dd_i, ds_i, sd_i, and ss_i. The first letter of each type represents the first-root u, and the second letter represents the second-root v. The letters s and d are mnemonics for "same" and "distinct", indicating whether the corresponding 2-valent root vertex occurs twice on the same fb-walk or once on each of two distinct fb-walks. Each ***double-root partial*** counts the number of embeddings of one of these four basic types. The four double-root partials are further refined by [GKP10] to express the specific relationships of the fb-walks incident on both roots. These refinements are known as sub-partials, and are as follows:
The sub-partials of type dd_i are:

$$dd_i^0(G, u, v) \quad = \quad \text{the number of embeddings of type } dd_i \text{ such that neither of the fb-walks incident on } u \text{ is incident on } v.$$

$$dd_i'(G,u,v) = \text{the number of embeddings of type } dd_i \text{ such that exactly one fb-walk incident on } u \text{ is incident on } v.$$

$$dd_i''(G,u,v) = \text{the number of embeddings of type } dd_i \text{ such that both fb-walks incident on } u \text{ are incident on } v.$$

Similarly, the sub-partials of type ds_i and of type sd_i are as follows:

$$ds_i^0(G,u,v) = \text{the number of embeddings of type } ds_i \text{ such that neither fb-walk incident on } u \text{ is incident on } v.$$

$$ds_i'(G,u,v) = \text{the number of embeddings of type } ds_i \text{ such that exactly one fb-walk incident on } u \text{ is incident on } v.$$

$$sd_i^0(G,u,v) = \text{the number of embeddings of type } sd_i \text{ such that the fb-walk incident on } u \text{ is not incident on } v.$$

$$sd_i'(G,u,v) = \text{the number of embeddings of type } sd_i \text{ such that the fb-walk incident on } u \text{ is also incident on } v.$$

Finally, the sub-partials of type ss_i are as follows:

$$ss_i^0(G,u,v) = \text{the number of embeddings of type } ss_i \text{ such that the fb-walk incident on } u \text{ is not incident on } v.$$

$$ss_i^1(G,u,v) = \text{the number of embeddings of type } ss_i \text{ such that one } u\text{-strand of the fb-walk incident on } u \text{ contains both occurrences of } v.$$

$$ss_i^2(G,u,v) = \text{the number of embeddings of type } ss_i \text{ such that both } u\text{-strands of the fb-walk incident on } u \text{ contain an occurrence of } v.$$

There are also additional sub-partials that are refinements for the sub-partials of types sd' and ss^1. The definitions for these sub-partials and the context in which they are needed is discussed later.

The collection of values of all sub-partials, for all values of i, is known as a ***partitioned genus distribution*** of the graph (G,u,v).

Productions for Self-Amalgamation

A ***self-amalgamation*** of a double-rooted graph is an operation $(G,u,v) \longrightarrow W$, where the two roots of the graph are merged together to produce a new graph. We may alternatively use the terminology ***self-pasting*** to mean the same. We know from [Gr11a] that when both roots u and v are 2-valent, an embedding ι_G of G under self-amalgamation

induces six unique embeddings of W such that the rotations at vertices in ι_G are consistent with rotations at vertices in the six corresponding embeddings of W. We also know that the genus of each of these embeddings of W is a function of the genus of the embedding surface of ι_G and of the configuration of fb-walks on which the roots of G lie. This information can be represented in a form known as a ***production***.

Let p_i be a double-root sub-partial of the double-rooted graph (G, u, v). Then the standard representation for a self-amalgamation production, as laid out in [Gr11a], is of the form:

$$p_i(G, u, v) \longrightarrow \alpha_1 g_{i+k_1}(W) + \alpha_2 g_{i+k_2}(W)$$

where α_1, α_2 are non-negative integers whose sum is 6, and where k_1, k_2 are integers within the range of -1 to 2. This can be interpreted as follows:

> A type p embedding of (G, u, v) on surface S_i self-amalgamates on the root-vertices u and v to give six embeddings of the graph W. Out of these six resulting embeddings, α_1 embeddings are on surface S_{i+k_1}, and α_2 are on surface S_{i+k_2}.

The left-hand-side of a production is known as the ***production head***. The right-hand-side of a production is known as the ***production body***.

The complete set of productions for self-amalgamation on 2-valent roots is given in [Gr11a]. However, the form of the production defined above does not capture root-related information for the graph W that is produced as a result of the self-pasting. In our algorithm, we need to be able to repeatedly apply self-amalgamations and vertex-amalgamations, in order to build a larger graph from many of the smaller subgraphs. For this reason, after self-amalgamation, we ***pop*** new root vertices on the *exterior* edge e incident on the first-root u of the graph (G, u, v). This is illustrated in Figure 3.1, where the edges e and f are incident on the first-root u before self-amalgamation. The edge f is necessarily an interior edge. New roots are popped on the edge e after self-amalgamation. *Nota bene*, the root popped closer to the amalgamated vertex is considered the second-root of the resulting graph.

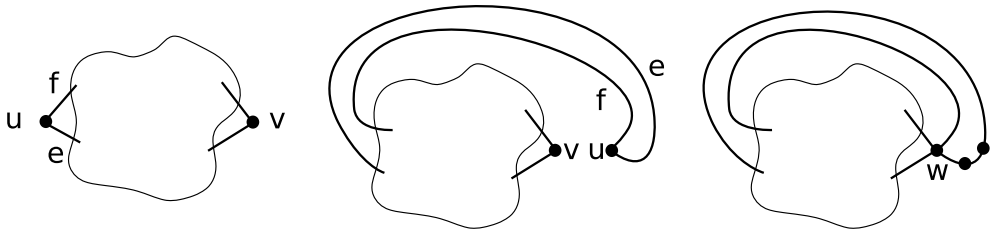

Figure 3.1: A model representing self-amalgamation.

This entails adapting the production body to reflect the new roots. In particular, we need to replace each occurrence of g_i in the production body by the relevant double-root

sub-partial type that is consistent with the face-boundaries incident on these new roots. In light of this, we redefine a production for self-amalgamation as follows:

$$p_i(G,u,v) \longrightarrow \sum_{\substack{x_k \text{ ranges over all} \\ \text{sub-partial types} \\ \text{with } k\in\{i-1,i,i+1,i+2\}}} \alpha_{x_k} x_k(W,s,v)$$

where each x_k is a double-root sub-partial type for the graph W and where the numbers α_{x_k} are non-negative integers whose sum is 6.

Since both new roots are popped on the same edge, the same fb-walk that passes through one root also passes through the other. Thus, each sub-partial type in the production body for a self-amalgamation production is either of type dd'' or of type ss^1.

Adaptation of productions in this manner is straightforward for all sub-partials p_i in the production head, except for the sub-partials sd_i' and ss_i^1. To facilitate the adaptation of productions for these two sub-partials, we further refine them as follows:

$\uparrow sd_i'(G,u,v)$ = the number of embeddings of type sd_i' such that the u-strand that contains the occurrence of vertex v also contains both occurrences of exterior edge e in it (see Figure 3.2).

$\downarrow sd_i'(G,u,v)$ = the number of embeddings of type sd_i' such that the u-strand that contains the occurrence of vertex v does not contain the two occurrences of exterior edge e in it (see Figure 3.2).

Therefore,

$$sd_i'(G,u,v) = \uparrow sd_i'(G,u,v) + \downarrow sd_i'(G,u,v)$$

Similarly,

$\uparrow ss_i{}^1(G,u,v)$ = the number of embeddings of type ss_i^1 such that the u-strand that contains both occurrences of the vertex v also contains both occurrences of exterior edge e in it (see Figure 3.2).

$\downarrow ss_i{}^1(G,u,v)$ = the number of embeddings of type ss_i^1 such that the u-strand that contains both occurrences of the vertex v does not contain the two occurrences of exterior edge e in it (see Figure 3.2).

Thus,

$$ss_i^1(G,u,v) = \uparrow ss_i{}^1(G,u,v) + \downarrow ss_i{}^1(G,u,v)$$

We now adapt the proofs in [Gr11a] by popping two new roots on edge e, as shown on the right side of Figure 3.1. This chosen edge e corresponds to the exterior edge of the outerplane embedding that is incident on the first-root undergoing self-amalgamation.

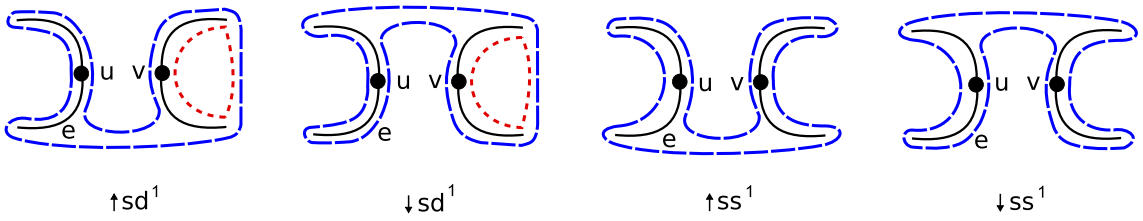

Figure 3.2: Refined partials types of sd' and ss^1.

The following theorem adapts the productions for self-amalgamation derived in [Gr11a] by making the modification above.

Theorem 3.1 *When an embedding of a double-rooted graph (G, s, t) with 2-valent roots is self-amalgamated, the following productions hold:*

$$dd_i^0(G,u,v) \longrightarrow 4dd''_{i+1}(W,s,t) + 2{\downarrow}ss^1_{i+2}(W,s,t) \tag{3.1}$$
$$dd'_i(G,u,v) \longrightarrow dd''_i(W,s,t) + 3dd''_{i+1}(W,s,t) + 2{\downarrow}ss^1_{i+1}(W,s,t) \tag{3.2}$$
$$dd''_i(G,u,v) \longrightarrow 4dd''_i(W,s,t) + 2{\downarrow}ss^1_{i+1}(W,s,t) \tag{3.3}$$
$$ds_i^0(G,u,v) \longrightarrow 6dd''_{i+1}(W,s,t) \tag{3.4}$$
$$ds'_i(G,u,v) \longrightarrow 3dd''_i(W,s,t) + 3{\downarrow}ss^1_{i+1}(W,s,t) \tag{3.5}$$
$$sd_i^0(G,u,v) \longrightarrow 6{\downarrow}ss^1_{i+1}(W,s,t) \tag{3.6}$$
$${\uparrow}sd'_i(G,u,v) \longrightarrow 3dd''_i(W,s,t) + 3{\downarrow}ss^1_{i+1}(W,s,t) \tag{3.7}$$
$${\downarrow}sd'_i(G,u,v) \longrightarrow 3{\downarrow}ss^1_i(W,s,t) + 3{\downarrow}ss^1_{i+1}(W,s,t) \tag{3.8}$$
$$ss_i^0(G,u,v) \longrightarrow 6{\downarrow}ss^1_{i+1}(W,s,t) \tag{3.9}$$
$${\uparrow}ss_i^1(G,u,v) \longrightarrow 6dd''_i(W,s,t) \tag{3.10}$$
$${\downarrow}ss_i^1(G,u,v) \longrightarrow 6{\downarrow}ss^1_i(W,s,t) \tag{3.11}$$
$$ss_i^2(G,u,v) \longrightarrow dd''_{i-1}(W,s,t) + 3dd''_i(W,s,t) + 2{\downarrow}ss^1_i(W,s,t) \tag{3.12}$$

Proof An sd'-type embedding of (G,u,v) has one fb-walk incident on root u and two on root v. Moreover, the fb-walk incident on u is also incident on root v. When such an embedding is self-amalgamated, the resulting graph W has six corresponding embeddings. This however results in two different scenarios based on whether the embedding of G is of sub-type ${\uparrow}sd'_i$ or ${\downarrow}sd'_i$. The first scenario, corresponding to an ${\uparrow}sd'_i$-type embedding of (G,u,v), is portrayed in Figure 3.3.

The six embedding models shown at the right of the figure correspond to the embeddings of W resulting from the self-amalgamation of an embedding of G. As a result of self-amalgamation, the fb-walks incident on both root vertices of (G,u,v) break into strands, that recombine to make new fb-walks. Two new roots are popped on the exterior edge e after self-amalgamation, as shown in the figure. The root farther from the amalgamated vertex is the first-root, and the one closer to it is the second-root. One observes that half of the embeddings of W resulting from self-amalgamation are of type dd'', while the remaining are of type ${\downarrow}ss^1$. This accounts for Production 3.7.

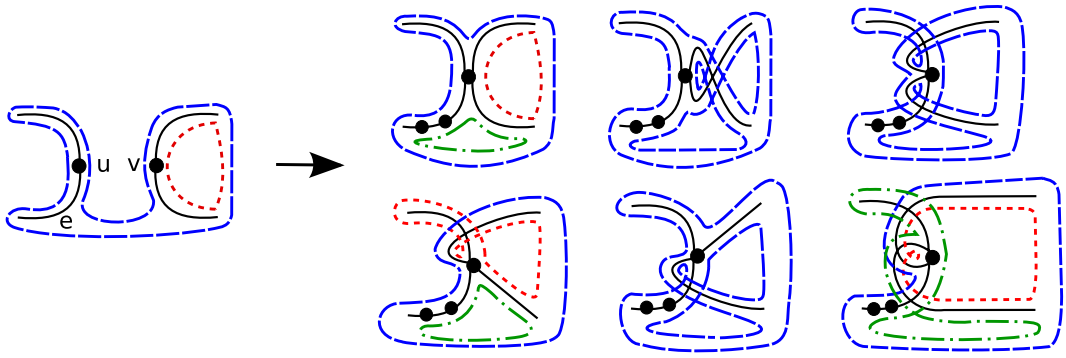

Figure 3.3: Self-amalgamation of a $\uparrow sd_i$-type embedding of G.

Contrast this with the second scenario illustrated in Figure 3.4. This constitutes the proof of Production 3.8.

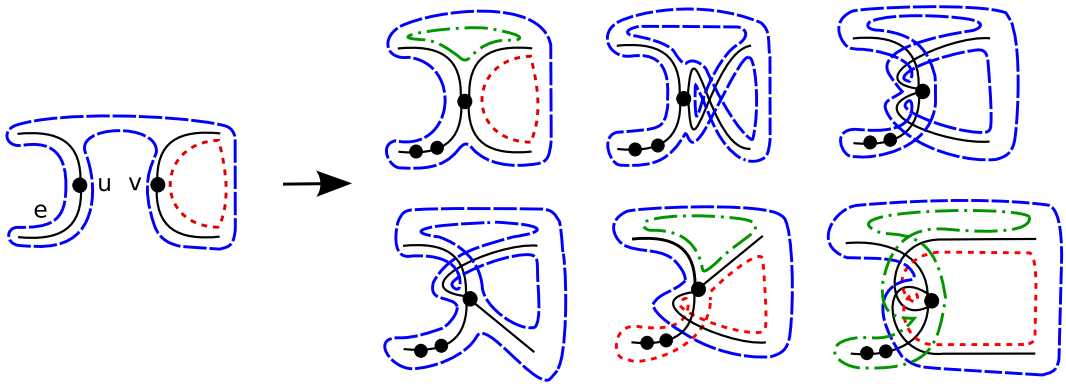

Figure 3.4: Self-amalgamation of a $\downarrow sd_i$-type embedding of G.

REMARK Figure 3.1 makes it clear that the ss^1-type partials resulting from the self-amalgamation are always $\downarrow ss^1$-sub-type.

The proofs for other productions are identical in substance to the proofs given for the corresponding productions in [Gr11a]. However, a fine-tuning of the classification of the embeddings resulting from self-amalgamation is necessitated, as in the proof of the productions above. For the sake of brevity, we leave the remaining productions for the reader to verify. ♢

Productions for Vertex-Amalgamation

Let (G, s, t) be a graph with the vertices s and t designated as roots, and let (H, u, v) be a graph with the vertices u and v as roots. Then amalgamating the graph G at root vertex t with the graph H at root vertex u yields a new graph (W, s, v) with the vertices s and v serving as roots. We denote the amalgamation operation by an asterisk as follows:

$$(W, s, v) \;=\; (G, s, t) * (H, u, v)$$

As in [GKP10], we assume 2-valent roots. Thus, when an embedding ι_G of G and an embedding ι_H of H amalgamate, they induce six unique embeddings of W, in which the rotations at all vertices of W are consistent with the rotations at the corresponding vertices in both ι_G and ι_H. Moreover, the genus of each of these embeddings of W is a function of the genera of ι_G and ι_H and of the fb-walks on which the roots of G and H lie as they undergo amalgamation.

Let p_i and q_j be double-root sub-partials. Then, a production is used to represent the ways in which a p-type embedding of (G, s, t) and a q-type embedding of (H, u, v) amalgamate on their root vertices t and u, respectively, to give various types of embeddings of the resulting graph (W, s, v). We write

$$p_i(G, s, t) * q_j(H, u, v) \longrightarrow \sum_{\substack{x_k \text{ ranges over all} \\ \text{sub-partial types} \\ \text{with } k \in \{i+j, i+j+1\}}} \alpha_{x_k} x_k(W, s, v)$$

where the coefficients α_{x_k} are non-negative integers that sum to six, and where each term in the production body indicates that there are α_{x_k} embeddings of the graph produced by the amalgamation, that have genus k and a sub-partial type x_k. This can be read as follows:

> Amalgamating a p-type embedding of (G, s, t) on surface S_i with a q-type embedding of (H, u, v) on surface S_j on the root-vertices t and u yields six embeddings of the graph (W, s, v). Each of these six embeddings corresponds to a partial type x on the surface S_{i+j} or S_{i+j+1}, as specified by the subscript of x.

A method for deriving productions for vertex-amalgamation was presented in [Gr11a], but no distinction was made between the $\uparrow ss^1$ and $\downarrow ss^1$ sub-partials, or between the $\uparrow sd'$ and $\downarrow sd'$ sub-partials. The method in [Gr11a] works equally well for these new sub-partials. The complete list of productions needed for our algorithm is given in Table 3.1. The productions not involving sub-partial types $\uparrow sd'$, $\downarrow sd'$, $\uparrow ss^1$ or $\downarrow ss^1$ in the production body are taken from [Gr11a] and are listed here only for the sake of completion. For brevity, we abbreviate the double-root partials by omitting the double-rooted graphs.

Even though there are twelve sub-partials defined in this paper, the number of productions directly needed for our algorithm is $2 \times 12 = 24$. This is because the order in which the various graph components are amalgamated necessitates that the roots of the first amalgamand in any vertex-amalgamation be adjacent. This allows three possibilities for the sub-partial types of such a component: dd'', $\uparrow ss^1$, and $\downarrow ss^1$. It turns out that an embedding of the first amalgamand is never of type $\uparrow ss^1$. The first amalgamand has an ss^1-type embedding only as an outcome of a previous self-amalgamation or as an outcome of a step in our algorithm that involves vertex-amalgamating a pair of dotted vertices. In our earlier remark, we mentioned that self-amalgamation produces only $\downarrow ss^1$-type embeddings. The same is also true for the latter scenario as will become evident in the next section. Therefore, the sub-partials of the first amalgamand are limited to only two valid types: dd'' and $\downarrow ss^1$.

Table 3.1: Productions for vertex-amalgamation $(G, s, t) * (H, u, v)$ where the embedding of graph G has partial type dd''.

$dd_i''(G, s, t)$ *productions*	$\downarrow ss_i^1(G, s, t)$ *productions*
$dd_i'' * dd_j^0 \longrightarrow 4dd_{i+j}^0 + 2sd_{i+j+1}^0$	$\downarrow ss_i^1 * dd_j^0 \longrightarrow 6sd_{i+j}^0$
$dd_i'' * dd_j' \longrightarrow 2dd_{i+j}^0 + 2dd_{i+j}' + \downarrow sd_{i+j+1}' + \uparrow sd_{i+j+1}'$	$\downarrow ss_i^1 * dd_j' \longrightarrow 3\downarrow sd_{i+j}' + 3sd_{i+j}^0$
$dd_i'' * dd_j'' \longrightarrow 4dd_{i+j}' + 2ss_{i+j+1}^2$	$\downarrow ss_i^1 * dd_j'' \longrightarrow 6\downarrow sd_{i+j}''$
$dd_i'' * ds_j^0 \longrightarrow 4ds_{i+j}^0 + 2ss_{i+j+1}^0$	$\downarrow ss_i^1 * ds_j^0 \longrightarrow 6ss_{i+j+1}^0$
$dd_i'' * ds_j' \longrightarrow 2ds_{i+j}^0 + 2ds_{i+j}' + \downarrow ss_{i+j+1}^1 + \uparrow ss_{i+j+1}^1$	$\downarrow ss_i^1 * ds_j' \longrightarrow 3ss_{i+j}^0 + 3\downarrow ss_{i+j}^1$
$dd_i'' * sd_j^0 \longrightarrow 6dd_{i+j}^0$	$\downarrow ss_i^1 * sd_j^0 \longrightarrow 6\downarrow sd_{i+j}'$
$dd_i'' * \downarrow sd_j' \longrightarrow 6dd_{i+j}'$	$\downarrow ss_i^1 * \uparrow sd_j' \longrightarrow 6\downarrow sd_{i+j}'$
$dd_i'' * \uparrow sd_j' \longrightarrow 6dd_{i+j}'$	$\downarrow ss_i^1 * \downarrow sd_j' \longrightarrow 6\downarrow sd_{i+j}'$
$dd_i'' * ss_j^0 \longrightarrow 6ds_{i+j}^0$	$\downarrow ss_i^1 * ss_j^0 \longrightarrow 6ss_{i+j}^0$
$dd_i'' * \uparrow ss_j^1 \longrightarrow 6ds_{i+j}'$	$\downarrow ss_i^1 * \uparrow ss_j^1 \longrightarrow 6\downarrow ss_{i+j}^1$
$dd_i'' * \downarrow ss_j^1 \longrightarrow 6ds_{i+j}'$	$\downarrow ss_i^1 * \downarrow ss_j^1 \longrightarrow 6\downarrow ss_{i+j}^1$
$dd_i'' * ss_j^2 \longrightarrow 4ds_{i+j}' + 2dd_{i+j}''$	$\downarrow ss_i^1 * ss_j^2 \longrightarrow 6\downarrow ss_{i+j}^1$

4 Algorithm

This section describes the algorithm that calculates the genus distribution of a 4-regular n-vertex outerplanar graph in $O(n^2)$ time. The later part of this section also demonstrates how the algorithm works by illustrating it for a simple example.

Input: A rotation system that specifies an outerplane embedding of a 4-regular outerplanar graph G.

Algorithm:

1. Normalize the outerplane embedding by changing rotations of all vertices that have a self-loop incident on them and by making the self-loops lie on the boundary of the face f_∞.

2. Obtain the split graph G' from the normalized outerplane embedding, and form an incidence tree T with respect to an arbitrarily designated root component and an arbitrarily chosen first child of the root component. At the outset, the only non-zero double-root sub-partial for each component of the split graph G' is $dd_0'' = 1$. As we see in an example developed in this section, splitting the base vertex of a self-loop leads to a component of G' with only one vertex and one edge. However, we pop a new root vertex adjacent to that one vertex and regard that component as also

having two roots and as having the double-root partial $dd''_0 = 1$, thereby avoiding exceptional handling of this case.

3. Perform a post-order traversal of the incidence tree T and *process* all the nodes of T in that order. *Processing* each node requires a vertex-amalgamation, a self-amalgamation, or both operations on its associated component, in addition to certain other actions. When performing a vertex-amalgamation or a self-amalgamation, we calculate the double-root sub-partials for the resulting subgraph by applying the relevant productions to the non-zero double-root sub-partials of the components involved in the operation.

   We elaborate on how to ***process a node*** based on its type:

   (a) Processing a round node of T requires two steps:

      i. First the component associated with the round node undergoes vertex-amalgamation on its first-root with the component associated with its parent node in the incidence tree.

      ii. After the vertex-amalgamation, check whether the vertex coupled with the second-root of the component belongs to a different component or to the same component. If it is the same component, perform a self-amalgamation.

   (b) Processing a square node simulates the amalgamation of coupled vertices that were initially produced by splitting a Type-II vertex. Let $\mathcal{P}$ be the component associated with the parent node of a square node, and let $\mathcal{S}$ be the component associated with the square node. Then processing the square node involves the following steps:

      i. First the component $\mathcal{P}$ is vertex-amalgamated on its second-root to the component $\mathcal{S}$. The resulting graph has the first-root on what was previously the component $\mathcal{P}$, while the second-root is on what was previously the component $\mathcal{S}$. There are no further amalgamations to be performed on the subgraph $\mathcal{S}$, whereas we still need two root vertices on the subgraph $\mathcal{P}$ in order to process the parent node (or the sibling node) of the square node in the post-order traversal of the incidence tree. This necessitates that we drop the second-root and pop a new root vertex adjacent to the first-root. Depending on whether the first-root lies in a type d or a type s embedding, the two new roots will now be in a type dd'' or in a type $\downarrow ss^1$ embedding, respectively. This explains our next step.

      ii. All dd_i and ds_i partials for the graph produced in the previous step are added and saved as dd''_i for each i, and all sd_i and ss_i partials for each i are added and saved as $\downarrow ss^1_i$. Other than these two sub-partials, all other sub-partials are made zero-valued.

4. Once the entire incidence tree has been processed, the values of sub-partials constitute the partitioned genus distribution of the given graph G. The genus distribution

can now be calculated by summing all non-zero double-root sub-partials for each i, i.e.,

$$g_i(G) \;=\; \sum_{\substack{x_i \text{ ranges over all} \\ \text{sub-partials}}} x_i(G,u,v)$$

Working Out an Example

We simulate the algorithm on a simple example of a 4-regular outerplanar graph, shown in Figure 4.1. The split graph and its corresponding incidence tree for an arbitrarily chosen root component are also shown in Figure 4.1. For ease of referencing, we have labeled the components of the split graph with letters of the alphabet.

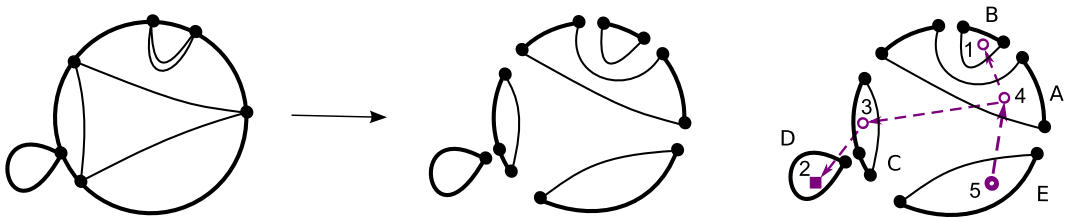

Figure 4.1: Graph G, its split graph and incidence tree.

1. Processing tree node 1 involves a vertex-amalgamation of components A and B, followed by a self-amalgamation. We refer to the subgraph obtained as a result of the vertex-amalgamation as U_1, and to the subgraph resulting from the self-amalgamation of U_1 as U_2.

   (a) Since $dd''_0(A) = 1$ and $dd''_0(B) = 1$ are the only non-zero sub-partials of components A and B, there is only one applicable production for vertex-amalgamation:

   $$dd''_i(A) * dd''_j(B) \longrightarrow 4dd'_{i+j}(U_1) + 2ss^2_{i+j+1}(U_1)$$

   $\Longrightarrow$

   $$\begin{aligned} dd'_k(U_1) &= 4dd''_k(A) \times dd''_0(B) = 4dd''_k(A) \times 1 = 4dd''_k(A) \\ ss^2_k(U_1) &= 2dd''_{k-1}(A) \times dd''_0(B) = 2dd''_{k-1}(A) \times 1 = 2dd''_{k-1}(A) \end{aligned}$$

   $\Longrightarrow$

   $$\begin{aligned} dd'_0(U_1) &= 4dd''_0(A) = 4 \times 1 = 4 \\ ss^2_1(U_1) &= 2dd''_0(A) = 2 \times 1 = 2 \end{aligned}$$

(b) For self-amalgamation of U_1, we need Productions 3.2 and 3.12:

$$\begin{aligned} dd'_i(U_1) &\longrightarrow dd''_i(U_2) + 3dd''_{i+1}(U_2) + 2{\downarrow}ss^1_{i+1}(U_2) \\ ss^2_i(U_1) &\longrightarrow dd''_{i-1}(U_2) + 3dd''_i(U_2) + 2{\downarrow}ss^1_i(U_2) \end{aligned}$$

$\Longrightarrow$

$$\begin{aligned} dd''_k(U_2) &= dd'_k(U_1) + 3dd'_{k-1}(U_1) + ss^2_{k+1}(U_1) + 3ss^2_k(U_1) \\ {\downarrow}ss^1_k(U_2) &= 2dd'_{k-1}(U_1) + 2ss^2_k(U_1) \end{aligned}$$

$\Longrightarrow$

$$\begin{aligned} dd''_0(U_2) &= dd'_0(U_1) + 0 + ss^2_1(U_1) + 0 = 4 + 2 = 6 \\ dd''_1(U_2) &= 0 + 3dd'_0(U_1) + 0 + 3ss^2_1(U_1) = 3 \times 4 + 3 \times 2 = 18 \\ {\downarrow}ss^1_1(U_2) &= 2dd'_0(U_1) + 2ss^2_1(U_1) = 2 \times 4 + 2 \times 2 = 12 \end{aligned}$$

2. Processing tree node 2 involves two steps, since it is a square vertex:

   (a) The first step involves amalgamating the component C to the component D.

   REMARK Notice that even though D has a single vertex, we can consider a second-root vertex adjacent to the single vertex and then work as before, using $dd''_0(D) = 1$ as the only non-zero sub-partial.

   Since $dd''_0(C) = 1$ and $dd''_0(D) = 1$, this case is similar to what occurred while processing tree node 1, where components A and B were vertex-amalgamated. The resulting graph $U_3 = C * D$ will have the same values for sub-partials as were produced for the subgraph $U_1 = A * B$. Thus, before the second step, the partials for U_3 are $dd'_0(U_3) = 4$ and $ss^2_1(U_3) = 2$.

   (b) In the second step, we save all partials of U_3 as dd''_i and ${\downarrow}ss^1_i$ in order to simulate dropping the second-root of U_3 and popping the new root on that part of U_3 which was previously the component C:

$$dd''_0(U_3) = 4,\ {\downarrow}ss^1_1(U_3) = 2$$

3. Processing tree node 3 means amalgamating the component U_2, that was produced while processing node 1, to the component U_3 produced while processing node 2. We refer to the component $U_2 * U_3$ as U_4. The non-zero sub-partials of U_2 are

$$dd''_0(U_2) = 6,\ \ dd''_1(U_2) = 18,\ \ {\downarrow}ss^1_1(U_2) = 12$$

   and the non-zero sub-partial of U_3 are

$$dd''_0(U_3) = 4,\ \ {\downarrow}ss^1_1(U_3) = 2$$

The productions needed for vertex-amalgamation of U_2 and U_3 are

$$\begin{aligned}
dd''_i(U_2) * dd''_j(U_3) &\longrightarrow 4dd'_{i+j}(U_4) + 2ss^2_{i+j+1}(U_4)\\
\downarrow ss^1_i(U_2) * dd''_j(U_3) &\longrightarrow 6\downarrow sd'_{i+j}(U_4)\\
dd''_i(U_2) * \downarrow ss^1_j(U_3) &\longrightarrow 6ds'_{i+j}(U_4)\\
\downarrow ss^1_i(U_2) * \downarrow ss^1_j(U_3) &\longrightarrow 6\downarrow ss^1_{i+j}(U_4)
\end{aligned}$$

$\Longrightarrow$

$$\begin{aligned}
dd'_k(U_4) &= 4dd''_k(U_2) \times dd''_0(U_3) = 4dd''_k(U_2) \times 4 = 16dd''_k(U_2)\\
ss^2_k(U_4) &= 2dd''_{k-1}(U_2) \times dd''_0(U_3) = 2dd''_{k-1}(U_2) \times 4 = 8dd''_{k-1}(U_2)\\
\downarrow sd'_k(U_4) &= 6\downarrow ss^1_k(U_2) \times dd''_0(U_3) = 6\downarrow ss^1_k(U_2) \times 4 = 24\downarrow ss^1_k(U_2)\\
ds'_k(U_4) &= 6dd''_{k-1}(U_2) \times \downarrow ss^1_1(U_3) = 6dd''_{k-1}(U_2) \times 2 = 12dd''_{k-1}(U_2)\\
\downarrow ss^1_k(U_4) &= 6\downarrow ss^1_{k-1}(U_2) \times \downarrow ss^1_1(U_3) = 6\downarrow ss^1_{k-1}(U_2) \times 2 = 12\downarrow ss^1_{k-1}(U_2)
\end{aligned}$$

$\Longrightarrow$

$$\begin{aligned}
dd'_0(U_4) &= 16dd''_0(U_2) = 16 \times 6 = 96\\
\downarrow sd'_1(U_4) &= 24\downarrow ss^1_1(U_2) = 24 \times 12 = 288\\
dd'_1(U_4) &= 16dd''_1(U_2) = 16 \times 18 = 288\\
ds'_1(U_4) &= 12dd''_0(U_2) = 12 \times 6 = 72\\
ss^2_1(U_4) &= 8dd''_0(U_2) = 8 \times 6 = 48\\
ds'_2(U_4) &= 12dd''_1(U_2) = 12 \times 18 = 216\\
ss^2_2(U_4) &= 8dd''_1(U_2) = 8 \times 18 = 144\\
\downarrow ss^1_2(U_4) &= 12\downarrow ss^1_1(U_2) = 12 \times 12 = 144
\end{aligned}$$

4. Processing tree node 4 involves amalgamating subgraphs E and U_4, followed by a self-amalgamation. We refer to the subgraph $E * U_4$ as U_5, and we refer to the subgraph that results from self-amalgamating U_5 as U_6.

   (a) For this purpose, five productions are needed for the cases $dd'' * dd'$, $dd'' * ss^2$, $dd'' * \downarrow sd'$, $dd'' * ds'$, and $dd'' * \downarrow ss^1$, since $dd''_0(E) = 1$ is the only non-zero sub-partial of E. These are the relevant productions:

$$\begin{aligned}
dd''_i(E) * dd'_j(U_4) &\longrightarrow 2dd^0_{i+j}(U_5) + 2dd'_{i+j}(U_5) + \uparrow sd'_{i+j+1}(U_5) + \downarrow sd'_{i+j+1}(U_5)\\
dd''_i(E) * ss^2_j(U_4) &\longrightarrow 4ds'_{i+j}(U_5) + 2dd''_{i+j}(U_5)\\
dd''_i(E) * \downarrow sd'_j(U_4) &\longrightarrow 6dd'_{i+j}(U_5)\\
dd''_i(E) * ds'_j(U_4) &\longrightarrow 2ds^0_{i+j}(U_5) + 2ds'_{i+j}(U_5) + \downarrow ss^1_{i+j+1}(U_5) + \uparrow ss^1_{i+j+1}(U_5)\\
dd''_i(E) * \downarrow ss^1_j(U_4) &\longrightarrow 6ds'_{i+j}(U_5)
\end{aligned}$$

$$\Longrightarrow$$

$$\begin{aligned}
dd_k^0(U_5) &= 2dd_0''(E) \times dd_k'(U_4) = 2dd_k'(U_4)\\
dd_k'(U_5) &= 2dd_0''(E) \times dd_k'(U_4) + 6dd_0''(E) \times \downarrow sd_k'(U_4)\\
&= 2dd_k'(U_4) + 6\downarrow sd_k'(U_4)\\
\uparrow sd_k'(U_5) &= dd_0''(E) \times dd_{k-1}'(U_4) = dd_{k-1}'(U_4)\\
\downarrow sd_k'(U_5) &= dd_0''(E) \times dd_{k-1}'(U_4) = dd_{k-1}'(U_4)\\
ds_k'(U_5) &= 4dd_0''(E) \times ss_k^2(U_4) + 2dd_0''(E) \times ds_k'(U_4) + 6dd_0''(E) \times \downarrow ss_k^1(U_4)\\
&= 4ss_k^2(U_4) + 2ds_k'(U_4) + 6\downarrow ss_k^1(U_4)\\
dd_k''(U_5) &= 2dd_0''(E) \times ss_k^2(U_4) = 2ss_k^2(U_4)\\
ds_k^0(U_5) &= 2dd_0''(E) \times ds_k'(U_4) = 2ds_k'(U_4)\\
\uparrow ss_k^1(U_5) &= dd_0''(E) \times ds_{k-1}'(U_4) = ds_{k-1}'(U_4)\\
\downarrow ss_k^1(U_5) &= dd_0''(E) \times ds_{k-1}'(U_4) = ds_{k-1}'(U_4)
\end{aligned}$$

$$\Longrightarrow$$

$$\begin{aligned}
dd_0^0(U_5) &= 2dd_0'(U_4) = 2 \times 96 = 192\\
dd_1^0(U_5) &= 2dd_1'(U_4) = 2 \times 288 = 576\\
dd_0'(U_5) &= 2dd_0'(U_4) + 6\downarrow sd_0'(U_4) = 2 \times 96 + 0 = 192\\
dd_1'(U_5) &= 2dd_1'(U_4) + 6\downarrow sd_1'(U_4) = 2 \times 288 + 6 \times 288 = 2304\\
\uparrow sd_1'(U_5) &= dd_0'(U_4) = 96\\
\uparrow sd_2'(U_5) &= dd_1'(U_4) = 288\\
\downarrow sd_1'(U_5) &= dd_0'(U_4) = 96\\
\downarrow sd_2'(U_5) &= dd_1'(U_4) = 288\\
ds_1'(U_5) &= 4ss_1^2(U_4) + 2ds_1'(U_4) + 6\downarrow ss_1^1(U_4) = 4 \times 48 + 2 \times 72 + 0 = 336\\
ds_2'(U_5) &= 4ss_2^2(U_4) + 2ds_2'(U_4) + 6\downarrow ss_2^1(U_4)\\
&= 4 \times 144 + 2 \times 216 + 6 \times 144 = 1872\\
dd_1''(U_5) &= 2ss_1^2(U_4) = 2 \times 48 = 96\\
dd_2''(U_5) &= 2ss_2^2(U_4) = 2 \times 144 = 288\\
ds_1^0(U_5) &= 2ds_1'(U_4) = 2 \times 72 = 144\\
ds_2^0(U_5) &= 2ds_2'(U_4) = 2 \times 216 = 432\\
\uparrow ss_2^1(U_5) &= ds_1'(U_4) = 72\\
\uparrow ss_3^1(U_5) &= ds_2'(U_4) = 216\\
\downarrow ss_2^1(U_5) &= ds_1'(U_4) = 72\\
\downarrow ss_3^1(U_5) &= ds_2'(U_4) = 216
\end{aligned}$$

(b) Productions 3.1−3.5, 3.7−3.8, and 3.10−3.11 are needed for self-amalgamation of U_5:

$$dd_i^0(U_5) \longrightarrow 4dd_{i+1}''(U_6) + 2\downarrow ss_{i+2}^1(U_6) \qquad dd_i'(U_5) \longrightarrow dd_i''(U_6) + 3dd_{i+1}''(U_6)$$

$$
\begin{aligned}
dd''_i(U_5) &\longrightarrow 4dd''_i(U_6) + 2{\downarrow}ss^1_{i+1}(U_6) && + 2{\downarrow}ss^1_{i+1}(U_6)\\
ds^0_i(U_5) &\longrightarrow 6dd''_{i+1}(U_6) && ds'_i(U_5) \longrightarrow 3dd''_i(U_6) + 3{\downarrow}ss^1_{i+1}(U_6)\\
{\uparrow}sd'_i(U_5) &\longrightarrow 3dd''_i(U_6) + 3{\downarrow}ss^1_{i+1}(U_6) && {\downarrow}sd'_i(U_5) \longrightarrow 3{\downarrow}ss^1_i(U_6) + 3{\downarrow}ss^1_{i+1}(U_6)\\
{\uparrow}ss^1_i(U_5) &\longrightarrow 6dd''_i(U_6) && {\downarrow}ss^1_i(U_5) \longrightarrow 6{\downarrow}ss^1_i(U_6)
\end{aligned}
$$

$\Longrightarrow$

$$
\begin{aligned}
dd''_k(U_6) &= 4dd^0_{k-1}(U_5) + dd'_k(U_5) + 3dd'_{k-1}(U_5) + 4dd''_k(U_5) + 6ds^0_{k-1}(U_5)\\
&\quad + 3ds'_k(U_5) + 3{\uparrow}sd'_k(U_5) + 6{\uparrow}ss^1_k(U_5)\\
{\downarrow}ss^1_k(U_6) &= 2dd^0_{k-2}(U_5) + 2dd'_{k-1}(U_5) + 2dd''_{k-1}(U_5) + 3ds'_{k-1}(U_5)\\
&\quad + 3{\uparrow}sd'_{k-1}(U_5) + 3{\downarrow}sd'_k(U_5) + 3{\downarrow}sd'_{k-1}(U_5) + 6{\downarrow}ss^1_k(U_5)
\end{aligned}
$$

$\Longrightarrow$

$$
\begin{aligned}
dd''_0(U_6) &= 0 + dd'_0(U_5) + 0 + 0 + 0 + 0 + 0 + 0 = 192\\
dd''_1(U_6) &= 4dd^0_0(U_5) + dd'_1(U_5) + 3dd'_0(U_5) + 4dd''_1(U_5) + 0 + 3ds'_1(U_5)\\
&\quad + 3{\uparrow}sd'_1(U_5) + 0\\
&= 4\times 192 + 2304 + 3\times 192 + 4\times 96 + 3\times 336 + 3\times 96 = 5328\\
dd''_2(U_6) &= 4dd^0_1(U_5) + 0 + 3dd'_1(U_5) + 4dd''_2(U_5) + 6ds^0_1(U_5) + 3ds'_2(U_4)\\
&\quad + 3{\uparrow}sd'_2(U_5) + 6{\uparrow}ss^1_2(U_5)\\
&= 4\times 576 + 3\times 2304 + 4\times 288 + 6\times 144 + 3\times 1872 + 3\times 288\\
&\quad + 6\times 72 = 18144\\
dd''_3(U_6) &= 0 + 0 + 0 + 0 + 6ds^0_2(U_5) + 0 + 0 + 6{\uparrow}ss^1_3(U_5)\\
&= 6\times 432 + 6\times 216 = 3888
\end{aligned}
$$

$$
\begin{aligned}
{\downarrow}ss^1_0(U_6) &= 0\\
{\downarrow}ss^1_1(U_6) &= 0 + 2dd'_0(U_5) + 0 + 0 + 0 + 3{\downarrow}sd'_1(U_5) + 0 + 0\\
&= 2\times 192 + 3\times 96 = 672\\
{\downarrow}ss^1_2(U_6) &= 2dd^0_0(U_5) + 2dd'_1(U_5) + 2dd''_1(U_5) + 3ds'_1(U_5) + 3{\uparrow}sd'_1(U_5)\\
&\quad + 3{\downarrow}sd'_2(U_5) + 3{\downarrow}sd'_1(U_5) + 6{\downarrow}ss^1_2(U_5)\\
&= 2\times 192 + 2\times 2304 + 2\times 96 + 3\times 336 + 3\times 96 + 3\times 288\\
&\quad + 3\times 96 + 6\times 72 = 8064\\
{\downarrow}ss^1_3(U_6) &= 2dd^0_1(U_5) + 0 + 2dd''_2(U_5) + 3ds'_2(U_5) + 3{\uparrow}sd'_2(U_5) + 0 + 0\\
&\quad + 3{\downarrow}sd'_2(U_5) + 6{\downarrow}ss^1_3(U_5)\\
&= 2\times 576 + 2\times 288 + 3\times 1872 + 3\times 288 + 3\times 288 + 6\times 216\\
&= 10368
\end{aligned}
$$

5. Processing tree node 5 returns immediately, since it is the root node. Thus, the assembled graph U_6 is the outerplanar graph G.

6. By summing the sub-partials, we obtain the genus distribution for G:

$$\begin{aligned} g_0(G) &= dd_0''(G) + 0 = 192 \\ g_1(G) &= dd_1''(G) + \downarrow ss_1^1(G) = 5328 + 672 = 6000 \\ g_2(G) &= dd_2''(G) + \downarrow ss_2^1(G) = 18144 + 8064 = 26208 \\ g_3(G) &= dd_3''(G) + \downarrow ss_3^1(G) = 3888 + 10368 = 14256 \end{aligned}$$

5 Time-Complexity Analysis

Normalizing the outerplane embedding and obtaining the split graph are $O(n)$ operations, where n is the number of vertices of the given graph. Since the split graph has fewer than n components, it follows that forming an incidence tree is also $O(n)$.

Theorem 5.1 *A subgraph H of a 4-regular outerplanar graph, with $|V(H)| = k$ number of vertices, has $O(k)$ number of partials.*

Proof Let k be the number of vertices in a subgraph H assembled using the algorithm. A connected 4-regular graph with k vertices has cycle rank $\beta = k+1$. Since H is a subgraph of a connected 4-regular graph, the maximum genus of H is bounded by $\left\lfloor \frac{\beta(H)}{2} \right\rfloor \leq \left\lfloor \frac{k+1}{2} \right\rfloor$. As there are 12 sub-partial types, the number of sub-partials of H is bounded from above by $12 \times \left\lfloor \frac{k+1}{2} \right\rfloor$. ♢

1. **Time-Complexity of an Amalgamation Operation:** If the parent component has p vertices and the child component has q vertices, then by Theorem 5.1 their number of partials are $O(p)$ and $O(q)$, respectively. Applying a single production for an amalgamation step is $O(1)$. Consequently, the complexity of applying all productions for a single amalgamation is $O(pq)$. The number of vertices in the subgraph resulting from amalgamation is $O(p+q)$.

2. **Time-Complexity of a Self-Amalgamation Operation:** If the graph component undergoing self-amalgamation has p vertices then the complexity of applying self-amalgamation productions is $O(p)$. The number of vertices in the resulting graph component is $O(p)$.

Let $n_1, n_2, \cdots, n_r$ be the number of vertices in the components of the split graph of a graph G. From the first point above, it follows that if a component of size $\sum_{i\in I} n_i$ amalgamates to a component of size $\sum_{j\in J} n_j$, where I and J are some disjoint sets, then the time-complexity of performing the operation is

$$\sum_{i\in I} n_i \sum_{j\in J} n_j$$

and the size of the resulting graph is

$$\sum_{i \in I \cup J} n_i$$

As each coupled pair of vertices is amalgamated only once, the complexity of reconstructing the original graph is $O(\sum_{i \in I,\ j \in J} n_i n_j)$ for some disjoint sets I and J. Therefore, the complexity of the given algorithm is $O((n_1 + \cdots + n_r)(n_1 + \cdots + n_r)) = O(n \cdot n) = O(n^2)$.

6 Correctness

In order to show that the algorithm given in §4 correctly computes genus distribution of 4-regular outerplanar graphs, we need to address the question of whether root vertices will be available at the right time and the right place for amalgamations and self-amalgamations. As before, we regard the components represented by round and square nodes of an incidence tree as nodes of the tree themselves, and we use expressions like "parent component", "child component" etc. We argue inductively for a tree node that the graph constructed by processing each of the child nodes of that node contains two root vertices and that these roots are available for the next amalgamation operation on that graph.

Lemma 6.1 *Components of an incidence tree that are coupled are always in an ancestor-descendant relationship.*

Proof When two components are not coupled, they are said to be ***separated***. Since an incidence tree is created in a depth-first manner and since depth-first trees have no cross-edges, it follows that the components from sibling subtrees of an incidence tree are separated from each other. Thus, the vertices that recombine under amalgamation or self-amalgamation must initially belong to coupled components that are in an ancestor-descendant relationship. ♢

Theorem 6.2 *Let $\mathcal{P}$ be a component with one or more child components, none of which correspond to square nodes. Then every graph in the sequence of graphs produced by processing the children of $\mathcal{P}$ contains two root vertices, such that these root vertices are available for the next amalgamation.*

Proof Before any of its child nodes are processed, $\mathcal{P}$ is homeomorphic to a cycle graph and has two roots. When $\mathcal{P}$ has more than one child, processing its first child involves an amalgamation of the child with $\mathcal{P}$ and necessarily ends with a self-amalgamation that produces two consecutive roots on the resulting graph. This is illustrated in Figure 6.1. The first-root of the double-rooted graph produced as a result of the self-amalgamation corresponds to the vertex popped farther in the counter-clockwise direction, as shown. The first- and second-root of each amalgamand are labeled 1 and 2, respectively. Thus,

the roots are available at the right place for the next child to be processed. All but the last child will be eventually processed similarly during the post-order traversal. In case of the last child or the only child, if $\mathcal{P}$ has exactly one child component, its eventual amalgamation with $\mathcal{P}$ may or may not be immediately followed by a self-amalgamation. If there is an immediate self-amalgamation then as before, it will produce two adjacent roots, which can again be used for attaching the resulting subgraph to its parent. On the other hand, if there is no immediate self-amalgamation, then the second-root is preserved till a later time, when it undergoes a self-amalgamation while processing an ancestor of $\mathcal{P}$ or $\mathcal{P}$ itself. ♢

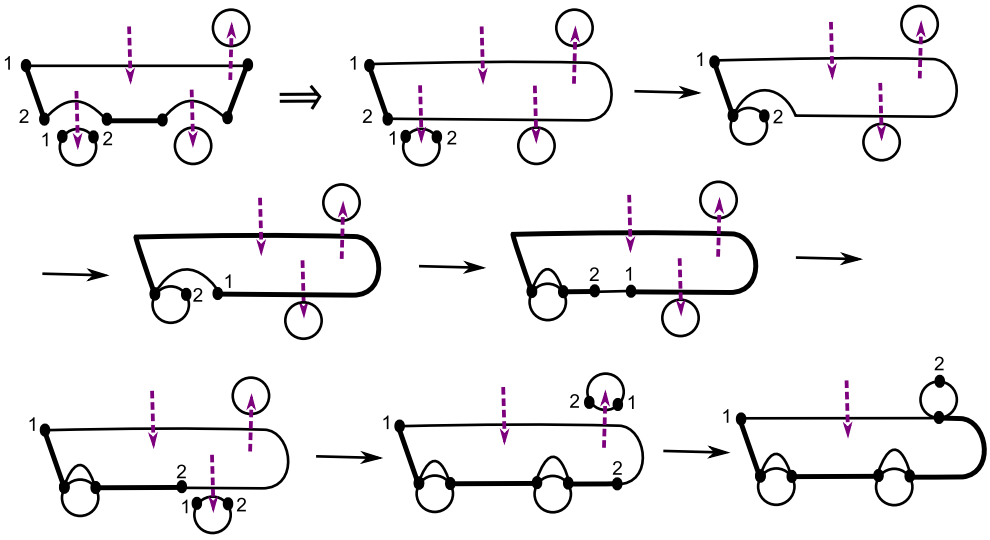

Figure 6.1: An example of propagation of root vertices.

Lemma 6.3 *No self-amalgamation is required while processing a square node.*

Proof Processing a square node represents the need to amalgamate coupled vertices that arise either by splitting a cut vertex or by splitting the endpoint of a loop. A square node has descendant components only in the former case. The descendants of a square node are separated from its ancestor components since these two sets of components arise from splitting different blocks of the outerplanar graph embedding. In addition, a component corresponding to a square node has exactly one vertex coupled to its parent component. Therefore, no self-amalgamation is required when processing a square node. ♢

Theorem 6.4 *Let $\mathcal{P}$ be a component with at-least one child component corresponding to a square node. Then every graph in the sequence of graphs produced by processing the children of $\mathcal{P}$ contains two root vertices, such that these root vertices are available for the next amalgamation.*

Proof Let $\mathcal{S}$ be the first child component of $\mathcal{P}$ corresponding to a square node. Then $\mathcal{S}$ is necessarily separated from its siblings as well as from the ancestor of $\mathcal{P}$, by Lemma 6.1 and 6.3. The amalgamation of $\mathcal{S}$ to its parent component produces two roots on the resulting graph, only the first of which lies on its subgraph $\mathcal{P}$. The other root, that lies on the subgraph $\mathcal{S}$, is redundant by Lemma 6.3. So we drop the redundant root, retaining only the information for the first-root in the form of single-root partials s_i and d_i. We then pop up a new second-root adjacent to our first-root on the subgraph $\mathcal{P}$ and re-adjust the numbers we had for s_i and d_i as $\downarrow ss_i^1$ and dd_i'', respectively. Thus, as we continue to process the remaining child components of $\mathcal{P}$ or $\mathcal{P}$ itself, both roots will be available for the next amalgamation. $\diamondsuit$

By Theorem 6.2 and 6.4, each component is readily amalgamated to its parent node. If a self-amalgamation is required, it is performed as soon as the opportunity presents itself. In this bottom up fashion, eventually the entire graph is reconstructed.

7 Conclusions

We have presented an $O(n^2)$-time algorithm for calculating the genus distribution of 4-regular outerplanar graphs, where n is the number of vertices in the graph. The algorithm accomplishes this by simulating the synthesis of the graph by iteratively applying productions for amalgamation and self-amalgamation. This in turn demonstrates the power of theoretical results developed by [GKP10], [Gr11a] and other related papers. We have already seen in [Gr11b] an $O(n^2)$-time algorithm for computing the genus distribution of 3-regular outerplanar graphs. A natural direction for future research is to ask whether such polynomial-time algorithms exist for k-regular outerplanar graphs for larger k, as well as for all outerplanar graphs.

Another point of interest is that regular projections of all knots and links correspond to 4-regular graphs. This raises the question, what kinds of knots and links correspond to 4-regular outerplanar graphs?

A special case of 4-regular outerplanar graphs are the 4-regular Hamiltonian outerplanar graphs, for which the boundary of f_∞ is a Hamiltonian cycle, and for which the interior edges can be regarded as comprising the polygons inscribed inside the Hamiltonian cycle. One observes that all 4-regular Hamiltonian planar graphs can also be characterized in a similar manner, as a Hamiltonian cycle with "outer" polygons as well as "inner" polygons. Are the techniques in this paper, therefore, extendible to 4-regular planar Hamiltonian graphs?

References

[BGGS00] C. P. Bonnington, M. J. Grannell, T. S. Griggs, and J. Širáň, Exponential families of non-isomorphic triangulations of complete graphs, *J. Combin. Theory (B)* **78** (2000), 169–184.

[ChLiWa06] Y. C. Chen, Y. P. Liu, and T. Wang, The total embedding distributions of cacti and necklaces, *Acta Math. Sinica — English Series* **22** (2006), 1583–1590.

[FuGrSt89] M. L. Furst, J. L. Gross and R. Statman, Genus distribution for two classes of graphs, *J. Combin. Theory (B)* **46** (1989), 22–36.

[GoRiSi07] L. Goddyn, R. B. Richter, and J. Širáň, Triangular embeddings of complete graphs from graceful labellings of paths, *J. Combin. Theory (B)* **97** (2007), 964–970.

[GrGr08] M. J. Grannell and T. S. Griggs, A lower bound for the number of triangular embeddings of some complete graphs and complete regular tripartite graphs, *J. Combin. Theory (B)* **98** (2008), 637–650.

[Gr10] J. L. Gross, Genus distribution of graphs under surgery: Adding edges and splitting vertices, *New York J. Math.* **16** (2010), 161-178.

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

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

[GrFu87] 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.

[GrRoTu89] J. L. Gross, D. P. Robbins and T. W. Tucker, Genus distributions for bouquets of circles, *J. Combin. Theory (B)* **47** (1989), 292–306.

[GrTu87] J. L. Gross and T. W. Tucker, *Topological Graph Theory*, Dover, 2001; (original edn. Wiley, 1987).

[Ja87] D. M. Jackson, Counting cycles in permutations by group characters with an application to a topological problem, *Trans. Amer. Math. Soc.* **299** (1987), 785–801.

[KPG10] I. F. Khan, M. I. Poshni, and J. L. Gross, Genus distribution of graph amalgamations: Pasting when one root has arbitrary degree, *Ars Math. Contemporanea* **3** (2010), 121-138.

[KoVo02] V. P. Korzhik and H-J Voss, Exponential families of non-isomorphic non-triangular orientable genus embeddings of complete graphs, *J. Combin. Theory (B)* **86** (2002), 86–211.

[KwLe93] J. H. Kwak and J. Lee, Genus polynomials of dipoles, *Kyungpook Math. J.* **33** (1993), 115–125.

[KwLe94] J. H. Kwak and J. Lee, Enumeration of graph embeddings, *Discrete Math.* **135** (1994), 129–151.

[KwSh02] J. H. Kwak and S. H. Shim, Total embedding distributions for bouquets of circles, *Discrete Math.* **248** (2002), 93–108.

[McG87] L. A. McGeoch, Algorithms for two graph problems: computing maximum-genus imbedding and the two-server problem, PhD thesis, Carnegie-Mellon University, 1987.

[Mu99] B. P. Mull, Enumerating the orientable 2-cell imbeddings of complete bipartite graphs, *J. Graph Theory* **30** (1999), 77–90.

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

[PKG11] M. I. Poshni, I. F. Khan, and J. L. Gross, Genus distributions of graphs under self-edge-amalgamations, *Ars Math. Contemporanea* (2011), 22 pages, to appear.

[St90] S. Stahl, Region distributions of graph embeddings and Stirling numbers, *Discrete Math.* **82** (1990), 57–78.

[St91a] S. Stahl, Permutation-partition pairs III: Embedding distributions of linear families of graphs, *J. Combin. Theory (B)* **52** (1991), 191–218.

[St91b] S. Stahl, Region distributions of some small diameter graphs, *Discrete Math.* **89** (1991), 281–299.

[Tesa00] E. H. Tesar, Genus distribution of Ringel ladders, *Discrete Math.* **216** (2000) 235–252.

[ViWi07] T. I. Visentin and S. W. Wieler, On the genus distribution of (p, q, n)-dipoles, *Electronic J. Combin.* **14** (2007), Art. No. R12.

[WaLi06] L. X. Wan and Y. P. Liu, Orientable embedding distributions by genus for certain types of graphs, *Ars Combin.* **79** (2006), 97–105.

[WaLi08] L. X. Wan and Y. P. Liu, Orientable embedding genus distribution for certain types of graphs, *J. Combin. Theory (B)* **47** (2008), 19–32.

[Wh01] A. T. White, *Graphs of Groups on Surfaces*, North-Holland, 2001.