Finding Obstructions within Irreducible Triangulations

by

Russell Campbell
B.Sc., University of the Fraser Valley, 2007
M.Sc., University of Victoria, 2009

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Computer Science

Finding Obstructions within Irreducible Triangulations

by

Russell Campbell
B.Sc., University of the Fraser Valley, 2007
M.Sc., University of Victoria, 2009

Supervisory Committee

Dr. Wendy Myrvold, Supervisor
Department of Computer Science

Dr. Frank Ruskey, Departmental Member
Department of Computer Science

Dr. Ryan Budney, Outside Member
Department of Mathematics and Statistics

Abstract

The main results of this dissertation show evidence supporting the Successive Surface Scaffolding Conjecture. This is a new conjecture that, if true, guarantees the existence of all the wye-delta-order minimal obstructions of a surface $\mathbb{S}$ as subgraphs of the irreducible triangulations of the surface $\mathbb{S}$ with a crosscap added. A new data structure, i.e. an augmented rotation system, is presented and used to create an exponential-time algorithm for embedding graphs in any surface with a constant-time check of the change in genus when inserting an edge. A *depiction* is a new formal definition for representing an embedding graphically, and it is shown that more than one depiction can be given for nonplanar embeddings, and that sometimes two depictions for the same embedding can be drastically different from each other. An algorithm for finding the essential cycles of an embedding is given, and is used to confirm for the projective-plane obstructions, a theorem that shows any embedding of an obstruction must have every edge in an essential cycle. Obstructions of a general surface $\mathbb{S}$ that are minor-minimal and not double-wye-delta-minimal are shown to each have an embedding on the surface $\mathbb{S}$ with a crosscap added. Finally, open questions for further research are presented.

Table of Contents

Supervisory Committee ii

Abstract iii

Table of Contents iv

List of Figures vi

List of Tables ix

Acknowledgements xi

1 Motivations for Research 1
- 1.1 Definitions . . . 1
- 1.2 Successive Surface Scaffolding Conjecture . . . 2
- 1.3 Summary of Results . . . 4

2 Topological Graph Theory 6

3 Combinatorial Embeddings 10
- 3.1 Definitions . . . 10
- 3.2 Extending an Embedding . . . 14
- 3.3 Augmented Rotation Systems . . . 16

4 Embedding Algorithms 26
- 4.1 Literature Review . . . 26
- 4.2 Equivalence Relations . . . 28
- 4.3 Reducing Repetition . . . 30
- 4.4 Exponential Embedding Algorithm . . . 32

5 Obstructions **40**
5.1 Definitions . . . 40
5.2 Literature Review for Obstructions . . . 43
5.3 Results . . . 44

6 Depictions of Embeddings **51**
6.1 Definitions . . . 51
6.2 Different Depictions of the Same Embedding . . . 56
6.3 Algorithm to Obtain a Depiction . . . 60

7 Irreducible Triangulations **75**
7.1 Introduction . . . 75
7.2 Results . . . 78
7.2.1 Plane Obstructions in $\mathbb{N}_1$ Irreducible Triangulations . . . 86
7.2.2 $\mathbb{N}_1$ Obstructions in $\mathbb{N}_2$ Irreducible Triangulations . . . 87
7.2.3 Torus Obstructions in $\mathbb{N}_3$ Irreducible Triangulations . . . 94

8 Essential Cycles **103**
8.1 Cutting Along a Cycle . . . 103
8.2 Algorithm for Cutting . . . 107

9 Open Problems **113**

Appendices **115**
A Projective-Planar Obstruction Names . . . 115
B Klein-Bottle Irreducible-Triangulation Names . . . 116

Bibliography **117**

List of Figures

2.1 Adding a handle to a sphere 8

3.1 Edge uv inserted into a face of some embedding. The blue curves correspond to the new facial walks of $\widetilde{H}$. The dotted lines complete the boundary edges of old facial walks in $\widetilde{G}$, and note that we do not indicate the signatures of the edges in old facial walks. 17

3.2 Edge uv inserted with each end into a different face of some embedding. The blue curves correspond to the new facial walks of $\widetilde{H}$. The dotted lines complete the boundary edges of old facial walks in $\widetilde{G}$. 18

4.1 An image of an embedding of two blocks of K_7 in $\mathbb{S}_2$ with two handles shown as disks A and B with orientations marked as indicated in blue arrows. The edges using the handles are matched by colour. 39

6.1 An example of an embedding drawn with 5 edges crossing in a cyclic order. 52

6.2 An example of an embedding where 5 crossing edges in the cyclic order presented in Figure 6.1 are subdivided and modified as discussed. A cycle C representing a crosscap is indicated with red edges, and note that the resulting embedding is planar. 53

6.3 An example of a depiction of a torus obstruction with 15 vertices, labelled 0 to 14, embedded in $\mathbb{N}_3$ where the handle faces are indicated in blue, the crosscap face is indicated in red, and the feature vertices are labelled 15 to 28, with the first of each pair numbered odd $2k-1$ and the second in the pair numbered $2k$, for $k = 8, 9, \ldots, 16$. 56

6.4 A depiction of an embedding $\widetilde{G_1}$ with 12 vertices embedded in the surface $\mathbb{N}_3$ with one crosscap (outlined red circle) and one handle (two outlined blue circles identified as indicated). Three separate faces are coloured yellow, green, and orange to help indicate the same faces in Figure 6.5. Note that feature vertices are omitted. . 58
6.5 Another depiction of the embedding $\widetilde{G_1}$ from Figure 6.4 with 12 vertices embedded in the surface $\mathbb{N}_3$ with three crosscaps (outlined red circles), and with three separate faces coloured yellow, green, and orange to help indicate the same faces in Figure 6.4. . 58
6.6 A depiction of an embedding $\widetilde{G_2}$ with 15 vertices embedded in the surface $\mathbb{N}_3$ with three crosscaps (outlined red circles), and with three separate faces coloured yellow, green, and orange to help indicate the same faces in Figure 6.7. 59
6.7 A depiction of the embedding $\widetilde{G_2}$ from Figure 6.6 with 15 vertices embedded in the surface $\mathbb{N}_3$ with one crosscap (outlined red circle) and one handle (two outlined blue circles identified as indicated). Three separate faces are coloured yellow, green, and orange to help indicate the same faces in Figure 6.6. Note that the dashed edges in the green and orange faces use both the crosscap and handle. 60

7.1 Case 1 for the triangulated region between homotopic essential 3-cycles $C_1 = vv_1v'$ and $C_2 = vv_2v'$. 80
7.2 Case 2 for the triangulated region between homotopic essential 3-cycles $C_1 = vv_1v_3$ and $C_2 = vv_2v_4$. 81
7.3 Case 3 for the triangulated region between homotopic essential 3-cycles $C_1 = vv_1v_3$ and $C_2 = vv_2v_4$. 83
7.4 Case 4 for the triangulated region between homotopic essential 3-cycles $C_1 = vv_1v_3$ and $C_2 = vv_2v_4$. 84
7.5 Irreducible triangulations of the projective plane obtained from adding red edges to embeddings of $K_{3,3}$. 86
7.6 Irreducible triangulations of the projective plane obtained by adding one or two red vertices to the two embeddings of K_5. The red vertices are made adjacent to every edge of the face they are placed inside. 86
7.7 Triangulations of the projective plane obtained by adding red edges to embeddings of $K_{3,3}$. Some red vertices are also added. Edges which are contractible are dashed. 87

7.8 A_1 obstruction of $\mathbb{N}_1$ — $Kc4$ triangulation of $\mathbb{N}_2$ 87
7.9 A_2 obstruction of $\mathbb{N}_1$ — $Kh6$ triangulation of $\mathbb{N}_2$ 88
7.10 B_1 obstruction of $\mathbb{N}_1$ — $Kh6$ triangulation of $\mathbb{N}_2$ 88
7.11 B_3 obstruction of $\mathbb{N}_1$ — $Kh6$ triangulation of $\mathbb{N}_2$ 89
7.12 C_7 obstruction of $\mathbb{N}_1$ — $Kh4$ triangulation of $\mathbb{N}_2$ 89
7.13 D_9 obstruction of $\mathbb{N}_1$ — $Kh25$ triangulation of $\mathbb{N}_2$ 90
7.14 D_{12} obstruction of $\mathbb{N}_1$ — $Kh7$ triangulation of $\mathbb{N}_2$ 90
7.15 D_{17} obstruction of $\mathbb{N}_1$ — $Kh1$ triangulation of $\mathbb{N}_2$ 91
7.16 E_3 obstruction of $\mathbb{N}_1$ — $Kh6$ triangulation of $\mathbb{N}_2$ 91
7.17 E_{18} obstruction of $\mathbb{N}_1$ — $Kh2$ triangulation of $\mathbb{N}_2$ 92
7.18 E_{22} obstruction of $\mathbb{N}_1$ — $Kh13$ triangulation of $\mathbb{N}_2$ 92
7.19 C_1 obstruction of $\mathbb{N}_1$ — $Kc2$ triangulation of $\mathbb{N}_2$ 93
7.20 D_3 obstruction of $\mathbb{N}_1$ — $Kh2$ triangulation of $\mathbb{N}_2$ 93
7.21 D_4 obstruction of $\mathbb{N}_1$ — $Kh19$ triangulation of $\mathbb{N}_2$ 94
7.22 E_{19} obstruction of $\mathbb{N}_1$ — $Kc1$ triangulation of $\mathbb{N}_2$ 94
7.23 F_1 obstruction of $\mathbb{N}_1$ — $Kc1$ triangulation of $\mathbb{N}_2$ 97
7.24 F_6 obstruction of $\mathbb{N}_1$ — $Kc2$ triangulation of $\mathbb{N}_2$ 97
7.25 G_1 obstruction of $\mathbb{N}_1$ — $Kc2$ triangulation of $\mathbb{N}_2$ 98
7.26 The minor-order projective-plane obstruction E_2. 98

List of Tables

5.1 Operations which reduce the order of a graph. 41
5.2 Number of obstructions on the projective plane, $\mathbb{N}_1$. The column labelled M_i gives the number of obstructions in M_i, but not in M_{i+1}, for $i = 1, \ldots, 3$. 43
5.3 The number of 3-regular topological obstructions of the torus. . . 44

7.1 ([Sul06a]) Number of triangulations of the torus, $\mathbb{S}_1$. For all tables of triangulation counts δ denotes minimum degree of the triangulations. There is a total of 21 irreducible triangulations. . . 79
7.2 ([Sul06a]) Number of triangulations of the double torus, $\mathbb{S}_2$. . . . 79
7.3 ([Sul06a]) Number of triangulations of the projective plane, $\mathbb{N}_1$. . 80
7.4 ([Sul06a]) Number of triangulations of the Klein bottle, $\mathbb{N}_2$, with total 29 irreducible triangulations. 80
7.5 ([Sul06a]) Number of triangulations of the surface $\mathbb{N}_3$, with 9708 total irreducible triangulations. 81
7.6 ([Sul06a]) Number of triangulations of the surface $\mathbb{N}_4$. 82
7.7 Columns 1 to 14 of the total occurrences of projective-plane obstructions within the irreducible triangulations of the Klein bottle as either a subgraph or as a subdivision. 95
7.8 Columns 15 to 29 of the total occurrences of projective-plane obstructions within the irreducible triangulations of the Klein bottle as either a subgraph or as a subdivision. 96
7.9 Columns 1 to 14 of the occurrences of projective-plane obstructions within the irreducible triangulations of the Klein bottle as a subdivision only. 98
7.10 Columns 15 to 29 of the occurrences of projective-plane obstructions within the irreducible triangulations of the Klein bottle as a subdivision only. 99

7.11 Columns 1 to 14 of the occurrences of projective-plane obstructions within the irreducible triangulations of the Klein bottle as a subgraph only. 100
7.12 Columns 15 to 29 of the occurrences of projective-plane obstructions within the irreducible triangulations of the Klein bottle as a subgraph only. 101
7.13 Number of known wye-delta-order obstructions on the torus by order (n) and size (m). 102

Acknowledgements

First and foremost, many thanks are given to my parents, my sister Maria, and her husband, Sage. Without their support no one would be reading this dissertation because it would not exist. Quite equal to this is the thoughtful mentoring of Wendy Myrvold, with her financial support, academic support, and generous feedback. I would be hard pressed to find another supervisor with greater ability to see me through the more difficult times of research paired with the stresses of life. These people have truly wanted me to succeed to the best of my ability and I want to acknowledge them for this.

I also want to thank the University of Victoria administration in both the general offices and the offices of the Department of Computer Science. This dissertation would not be possible without their careful efforts. I would like to acknowledge the Resource Centre for Students with a Disability and their training which helped me to see learning from different perspectives.

Finally, a thank you to all of my friends. They made writing this dissertation an experience interspersed with many interesting visits.

Chapter 1

Motivations for Research

This chapter explains the reasons for the directions this research has taken, beginning with basic definitions in Section 1.1. Then in Section 1.2, an introduction to the Successive Surface Scaffolding Conjecture is presented in both its weak and strong forms. Finally, a summary of results is given in Section 1.3.

1.1 Definitions

A *graph* $G = (V, E)$ is a set V of points called *vertices* with a set E of pairwise connections called *edges* between vertices. By convention the notations n and m are reserved to indicate the cardinality of the sets V and E, respectively. We choose to restrict n and m to be finite and any input graphs for algorithms have no multiple edges or loops where a vertex is connected to itself. In other words, when we discuss graphs we mean *simple* graphs. However, in Chapter 6 graph depictions used to represent embeddings may have loops or multiple edges.

A graph G is *embedded* in a surface $\mathbb{S}$ if the vertices of G are distinct elements of $\mathbb{S}$ and every edge e of G is a simple arc connecting in $\mathbb{S}$ its two incident vertices, such that the interior of e is disjoint from $G - e$. An *embedding* of a graph G in the surface $\mathbb{S}$ is an isomorphism of G with a graph $\widetilde{G}$ embedded in $\mathbb{S}$. Then $\widetilde{G}$ is referred to as a *representation* of G in $\mathbb{S}$. If a representation of G in $\mathbb{S}$ exists, then we say G *can be embedded* into $\mathbb{S}$. The *faces* of $\widetilde{G}$ are the regions (any two points in a region can be joined by a simple arc) of $X \setminus G$. If a region R of a surface $\mathbb{S}$ is homeomorphic to an

open disk in the plane, then R is called a 2-*cell*. When all the faces of an embedding $\widetilde{G}$ are 2-cell, then $\widetilde{G}$ is also called 2-cell. If a region R together with its boundary δR is homeomorphic to a closed disk in the plane, then R is called *closed* 2-cell. When all the faces of an embedding $\widetilde{G}$ are closed 2-cell, then $\widetilde{G}$ is also called closed 2-cell.

A *triangulation* of a surface $\mathbb{S}$ is an embedding on $\mathbb{S}$ such that every face is bound by exactly three edges. An edge e of a triangulation is *contractible* if contracting e and removing multiple edges results in another triangulation of the surface. A triangulation is *irreducible* if it has no contractible edges, and it also has the property that any two triangles share at most one edge.

The *orientable surface* $\mathbb{S}_h$ is the sphere $\mathbb{S}_0$ with $h \geq 0$ handles attached. For example, the torus is $\mathbb{S}_1$. The *nonorientable surface* $\mathbb{N}_k$ is the sphere with $k \geq 1$ Möbius strips attached. For example, the projective plane is $\mathbb{N}_1$. The attached Möbius strips are more commonly referred to as *crosscaps* on the nonorientable surface. For simplicity, let $\mathbb{N}_0 = \mathbb{S}_0$.

A *topological obstruction* of a surface $\mathbb{S}$ is a graph with neither isolated nor degree two vertices that does not embed in $\mathbb{S}$, yet $G - e$ does embed for any edge e. A *minor* obstruction has the added property that for any single edge contraction, the resulting graph embeds on $\mathbb{S}$. A *wye-delta* obstruction is a minor obstruction with the added property that if for any degree three vertex v the resulting graph G of removing v and its incident edges, and inserting all edges on the neighbours of v, then G embeds on $\mathbb{S}$. The set of all topological, minor, and wye-delta obstructions of a surface $\mathbb{S}$ are denoted $M_1(\mathbb{S})$, $M_2(\mathbb{S})$, and $M_3(\mathbb{S})$, respectively.

1.2 Successive Surface Scaffolding Conjecture

The majority of results in this research were motivated by the following conjecture, which was formulated by the author, Campbell.

Conjecture 1.1. *(***Successive Surface Scaffolding Conjecture involving subgraphs***) For all $k \geq 0$ the irreducible triangulations of $\mathbb{N}_{k+1}$ have all the connected obstructions in $M_3(\mathbb{N}_k)$ as subgraphs, and when $k = 2g$ for some $g \geq 0$ they also contain all the connected obstructions in $M_3(\mathbb{S}_g)$.*

Conjecture 1.2. *(***Successive Surface Scaffolding Conjecture involving homeomorphic subgraphs***) For all $k \geq 0$ the irreducible triangulations of $\mathbb{N}_{k+1}$ have all the connected obstructions in $M_3(\mathbb{N}_k)$ as subgraphs*

or subdivisions, and when $k = 2g$ for some $g \geq 0$ they also contain all the connected obstructions in $M_3(\mathbb{S}_g)$ as subgraphs or subdivisions.

Note that Conjecture 1.1 states a stronger condition than Conjecture 1.2 by not considering subdivisions of the obstructions in $M_3(\mathbb{N}_k)$. The weak form was suggested by W. Myrvold. The opportunity Conjectures 1.1 and 1.2 present is that the obstructions for a surface $\mathbb{S}$ can be found in the irreducible triangulations of the surface consisting of $\mathbb{S}$ plus a crosscap.

The conjecture does not hypothesize that the disconnected obstructions from the set $M_3(\mathbb{N}_k)$ nor $M_3(\mathbb{S}_g)$, for $k = 2g$, can be found as subgraphs of the irreducible triangulations to $\mathbb{N}_{k+1}$. For example, two disjoint copies of K_5 is an obstruction to the projective plane, and in Chapter 7, we show that this graph is not a subgraph of any of the irreducible triangulations of the Klein bottle. Therefore, we do not include disconnected obstructions of a surface within the Successive Surface Scaffolding Conjecture. Evidence to show the necessity of the wye-delta-order condition is given in Section 7.2.2 with respect to the graph $E_2 \in M_(\mathbb{N}_1)$.

Proving the Successive Surface Scaffolding Conjecture for $g = 1$ would immediately give a back door approach to obtaining an upper bound on the number of vertices of obstructions for $M_3(\mathbb{S}_1)$. The irreducible triangulations for $\mathbb{N}_3$ have been generated by Sulanke [Sul06a] and totals based on their order are given in Table 7.5. It is worth considering feasible strategies to search for wye-delta-order torus obstructions within the irreducible triangulations of $\mathbb{N}_3$.

To investigate the truth of these conjectures, faster embedding algorithms were needed, and described in Chapters 3 and 4. Ways to draw pictures of embeddings in Chapter 6 perhaps will aid insights into working towards proofs. A better understanding of obstructions motivates Chapter 5. A better understanding of triangulations of surfaces motivates Chapter 7.

In order for the Successive Surface Scaffolding Conjecture to be true, another conjecture based off of a known open problem [Arc95] regarding the embeddings of connected wye-delta obstructions must also hold true.

Conjecture 1.3. *Each obstruction in $M_3(\mathbb{S})$ for a surface $\mathbb{S}$ with Euler genus g has an embedding in $\mathbb{N}_{g+1}$.*

However, Conjecture 1.3 can be generalized to include all obstructions of $\mathbb{S}$. It is proved in Chapter 5 for a subset of the minor obstructions of an orientable surface, and the remaining obstructions have yet to be shown such an embedding exists for each.

1.3 Summary of Results

Chapter 2 is an optional set of definitions with respect to graph theory from the perspective of topology. They are useful for gaining an intuitive understanding of the topics covered in this research.

Chapter 3 contains Section 3.1 which presents definitions for understanding embeddings. In Section 3.2, a known theorem is detailed for the change in Euler genus to an embedding when an edge is inserted. In Section 3.3, a new data structure called an augmented rotation system is explained followed by $\mathcal{O}(m)$ algorithms `WalkOneFace` and `WalkAllFaces` which use the new data structure. This new data structure also makes it possible to compute in $\mathcal{O}(1)$ time the change in genus of a subembedding when an edge is inserted into it.

Chapter 4 in Section 4.1 gives a summary of the literature concerning embedding algorithms. In Section 4.2, definitions are set up for equivalence relations between combinatorial embeddings called flip, switch, and switch-flip. These are not new definitions, but they are needed. Section 4.2 finishes with a canonical form for combinatorial embeddings. In Section 4.3 theorems are established helping to avoid generating embeddings that are isomorphic. Section 4.4 gives our new exponential-time branch-and-bound algorithms `InsertEdge` and `EmbedGraph` which show how to generate all embeddings of a graph of Euler genus lower than some upper bound.

Chapter 5 gives more insight into the case of the torus (as well as surfaces of higher Euler genus) with respect to the Successive Surface Scaffolding Conjecture. First, definitions are set up in Section 5.1, and then a short summary of literature about obstructions is given in Section 5.2. Then, in Section 5.3, a theorem is proved showing every edge e of embeddings of any obstructions of any surface must have e in an essential cycle. Also, for a surface $\mathbb{S}$, an embedding of any of the minor-order minimal obstructions of $\mathbb{S}$ that are not double-wye-delta-order minimal is given for the surface $\mathbb{S}$ with a crosscap added.

Work with various embeddings is eased by having a way to visualize them. Chapter 6 defines in Section 6.1 a depiction as a plane embedding representing some higher Euler genus embedding. There can be more than one depiction for a given nonorientable embedding of Euler genus greater than one, so Section 6.2 presents some possibilities. Then in Section 6.3, a $\mathcal{O}(m^2)$ algorithm is given for finding a depiction from an input combinatorial embedding.

In Chapter 7, Section 7.1 summarizes basic results of irreducible triangulations in the literature. The main result of Section 7.2 gives the different possible structures of a region of an irreducible triangulation bound by homotopic cycles. Also, Section 7.2 provides evidence in support of the Successive Surface Scaffolding Conjecture.

In Section 8.1, we describe how an embedding can be cut open along a cycle C so that the resulting embedding can be checked for properties that determine whether C is essential. The algorithm `IsEssential` is given in Section 8.2 which formalizes cutting along a cycle as a subroutine.

Finally, in Chapter 9 a list of open questions is given. Future research is also considered in this chapter.

Chapter 2

Topological Graph Theory

The following chapter is supplementary to the rest of this document. It is optional to read, but helps in understanding the topics explored. They are modelled after the definitions in Basic Topology (see [Arm13]), and Topological Graph Theory (see [GT87]).

A *topology* endows a set X with a collection τ of subsets of X such that:

- $\emptyset$ and X are members of τ,
- any union of sets in τ is itself in τ,
- and any finite intersection of sets in τ is itself in τ.

A *topological space* is a set X together with a topology. A useful property for some topological spaces X and Y is that we can have a function $f : X \to Y$ be defined as *continuous* if for any an open set $y \subseteq Y$ we have $f^{-1}(y)$ as an open set of X.

A *curve*, or *arc* in a topological space X is the image of a continuous function $f : [0, 1] \to X$ with domain the closed interval of all real numbers x such that $0 \leq x \leq 1$. A curve is *simple* if f is also one-to-one. An arc is said to *join*, or *connect* its endpoints $f(0)$ and $f(1)$, and the interior of an arc is all other image points besides the endpoints. A *simple closed curve* is defined analogously, with the exception that $f(0) = f(1)$.

Let X and Y be topological spaces. We call X and Y *homeomorphic* if there exist continuous functions $f : X \to Y$, $g : Y \to X$ such that $f \circ g$ is the identity on Y and $g \circ f$ is the identity on X. The functions f and g are described as *homeomorphisms* between X and Y.

An *open neighbourhood* of a point x in a topological space X is simply an open set of X that contains x. A *neighbourhood* of a point x is a set of X that contains an open neighbourhood of x. A neighbourhood of a set of points $S \subseteq \mathbb{S}$ is a set $N(S) \subseteq \mathbb{S}$ where every point of S has at least one open neighbourhood contained in $N(S)$. A topological space X is *Hausdorff* when any two distinct points in X have disjoint neighbourhoods. A *surface* $\mathbb{S}$ is a connected, compact, Hausdorff topological space which is at any point locally homeomorphic to an open disc in the Euclidean plane. A surface $\mathbb{S}$ is *nonorientable* if there exists a closed curve $C \subset \mathbb{S}$ that has a neighbourhood homeomorphic to a Möbius strip, and $\mathbb{S}$ is *orientable* otherwise.

A simple closed curve C is *trivial* on a surface $\mathbb{S}$ if $\mathbb{S} - C$ is disconnected and at least one of the regions of $\mathbb{S} - C$ is homeomorphic to an open disk in the plane, and C is called *essential* otherwise. Informally, a simple closed curve is trivial when it can be continuously contracted to a point on $\mathbb{S}$. Trivial curves on orientable surfaces can be traversed in a *clockwise* or *counterclockwise* orientation—the choice of deciding which traversal is clockwise or counterclockwise depends on the representation of a surface—and we always choose to traverse in a counterclockwise direction with our representations.

Let S^1 be the set of points in a unit circle. Two simple closed curves C_1 and C_2 on a surface $\mathbb{S}$ are *homotopic* when there exists a continuous map $h : [0,1] \times S^1 \to \mathbb{S}$ where $h(0,\cdot)$ is one-to-one and has image C_1 and $h(1,\cdot)$ is one-to-one and has image C_2.

The Schönflies Theorem states that any simple closed curve C on the sphere $\mathbb{S}_0$ creates two regions of $\mathbb{S}_0 - C$ that are each homeomorphic to a closed disk. We can choose either region to be the interior of C in this case. Otherwise, we simplify discussion on other surfaces, say $\mathbb{S}$, for a trivial closed curve C which creates a region of $\mathbb{S} - C$ that is homeomorphic to a disk by calling C a *circle*, and its *interior* is the region homeomorphic to a disk. Note that because any surface $\mathbb{S}$ is Hausdorff, this guarantees the existence of two disjoint open disks on $\mathbb{S}$. Also, because C is homotopic to S^1, there exists a one-to-one continuous function $f_C : C \to S^1$, so that an identification of any two circles A and B can be obtained by $f_B^{-1} \circ f_A$. When a circle C is in an orientable surface, parameterizing f_C induces a clockwise/counterclockwise identification to S^1.

To *add a handle* to a surface $\mathbb{S}$, let two circles $A, B \subseteq \mathbb{S}$ be disjoint with disjoint interiors—delete the interiors of A and B—and when $\mathbb{S}$ is orientable, identify A in opposite orientation to B. This is illustrated in Figure 2.1. When $\mathbb{S}$ is nonorientable, it is not possible to compare orientations of trivial

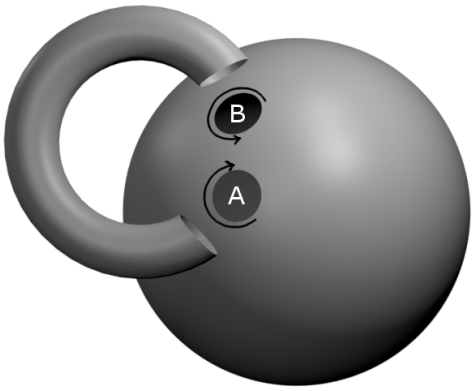


Figure 2.1: Adding a handle to a sphere

curves, and in this case A and B only need be identified.

To *add a crosscap*, let a trivial circle C be embedded in a surface $\mathbb{S}$, delete the interior of C and identify diametrically opposite points.

To *add a twisted handle*, when on an orientable surface proceed to add a handle, but identify trivial circles A and B in the same orientation. Otherwise, when on a nonorientable surface only the identification of A and B are required, and therefore a handle and twisted handle can be continuously deformed one to the other. But we emphasize that this deformation is only possible when the surface is nonorientable before adding a (twisted) handle.

We let $\mathbb{S}_0$ be the surface of the sphere, homeomorphic to the Euclidean plane R^2 with a point added for infinity. This is established by a well-known homeomorphism called stereographic projection. If we add $h \geq 0$ handles to the sphere $\mathbb{S}_0$, we obtain the *orientable* surface $\mathbb{S}_h$ of *genus* h.

If we add $k \geq 1$ crosscaps to $\mathbb{S}_0$, then we have the *nonorientable* surface $\mathbb{N}_k$ of *genus* k. The surfaces $\mathbb{S}_1$, $\mathbb{S}_2$, $\mathbb{N}_1$, $\mathbb{N}_2$ are also well known as the *torus*, the *double torus*, the *projective plane*, and the *Klein bottle*, respectively.

Note that identifying circles A and B with the same orientation on an orientable surface is actually adding two crosscaps to the surface. One intuitive way to see this is to observe that while the torus can be constructed from identifying two disjoint circles in the plane with opposite orientation, the Klein bottle can be constructed from identifying two disjoint circles in the plane with the same orientation (a twisted handle). Also, a (twisted) handle added to a nonorientable surface is the same as adding two crosscaps

(which make a Klein bottle).

It is worth mentioning that the genus of a surface can also be defined as the maximum number of non-intersecting non-homotopic essential simple closed curves that embed on the surface. The Classification Theorem of Surfaces (see [Kos80]) states that for a surface $\mathbb{S}$, it must be that $\mathbb{S}$ is homeomorphic to precisely one of $\mathbb{S}_h$ for some h or $\mathbb{N}_k$ for some k.

Note that signatures for the edges of an embedded graph as defined in Section 3.1 can be thought of as a 1-dimensional cohomology class over the graph, with coefficients in the group $\mathbb{Z}_2$. This gives a way to see how to modify the signatures to to get equivalent embeddings in surfaces.

Chapter 3

Combinatorial Embeddings

Exponential embedding algorithms operate faster if the change in genus when inserting an edge can be calculated quickly. With use of a new data structure, the calculation for the change in genus requires only $\mathcal{O}(1)$ time. This chapter first presents definitions in Section 3.1. In Section 3.2 a theorem is established for the change in Euler genus to an embedding when an edge is inserted. Then this theorem is applied when designing a new data structure called an *augmented rotation system* in Section 3.3 followed by a $\mathcal{O}(m)$ algorithm for walking the faces which use the new data structure.

3.1 Definitions

A *rotation system* for a vertex of a graph is a cyclic ordering of its neighbours. A *combinatorial embedding* of a graph G is a pair $\widetilde{G} = (\pi, \lambda)$ where $\pi = \{\pi_v | v \in V(G)\}$ is a set of rotation systems and λ is a function $\lambda : E(G) \to \{+1, -1\}$. Note that unless otherwise stated, multiple edges are not included. Intuitively, this gives an adjacency list representing the clockwise (or counterclockwise) appearance of neighbours surrounding v. We call the order that the vertices appear in π_v the *forward direction*. Then the *reverse direction* is the reverse order that vertices appear in π_v. To allow for nonorientable surfaces, a *signature* weight of $+1$ or -1 is assigned to each edge e, denoted by $\lambda(e)$. The edges weighted with -1 correspond to the edges that are embedded through an odd number of *twists*; i.e., crosscaps or twisted handles. A signature of $+1$ assigned to an edge e denotes that e uses an even number of twists. The subembedding of $\widetilde{G}$ of $+1$ weighted edges is

an embedding on an orientable surface.

In an embedding $\widetilde{G}$, for an edge $e = uv$, let $\pi_v^{+1}(u)$ be the neighbour that follows u in the rotation system of v, and let $\pi_v^{-1}(u)$ be the neighbour that precedes u. For any graph with no multiple edges,

$$x = \pi_v^{+1}(u) \Rightarrow u = \pi_v^{-1}(x).$$

In other words, if x follows u, then u precedes x in the list of neighbours of v. This is assuming $\lambda(uv) = +1$, and the case for $\lambda(uv) = -1$ is trivial. Mohar and Thomassen [MT01] describe similar notation for graph embeddings using rotation systems and signatures assigned to the edges.

A cycle of an embedding is *one-sided* if it contains an odd number of edges with negative signature, and *two-sided* if it contains an even number of edges with negative signature. To *switch* a vertex v of an embedding means to reverse the order of the rotation system of v and assign the opposite signature of each edge incident with v. Switching vertices does not change the faces of an embedding and neither does it change the property of cycles being one- or two-sided.

The image of an embedding mapped to a surface is also a simple closed curve on that surface. A simple closed curve C is two-sided if C has a neighbourhood homeomorphic to a cylinder and C is one-sided if C has a neighbourhood homeomorphic to a Möbius strip.

As described by Myrvold and Roth [RM05], we can use a rotation system and signatures to obtain all faces (or $\widetilde{G}$-faces) of an embedding. To this end, for each edge uv of the graph, an algorithm to determine the faces of an embedding uses two *records* $[(u, v), +1]$ and $[(u, v), -1]$. A *facial walk* is an iteration through the arcs of a face determined by a list of records, where for $[(u, v), r]$ the successive record in the walk is $[(v, x), r \cdot \lambda(v, x)]$, with $x = \pi_v^r(u)$. The walk terminates when the starting record is revisited. A facial walk in one direction should determine the same face as when the arcs are traversed in the opposite order. This means that the face with record $[(u, v), r]$ is the same as the one with $[(v, u), -r \cdot \lambda(u, v)]$, but walked in the reverse order. For this reason, we consider the record $[(u, v), r]$ to be equivalent to $[(v, u) - r \cdot \lambda(v, u)]$.

An *oriented walk* of length p in a combinatorial embedding $\widetilde{G}$ is a sequence of records

$$[(u_1, u_2), s_1], [(u_2, u_3), s_2], \ldots, [(u_p, u_{p+1}), s_p]$$

with $i = 1, \ldots, p$ such that $(u_i, u_{i+1}) \in E(G)$ and $s_i \in \{+1, -1\}$ and further,

for all $i = 2, \ldots, p$ we have $s_i = s_{i-1}\lambda(u_i, u_{i+1})$. A more compact notation is used, $u_1 s_1 u_2 s_2 \ldots u_p s_p u_{p+1}$, in Chapters 4 and 8.

A *closed oriented walk* is an oriented walk such that for the first record $[(u_1, u_2),\ s_1]$ and for the final record $[(u_p, u_{p+1}), s_p]$, $u_{p+1} = u_1$ and $s_1 = s_p\lambda(u_1, u_2)$. An *oriented cyclic walk* is a closed oriented walk such that either

- $u_1 u_2 \ldots u_p$ is a cycle in G; i.e., $u_1 = u_{p+1}$, and $s_1 = s_p$, which corresponds to one side of a two-sided cycle, or
- $p = 2k$ and, for $i = 1, \ldots, k$, $u_i = u_{k+i}$, and $u_1 u_2 \ldots u_k$ is a cycle in G; i.e., $u_1 = u_{k+1} = u_{p+1}$, and $s_1 = s_p \neq s_k$. This corresponds to a one-sided cycle.

A facial walk also corresponds to a closed oriented walk with the added property that for each pair of successive records $[(u, v), r]$ and $[(v, x), r\lambda(v, x)]$ it must be that $x = \pi_v^r(u)$.

To correspond with actual drawings of an embedding, we choose the clockwise appearance of neighbours of each vertex to correspond with the forward direction in combinatorial embeddings, and as a consequence, throughout our discussions and figures $r = +1$ gives a counterclockwise traversal of a face when the surface is orientable. For figures with a plane embedding, $r = +1$ determines a counterclockwise traversal of each internal face, and a clockwise traversal of the external face. Note that any face must have an even number of edges with negative signature.

If every facial walk has been considered, while keeping in mind the equivalences between records, then each record appears only once (ignore the final record in each walk). This is easy to see since in an embedding, each edge either appears on the boundary of two faces or appears twice on the boundary of one face. This does not distinguish whether an edge is traversed twice in the same or opposite direction. Hence, we can generate the facial walks by listing all possible records, marking those that are visited as they are used, and starting each walk with some unused record. Circular doubly-linked lists can be used for access to successive and previous neighbours of each vertex, as well as twin-link pointers (each node v in the adjacency list of u contains a pointer which keeps a reference to the node of u in the adjacency list of v) giving constant access time. Each linked list of a corresponding vertex v stores the adjacencies in the same rotational order that neighbours appear in π_v. Altogether, traversing all the facial walks can be done in $\mathcal{O}(m)$ time.

If there is a cycle C in an embedding such that there is an odd number of edges with negative signatures, there can be no sequence of switches that results in all positive signatures for the edges of C. In this case, the embedding is designated as *nonorientable*. Hence, a combinatorial embedding is *orientable* when every cycle has an even number of negative signature edges, and so there must exist a sequence of switches that results in every edge of the embedding assigned with a $+1$ signature.

Once the number of faces f from a combinatorial embedding $\widetilde{G}$ with c components of a connected graph G is determined, one can define the *genus* of the embedding as

$$g(\widetilde{G}) := \begin{cases} \dfrac{2c - n + m - f}{2} & \text{when } \widetilde{G} \text{ is orientable, and} \\ 2c - n + m - f & \text{when } \widetilde{G} \text{ is nonorientable.} \end{cases} \tag{3.1}$$

The *Euler genus* γ of a surface $\mathbb{S}$ with h handles, j crosscaps, and k twisted handles is defined as

$$\gamma(\mathbb{S}) := 2h + j + 2k.$$

The *Euler genus* γ of an embedding of a graph $\widetilde{G}$, either orientable or nonorientable, is defined as

$$\gamma(\widetilde{G}) := 2c - n + m - f$$

where c is the number of components, and f is the number of faces.

A *contraction* of an edge $e = uv$ in an embedding $\widetilde{G}$ is the embedding $\widetilde{G}/e$ constructed from identifying the vertices u, with $\pi_u = u_1 u_2 \dots u_{d(u)} = WvX$, and v, with $\pi_v = v_1 v_2 \dots v_{d(v)} = YuZ$, to a new vertex w such that $\pi_w = WZYX$ when e is assigned $+1$ signature, and $\pi_w = WY^R Z^R X$ with all the edges in Y and Z assigned opposite signature when e is assigned -1 signature.

A *minor* H of a graph G is such that H is isomorphic to a graph obtained from a subgraph G' of G by contracting a set of edges $A \subseteq E(G')$. This definition also includes G as a minor of itself when $G' = G$ and $A = \emptyset$. When there is no occurrence of some graph H as a minor of G we say that H is an *excluded* minor of G.

3.2 Extending an Embedding

Each face of an embedding corresponds to a finite sequence of vertices determined by a facial walk. For example, two successive records $[(u_i, v), r]$, $[(v, u_{i+r}), t]$ in a facial walk correspond to the subsequence of vertices u_i, v, u_{i+r}. Each successive pair of records in a facial walk thus determines a triple of vertices we define as an *angle* of the embedding, as similarly defined by Mohar and Thomassen [MT01]. Sometimes we denote angles with a sequence of three vertices $u_i v u_{i+1}$ where the corresponding rotation system is for v. Note that for a degree two vertex v, the subsequences $u_1 v u_2$ and $u_2 v u_1$ both correspond to different angles.

We generalize the notion of angles by considering two neighbours u and v of a vertex w with $\pi_w = XuYv$ where X and Y are subsequences of vertices, potentially empty. Then the subsequences uYv and vXu are called *compound angles* of the embedding.

Let P be an oriented walk between u_1 and u_{k+1},

$$u_1 s_1 u_2 s_2 \cdots u_k s_k u_{k+1}$$

and suppose $s_1 = r\lambda(u_1, u_2)$, with $r = +1$ or -1. Since P is an oriented walk, each $s_i = s_{i-1}\lambda(u_{i-1}, u_i)$ for $i = 2, \ldots, k$. Define the *path parity* of the rotation system of u_{k+1} with respect to that of u_1 along P to be the *same* ($r = s_k$) if P has even parity of negative signature edges, and *different* ($-r = s_k$) otherwise. Note that we need to define parity in combination with a path, or else it is not well defined. To define *angle parity* between two angles of the same face f, let it be the path parity of a path P as one of the two oriented walks between the angles on a facial walk of f. Note that by definition of facial walk, both oriented walks between two angles along a facial walk must have the same parity of negative signature edges. Therefore, assign an angle parity arbitrarily to the starting angle uvw with a path of length zero of a facial walk F and assign parity to every successive angle with respect to uvw along the facial walk F.

To *insert* an edge $e = uv$ into an embedding containing an angle wvx means to place u into the rotation system of v after w and before x. The other end of the edge e can also be inserted into an angle yuz by placing v into the rotation system of u after y and before z. It is not considered to insert an edge to form a loop, and so the two angles involved are assumed to be for distinct vertices u and v. Theorem 3.1, by Campbell, describes the change to the genus of an embedding that occurs when an edge is inserted.

This is well-known and easy, but the proof is given for the notation used in this dissertation and for the software developed.

Theorem 3.1. *Inserting an edge $e = uv$ in a combinatorial embedding $\widetilde{G}$ of genus g at angles yuz and wvx with angle parities p_1 and p_2, results in a new combinatorial embedding $\widetilde{H}$ of Euler genus $h = g + \Delta$ where if both angles are on the same face, then*

$$\begin{array}{ll} \Delta = 0 & \text{if } p_1 = p_2 \text{ and } \lambda(e) = +1, \text{ or} \\ & \quad p_1 \neq p_2 \text{ and } \lambda(e) = -1; \\ \Delta = 1 & \text{if } p_1 \neq p_2 \text{ and } \lambda(e) = +1, \text{ or} \\ & \quad p_1 = p_2 \text{ and } \lambda(e) = -1. \end{array}$$

If the two angles are on different faces, then

$$\Delta = 2 \qquad \text{regardless of the values of } p_1, p_2, \text{ and } \lambda(e).$$

In the last case, both sides of the edge e appear in the boundary of a new face that merges the two faces involved, as illustrated in Figure 3.2 on page 18.

Proof. Suppose H is a simple graph with a proper subgraph $G = H - e$ for some edge e, and that G has an embedding $\widetilde{G}$ on some surface $\mathbb{S}$. Consider the cases of inserting e into $\widetilde{G}$ to obtain an embedding of H not necessarily in $\mathbb{S}$. This involves selecting any two angles that do not appear consecutively in a facial walk, because embeddings with multiple edges are not considered. A signature for e is assigned either $+1$ or -1.

Suppose $\widetilde{G}$ has n vertices, m edges, and f faces, so that $\widetilde{G}$ has Euler genus $g = (2 - n + m - f)$. Now consider the possible Euler genus h of $\widetilde{H}$ by inserting e in different ways.

Case 1. *The genus of $\widetilde{H}$ is $h = g$.*

This case corresponds to two situations:

- Figure 3.1(a), and e has positive signature, where angles chosen for the insertion of e are in the same face with **the same** angle parity,
- Figure 3.1(a), but e has negative signature, where angles chosen for the insertion of e are in the same face with **opposite** angle parity.

Both situations can be visualized in Figure 3.1(a) on page 17, since the number of faces of $\widetilde{H}$ is one more than that of $\widetilde{G}$, but the formula for genus of $\widetilde{H}$ remains equal to $\widetilde{G}$ because H also has one more edge than G.

Case 2. *The genus of $\widetilde{H}$ is $h = g + 1$.*

This case corresponds to two situations:

- Figure 3.1(b), and e has negative signature, where the angles chosen for the insertion of e have the same angle parity (intuitively, adding a crosscap),
- Figure 3.1(b), and e has positive signature, where the angles chosen for the insertion of e have opposite angle parity.

In both situations, the number of faces remains the same, but H has one more edge, so that

$$h = 2 - n + (m + 1) - f = (2 - n + m - f) + 1 = g + 1.$$

Case 3. *The genus of $\widetilde{H}$ is $h = g + 2$*

This case corresponds to two situations, both shown in Figure 3.2. The signature of e and the angle parity between angles chosen for insertion do not affect the change to the genus. In these situations, $\widetilde{H}$ has one less face than $\widetilde{G}$, $f - 1$, and one more edge than G, $m + 1$, so that the genus of $\widetilde{H}$ is

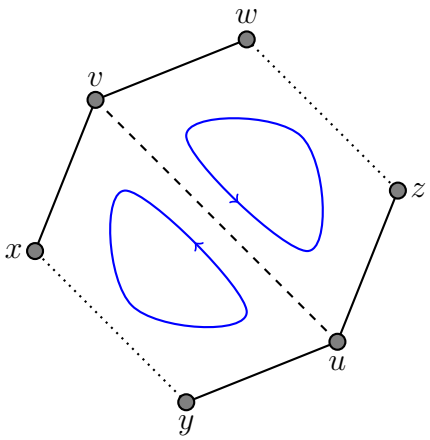
$$h = 2 - n + (m + 1) - (f - 1) = 2 - n + m - f + 2 = g + 2.$$

□

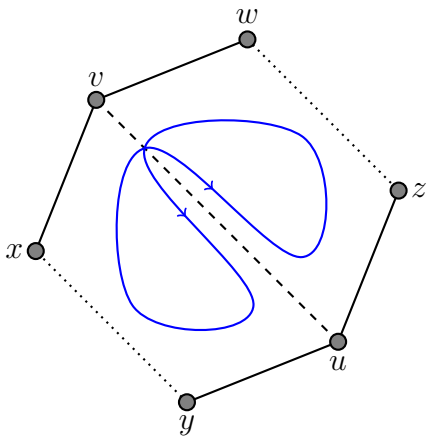
3.3 Augmented Rotation Systems

This section describes a new data structure, an *augmented rotation system*, that facilitates $\mathcal{O}(1)$ computation of the genus increase resulting from adding an edge with a given signature to an embedding. This is accomplished knowing for two angles their face labels and angle parities as per Theorem 3.1. After the augmented rotation system data structure is presented, Algorithm 3.2 gives the $\mathcal{O}(m)$ time pseudocode that updates the use of this data structure after inserting an edge.

Let the neighbours of a vertex v with its rotation system be in the order $u_1, u_2, \ldots, u_{d(v)}$. Note that reference to arbitrary vertices among neighbours of v have subscripts modulo $d(v)$, with subscript 0 replaced by $d(v)$.



(a) Either edge uv positive with its ends connecting angles with the same angle parity, or edge uv negative with its ends connecting angles of opposite angle parity.



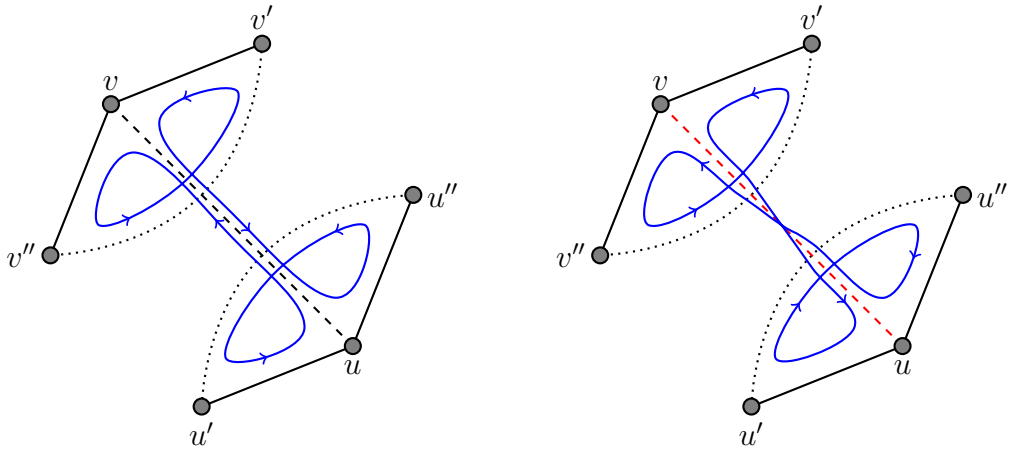
(b) Either edge uv negative and connecting angles of the same angle parity, or edge uv positive and connecting angles of opposite angle parity.

Figure 3.1: Edge uv inserted into a face of some embedding. The blue curves correspond to the new facial walks of $\widetilde{H}$. The dotted lines complete the boundary edges of old facial walks in $\widetilde{G}$, and note that we do not indicate the signatures of the edges in old facial walks.

The faces of an embedding are visited in some arbitrary order. In the new data structure it is chosen to store a face number for angle $u_i v u_{i+1}$ with the node for u_i in the adjacency list of v. The same choice is made for storing an angle parity of an angle when visited during a facial walk.

The augmented rotation system data structure `AugRotSystem` is described using object-oriented programming concepts. The other supporting object types that are needed are:

- `AdjNode` with members:
  - x: an integer vertex label,
  - u: an integer neighbour label,
  - $sign$: an integer in $\{+1, -1\}$ for the signature of the edge xu,
  - $next$: an `AdjNode` pointer to the next neighbour in the adjacency list of x,



(a) Edge uv positive.

(b) Edge uv negative.

Figure 3.2: Edge uv inserted with each end into a different face of some embedding. The blue curves correspond to the new facial walks of $\widetilde{H}$. The dotted lines complete the boundary edges of old facial walks in $\widetilde{G}$.

  - *prev*: an `AdjNode` pointer to the previous neighbour in the adjacency list of x,
  - *twin*: an `AdjNode` pointer to the node with neighbour label x in the adjacency list of u,
  - *face_num*: an integer giving the face number for the corresponding angle,
  - *angle_parity*: an integer in $\{+1, -1\}$ assigned as an angle parity in a facial walk,
  - *my_position*: an integer keeping track of the placement order in the adjacency list of x,
  - `ADJ_NULL`: a static integer constant -2 as a sentinel representing a null *face_num*.

- `UnusedNode` with members:
  - *face_number*: an integer reserving a face number as unused,
  - *next*: an `UnusedNode` pointer to the next node in a list,

and a constructor method that inserts a new `UnusedNode` with input *face_number* to the front of an already existing input list.

- `UnusedFaces` with members:
  - *max_face*: an integer keeping track of the maximum number of faces used in an `AugRotSystem`,
  - *start_unused*: an `UnusedNode` pointer to the first `UnusedNode` in a list,

  and methods:
  - `GetFaceNumber()`: returns either the *face_number* of the first `UnusedNode` in *start_unused* while removing this node from *start_unused*, or the *max_face* while incrementing *max_face* by one,
  - `ReturnFaceNumber( int`f `)`: creates a new `UnusedNode` with face_number f and inserts this to the front of *start_unused*.
- `AdjMatrix` with members:
  - n: an integer for the number of rows, and the number of columns,
  - A: a two-dimensional integer array,

  and methods
  - `SpanningTree()`: returns an `AdjMatrix` T that is a spanning tree of the graph defined by A determined by a depth-first-search, starting at the vertex labelled 0.

Then an augmented rotation system object `AugRotSystem` has members:

- M: an `AdjMatrix` which holds the adjacency matrix for the underlying graph,
- T: an `AdjMatrix` which holds the depth-first-search spanning tree for the underlying graph,
- `NMAX`: a static integer constant giving an upper bound on the number of vertices in the underlying graph,
- n: an integer giving the number of vertices in the underlying graph,

- m: an integer giving the number of edges in the underlying graph,
- f: an integer giving the number of faces in this embedding,
- g: an integer for the Euler genus of this embedding,
- *num_negative*: an integer for the number of negative signature edges in this embedding,
- *unused*: an `UnusedFaces` object to manage the assignment of *face_num* of each `AdjNode`,
- *degree*: an integer array keeping the degree of each vertex, with the index corresponding to vertex label,
- V: an array of `AdjNode`, with the index corresponding to vertex label, so that $V[i]$ corresponds to the start of an adjacency list for vertex i,

and methods:

- `AugRotSystem( AdjMatrix` T `)`: constructor which initializes members and sets adjacency lists in V for some spanning tree T, in preparation for the algorithm that finds embeddings,
- `InitPositions()`: sets the *my_position* for each `AdjNode` in V,
- `GetTwinLinks()`: for each `AdjNode` in V, sets the *twin* of an `AdjNode` in the adjacency list of x with vertex label u to be a pointer to the `AdjNode` in the adjacency list of u with vertex label x,
- `NewGenus( AdjNode` $ptr1$, `AdjNode` $ptr2$, `int` *new_sign* `)`: returns the updated genus of this `AugRotSystem` embedding for an edge considered to be inserted at angles $ptr1$ and $ptr2$ with signature *new_sign* as per Theorem 3.1,

as well as other methods given as pseudocode later to explain in greater detail:

- `WalkOneFace` (Algorithm 3.2)
- `WalkFaces` (Algorithm 3.3)
- `AddEdge` (Algorithm 3.4)

- `RemoveEdge` (Algorithm 3.5)

The angle parity can differ depending on which angle a facial walk starts. An initial angle, which corresponds to a neighbour in an adjacency list, is chosen to start a facial walk and given angle parity +1. Each successive angle that is visited is assigned an angle parity of $(-1)^k$ where k is the number of edges traversed with signature -1 when walking the face from the initial angle in the +1 direction.

Algorithm 3.2. *Walk One Face*

INPUT:

- an AdjNode *start_u* pointer to one of the nodes in the adjacency list of this `AugRotSystem` corresponding to the angle vuw where $v = start\_u.u$, $u = start\_u.twin.u$, and $w = start\_u.next.u$,
- a boolean value *unwalk* that when `true` sets *face_num* of angles visited to `AdjNode.ADJ_NULL`.

ACTION:

- a face value is assigned to every angle of the walked face,
- an angle parity value is computed for every angle of the walked face.

```
WalkOneFace (start_u, unwalk) {
 1 ) int face_num;
 2 ) if ( unwalk ) {
 3 )   if ( start_u.face_num ≠ AdjNode.ADJ_NULL )
 4 )     unused.ReturnFaceNumber( start_u.face_num );
 5 )   face_num = AdjNode.ADJ_NULL;
 6 ) } else {
 7 )   face_num = unused.GetFaceNum();
 8 ) }
 9 ) AdjNode ptr = start_u;
10 ) int dir = 1;
11 ) do {
12 )   if ( dir == 1 ) {
13 )     ptr.face_num = face_num;
14 )     ptr.angle_parity = dir;
```

```
15 )       ptr = ptr.next.twin;
16 )       dir = dir * (ptr.sign);
17 )     } else {
18 )       ptr = ptr.prev;
19 )       ptr.face_num = face_num;
20 )       ptr.angle_parity = dir;
21 )       ptr = ptr.twin;
22 )       dir = dir * (ptr.sign);
23 )     }
24 ) } while (ptr ≠ start_u or dir ≠ 1);
}
```

Algorithm 3.3. *Walk Faces*

ACTION:

- a face value is assigned to every angle of this `AugRotSystem`,
- an angle parity value is assigned to every angle of this `AugRotSystem`.

```
WalkFaces() {
 1 ) AdjNode ptr;
 2 ) int i, j;
 3 ) unused = new UnusedFaces();
 4 ) f = 0;
 5 ) // set all the face numbers to be null
 6 ) for (i = 0; i < n; i++) {
 7 )   ptr = V[i];
 8 )   for (j = 0; j < degree[i]; j++) {
 9 )     ptr.face_num = AdjNode.ADJ_NULL;
10 )     ptr = ptr.next;
11 )   }
12 ) }
13 ) // walk all the faces
14 ) for (i = 0; i < n; i++) {
15 )   ptr = V[i];
16 )   for (j = 0; j < degree[i]; j++) {
17 )     if ( ptr.face_num == AdjNode.ADJ_NULL ) {
18 )       WalkOneFace( ptr, false );
```

```
19 )          f++;
20 )        }
21 )        ptr = ptr.next;
22 )     }
23 ) }
24 ) g = 2 − n + m − f;
}
```

Algorithm 3.4. *Add Edge*

INPUT:

- an AdjNode pointer $ptr1$ corresponding to an angle of this `AugRotSystem`,
- an AdjNode pointer $ptr2$ corresponding to an angle of this `AugRotSystem`,
- an integer new_sign for the signature of the edge to be inserted.

ACTION:

- this `AugRotSystem` updated to include an edge inserted with one end at $ptr1$ and the other end at $ptr2$.

```
AddEdge( ptr1, ptr2, new_sign ) {
1 ) AdjNode n1, n2;
2 ) int u, v;
3 ) // reset face data
4 ) WalkOneFace( ptr1, true );
5 ) f--;
6 ) if ( ptr2.face_num ≠ AdjNode.ADJ_NULL ) {
7 )   WalkOneFace( ptr2, true );
8 )   f--;
9 ) }
10 ) u = ptr1.twin.u;
11 ) v = ptr2.twin.u;
12 ) // insert new nodes
13 ) n1 = new AdjNode( u, v, new_sign, ptr1, ptr1.next );
14 ) n2 = new AdjNode( v, u, new_sign, ptr2, ptr2.next );
15 ) n1.twin = n2;
16 ) n2.twin = n1;
```

```
17 ) degree[u]++;
18 ) degree[v]++;
19 ) m++;
20 ) if ( new_sign == -1 )
21 )   num_negative++;
22 ) // update face data
23 ) WalkOneFace(ptr1, false);
24 ) f++;
25 ) if ( n1.face_num == AdjNode.ADJ_NULL ) {
26 )   WalkOneFace( n1, false );
27 )   f++;
28 ) }
29 ) g = 2 - n + m - f;
}
```

Algorithm 3.5. *Remove Edge*

INPUT:

- an AdjNode pointer *ptr* corresponding to an edge of this `AugRotSystem`.

ACTION:

- this `AugRotSystem` with the edge corresponding to *ptr* removed.

```
RemoveEdge( ptr ) {
 1 ) AdjNode n1, n2, n3, n4;
 2 ) int u, v;
 3 ) // reset face data
 4 ) WalkOneFace( ptr, true );
 5 ) f--;
 6 ) if ( ptr.prev.face_num ≠ AdjNode.ADJ_NULL ) {
 7 )   WalkOneFace( ptr.prev, true );
 8 )   f--;
 9 ) }
10 ) if ( ptr.sign == -1 )
11 )   num_negative--;
12 ) u = ptr.twin.u;
13 ) v = ptr.u;
14 ) n1 = ptr.prev;
```

```
15 ) n2 = ptr.next;
16 ) n3 = ptr.twin.prev;
17 ) n4 = ptr.twin.next;
18 ) // remove the nodes corresponding to edge uv
19 ) n1.next = n2;
20 ) n2.prev = n1;
21 ) n3.next = n4;
22 ) n4.prev = n3;
23 ) degree[u]--;
24 ) degree[v]--;
25 ) m--;
26 ) // update face data
27 ) WalkOneFace( n1, false );
28 ) f++;
29 ) if ( n3.face_num == AdjNode.ADJ_NULL ) {
30 )   WalkOneFace( n3, false );
31 )   f++;
32 ) }
33 ) g = 2 - n + m - f;
}
```

Chapter 4

Embedding Algorithms

This chapter begins with a literature review of embedding algorithms in Section 4.1. Section 4.2 defines equivalence relations called flip, switch, and switch-flip between combinatorial embeddings, and follows this with a canonical form for combinatorial embeddings. In Section 4.3, a number of theorems are established helping to avoid generating some repetitions of embeddings. Finally, in Section 4.4, exponential-time algorithms are given for generating all embeddings of a graph with genus at most some input constant.

4.1 Literature Review

The earliest planar-embedding algorithms rely on planarity-testing algorithms. The first planarity-testing algorithm with polynomial runtime was originally proposed by Auslander and Parter [AP61], then corrected by Goldstein [Gol63], and independently by Bader [Bad64]. Hopcroft and Tarjan [HT74] refined these to present the first linear time planarity-testing algorithm, but the extension to a planar-embedding algorithm is complicated enough that Mehlhorn, Mutzel [MM96], Chiba, Nishizeki, Abe, and Ozawa [CNAO85] outline implementation details. The runtime results also depend on finding the blocks of a graph in linear time, as described by Ullman, Aho, and Hopcroft [UAH74], since a planar graph contains only planar blocks.

Demoucron, Malgrange, and Pertuiset [DMP64] were the first to design a $O(n^3)$ time planar-embedding algorithm that is theoretically simple to understand. This partly inspired Lempel, Even, and Cederbaum [LEC66] to create the so-called *Vertex Addition Algorithm*, which can be implemented

in linear-time as shown by Booth and Lueker [BL76].

Another planar-embedding algorithm is worth mentioning because it has an efficient implementation that is available in the Public Implementation of a Graph Algorithm Library Editor (PIGALE) software [dFdM02]. This algorithm was finally detailed by de Fraysseix, de Mendez, Rosenstiehl, and Brandes [dFdMR06, dF08, Bra09].

Some planar-embedding algorithms are worth mentioning for their simplicity, such as one by Klotz [Klo89] that has $O(n^2)$ runtime, or Boyer and Myrvold [BM04] which has $O(n)$ runtime. Yet there are still fairly new planar-embedding techniques, e.g. Schmidt [Sch13] attempts to simplify details for as concise a linear algorithm as possible.

There are currently only two correct published algorithms for embedding graphs in the projective plane. As with planar embedding algorithms, a lower time complexity seems to force added difficulty in implementation.

Perunicic and Duric [PD85] proposed the first projective-plane embedding algorithm with $O(n^3)$ runtime, but it does not always work correctly. Mohar [Moh93] has explained that their algorithm does not always produce an embedding, and so he designed a different linear-time algorithm which guarantees one, although it is difficult to implement.

Myrvold and Roth [RM05] have a simpler $O(n^2)$ algorithm that is similar to Mohar's approach [Moh93] for embedding on the projective plane. First, a graph G is tested for planarity, and if found to be planar they simply apply a planar-embedding algorithm, among any of those with $O(n^2)$ runtime or faster. The algorithm starts by finding a Kuratowski subgraph K and considers every emebbing $\widetilde{K}$ in the projective plane. Then for each $\widetilde{K}$, there is only a constant number of ways to embed the set of K-bridges that embed in more than two faces, and for the remaining bridges, an adaptation of a 2-Sat algorithm assigns them to faces.

Myrvold and Kocay [MK11] exposed errors from past attempts at writing a polynomial-time algorithm for embedding graphs in the torus, namely: Filotti [Fil78, Fil80]; Filotti, Miller, and Reif [FMR79]; and Djidjev, and Reif [DR91]. The errors proved that the given algorithms are either wrong or they actually run in exponential time, and there is no apparent way to fix these algorithms so that they can run in polynomial time.

Juvan, Marinček, and Mohar [JMM95] have a linear-time algorithm to embed graphs on the torus. The ideas from this paper clearly influenced Mohar [Moh96, Moh99] to use similar techniques to be able to embed a graph in an arbitrary surface in linear-time. As far as is known, these algorithms

currently have no correct implementation, and implementing them would be very difficult.

Gagarin, Kocay, and Neilson [GKN03], Neufeld, and Myrvold [NM97], Woodcock [Woo06], and Yu [Yu14] each have an exponential-time embedding algorithm for the torus. The strategy for such algorithms is to have a manageable implementation and yet still have fast enough results on smaller input. And finally, Gagarin and Kocay [GK02] have a linear-time algorithm for embedding graphs with a K_5 subdivision and no $K_{3,3}$ subdivision on the torus.

4.2 Equivalence Relations

The following equivalence relations are defined in order to avoid duplications when generating embeddings of a graph. Let $\widetilde{G}$ and $\widetilde{H}$ be combinatorial embeddings. Naturally, $\widetilde{G}$ and $\widetilde{H}$ are *identical* when they have the same cyclic ordering of neighbours for every vertex and the same signature for every edge. To enable easier comparison of rotation systems, we standardize them by choosing the smallest labelled neighbour to appear first in each rotation system.

Recall the definition of switching a vertex in Section 3.1. For $A \subseteq V(G)$, let $S(\widetilde{G}, A)$ denote the new embedding obtained from $\widetilde{G}$ by switching the vertices of A. Define $\widetilde{G}$ and $\widetilde{H}$ to be *switch-equivalent*, denoted $\widetilde{G} \overset{S}{=} \widetilde{H}$, when there exists some subset of vertices $A \subseteq V(G)$ such that $S(\widetilde{G}, A)$ is identical to $\widetilde{H}$.

Recall that orientable embeddings have every cycle with an even number of negative signature edges. For orientable $\widetilde{G}$, there exists a set of vertices $A \subsetneq V(G)$ such that $S(\widetilde{G}, A)$ has all edges with positive signature. If $\widetilde{G}$ has edges with -1 signature, then $A \neq \emptyset$ and $A \neq V$ since the signatures of $S(\widetilde{G}, V)$ are the same as in $\widetilde{G}$.

Let $F(\widetilde{G})$ denote the embedding identical to $S(\widetilde{G}, V)$, and call this the *flip* of $\widetilde{G}$ (it flips the order of each rotation system and has the same signatures). Further, $\widetilde{G}$ and $\widetilde{H}$ are *flip-equivalent*, denoted $\widetilde{G} \overset{F}{=} \widetilde{H}$, when either $\widetilde{G}$ or $F(\widetilde{G})$ is identical to $\widetilde{H}$. Embeddings $\widetilde{G}$ and $\widetilde{H}$ are *switch-flip-equivalent*, denoted $\widetilde{G} \overset{SF}{=} \widetilde{H}$, if there exists some subset of vertices $A \subseteq V(G)$ such that $S(\widetilde{G}, A)$ is flip-equivalent to $\widetilde{H}$.

Let α be an element of the symmetric group S_n, and let $P(\widetilde{G}, \alpha)$ be the

embedding resulting from permuting the labels of $V(G)$ according to α, but keeping the same signatures on all edges. We say embeddings $\widetilde{G}$ and $\widetilde{H}$ are *permute-equivalent*, denoted by $\widetilde{G} \stackrel{P}{=} \widetilde{H}$, when there exists some permutation $\alpha \in S_n$ such that $P(\widetilde{G}, \alpha)$ is identical to $\widetilde{H}$.

Two embeddings $\widetilde{G}$ and $\widetilde{H}$ are *switch-isomorphic*, $\widetilde{G} \stackrel{PS}{=} \widetilde{H}$, if there exists a permutation α such that $P(\widetilde{G}, \alpha)$ is switch-equivalent to $\widetilde{H}$. Two embeddings $\widetilde{G}$ and $\widetilde{H}$ are *flip-isomorphic*, denoted $\widetilde{G} \stackrel{PF}{=} \widetilde{H}$, if there exists a permutation α such that $P(\widetilde{G}, \alpha)$ is flip-equivalent to $\widetilde{H}$. An embedding $\widetilde{G}$ is *chiral* if $\widetilde{G}$ is not switch-isomorphic to $F(\widetilde{G})$. Two embeddings $\widetilde{G}$ and $\widetilde{H}$ are *isomorphic*, $\widetilde{G} \simeq \widetilde{H}$, if there exists a permutation α such that $P(\widetilde{G}, \alpha)$ is switch-flip-equivalent to $\widetilde{H}$.

We present Algorithm 4.3 as a means to generate, for an input graph G, all different orientable, or exclusively, nonorientable, combinatorial embeddings of genus smaller than some given upper bound. This algorithm takes exponential time, because it involves iterating through all the possible rotation system orderings for the vertices in one spanning tree of G, followed by inserting the remaining edges in all possible ways without exceeding the desired genus.

Our motivation for the following notation is to avoid generating switch-flip-equivalent embeddings. We define the format of the output for each combinatorial embedding, as well as an ordering on the output. Let $u_1, u_2, \ldots,$ $u_{d(u)}$ be the neighbours of vertex u listed in the cyclic order for the rotation system of u starting from u_1 as the smallest label among neighbours of u, and where $d(u)$ is the degree of u. Let the *vertex-sequence* for vertex u of an embedding $\widetilde{G}$ be given by

$$\Big(d(u), u_1, \lambda(u, u_1), u_2, \lambda(u, u_2), \ldots, u_{d(u)}, \lambda(u, u_{d(u)})\Big).$$

However, replace every $+1$ signature with a 0 character and every -1 signature with a 1 character in each vertex-sequence. Then define the *embedding sequence* of $\widetilde{G}$ to start with n followed by the vertex sequences for vertices 0 to $n-1$. Then comparing two different embeddings is done by comparing one embedding sequence lexicographically to another embedding sequence.

4.3 Reducing Repetition

Assume the vertices of input graph G have labels 0 to $n-1$. Note that for a vertex v with degree d, the number of cyclic permutations of the sequence of neighbours of v is $(d-1)!$. In general, if G has degree sequence $d_0, d_1, \ldots, d_{n-1}$, then there are $\prod_{i=0}^{n-1}(d_i-1)!$ choices for ordering neighbours, and so it is obvious some strategy is needed to focus on the embeddings that are of interest.

To avoid equivalent cyclic permutations the first neighbour listed is chosen to be the neighbour with minimum vertex label. Further, for any combinatorial embedding, there are 2^n ways to get switch-equivalent embeddings using switching of vertices. Our efforts benefit from restricting computations to representatives from the different equivalence classes of combinatorial embeddings established by our definition of switch-flip-equivalent in Section 4.2.

Lemma 4.1. *(see [MT01]) For some fixed spanning tree T of G, the embedding $\widetilde{G}$ is switch-equivalent to a combinatorial embedding where the edges of T are assigned positive signature.*

Proof. Select a root r to be a vertex of T and assign all vertices of T an integer index with the order each vertex is visited in a breadth-first-search traversal of T starting at r. Then for each non-root vertex v in T in the index order, if edge $(v, parent(v))$ has -1 signature, perform a switch at v. □

Note that it is never necessary to switch all the vertices of an embedding to get a spanning tree with $+1$ edges, so that the flip of an embedding is never used for this purpose. For an embedding $\widetilde{G}$ and a spanning tree T, define $\widetilde{G}$ to be in *standard form* with respect to T, if the edges of T are assigned $+1$ in $\widetilde{G}$.

Theorem 4.2. *An embedding $\widetilde{G}$, in standard form with respect to some spanning tree T, is orientable if and only if $\widetilde{G}$ has no -1 signature edges.*

Proof. Suppose there is an orientable embedding $\widetilde{G}$ in standard form with $+1$ signature spanning tree T. Since any edge e of G not in T forms a cycle C when inserted into T, e cannot be assigned -1 else the parity of the number of -1 edges in C is odd, which would contradict $\widetilde{G}$ orientable.

The converse is trivial. □

Algorithm 4.3 recurses through all possible planar embeddings of a chosen spanning tree T, so a good strategy is to try to find a spanning tree T that has small maximum degree. Perhaps a hamiltonian path is possible, but finding one of these is a hard problem. Instead, one can choose a depth-first-search tree T in the hopes of decreasing the maximum degree. The first step in generating all embeddings is to apply Algorithm 4.3—starting with an empty subembedding $\widetilde{H}$—inserts all edges of T into $\widetilde{H}$. Then recursive calls are made in order to permute the rotation systems of each vertex of $\widetilde{H}$.

For each way to embed T, the remaining edges of G must be inserted in all possible ways. This is done recursively via Algorithm 4.4. Note that if the rotation system of 0 for $\widetilde{G}$ is $a_1, a_2, \ldots, a_{d(0)}$ then the rotation system for 0 in $F(\widetilde{G})$ must be $a_1, a_{d(0)}, \ldots, a_2$. Therefore, once all edges are inserted and the genus of the resulting embedding $\widetilde{H}$ is small enough, an additional rule is checked on the rotation system of vertex 0: restrict the second neighbour listed in the rotation system of the root to have a smaller label than that of the last neighbour. This ensures that only one representative is processed from the class of flip-equivalent embeddings. Maintaining the edges of T with $+1$ signature ensures only one representative from each class of switch-equivalent embeddings of G is processed. Together, these strategies ensure one representative from each class of switch-flip-equivalent embeddings of G.

Algorithm 4.4 can be used to output a representative for each switch-flip-equivalent class of embeddings. However, The next step is to determine if two different switch-flip equivalent embeddings are in fact permute-equivalent to each other, which follows.

A *clockwise breadth-first search* (CBFS) takes as input a root vertex r and a first child c that is a neighbour of r. A breadth-first search is then used to relabel the graph subject to the following restrictions. For the root r the breadth-first search starts at c and proceeds "clockwise" by selecting the next neighbour of r in the forward direction from c in the rotation system of r. For a vertex $v \neq r$, the breadth-first search starts at the breadth-first-search parent of v and proceeds clockwise. Note that switches are performed so that edges of the CBFS tree all have positive signature.

For each possible pair of r and c, all possible CBFS labelings are considered. Then the same is done on $F(\widetilde{G})$. The lexicographical minimum of the sequences of these labelled embeddings is set as the canonical form of $\widetilde{G}$.

There is a different breadth-first-search tree for every choice of edge uv with u as root and v as first child, and two directions to consider visiting

vertices via the rotation systems, so that in total there are $4m$ breadth-first-search trees to consider. It is sufficient to consider the breadth-first-search trees where a root has minimum degree δ and first child has the smallest degree δ' among the neighbours of vertices of degree δ. Then it is guaranteed the vertex-sequences of 0 will be the same for any such minimum-degree vertices. If vertex r has degree d_0 and the first child c of r has degree d_1, then the sequence of the embedding will begin with $(n, d_0, 1, 2, \ldots, (d_0 - 1), d_0, d_1, 0, \ldots)$.

Note that m is bound by a linear function of n if the graphs are restricted to those of bounded genus. The canonical sequence is chosen as a representative for the class of embeddings which are isomorphic to $\widetilde{G}$. The output of Algorithm 4.3 can then be analyzed using CBFS so that each embedding is output in its isomorphic canonical form. Given a file of canonical forms of embeddings printed one per line, the file can be sorted and duplicates can be removed. Breadth-first-search has $\mathcal{O}(m)$ run time for low genus graphs, and because a breadth-first-search for each root, child, and each direction is performed, our isomorphism algorithm runs in $\mathcal{O}(m^2)$ time. It should be noted that many breadth-first-search trees could sometimes be ignored, because the corresponding lexicographically smallest sequence necessarily occurs with a root of minimum degree.

Pseudocode for finding a canonical form is formalized in Algorithm 4.5 as a method of the `AugRotSystem` data structure. The subroutine `CBFS` on Line 6 takes as parameters an `AugRotSystem` G, a root vertex r of the `AugRotSystem`, a first child c of r, and sets H as the CBFS labelled embedding of G. The embeddings sequences which have the same canonical form show that such embeddings are isomorphic.

4.4 Exponential Embedding Algorithm

The following objects are also needed to further implement `AugRotSystem`:

- `ChordNode` with members:
  - u: an integer for a vertex label corresponding to one end of an edge in a graph,
  - v: an integer for a vertex label corresponding to the other end of an edge in a graph,

  - *next*: a `ChordNode` pointer to the next node in a list,

  and method:

  - `ChordList( AdjMatrix` M, `AdjMatrix` T `)`: this method is static, and returns a `ChordNode` pointer to the start of a list corresponding to edges in M that are not in T.

- `Permutation` with members:

  - n: an integer for the number of vertices involved in this permutation,
  - p: an integer array of size n,

  and methods:

  - `Permutation( int` nv `)`: a constructor that initializes p as an array of size nv,
  - `LexNexPerm()`: permutes the elements of p to the next lexicographic ordering and returns `true` if such a next ordering exists, and otherwise the elements of p are returned to their smallest lexicographic order and returns `false`.

The following algorithms are included in the methods of an `AugRotSystem` object and described with pseudocode in greater detail:

- `TreeEmbeddings` (Algorithm 4.3),
- `EmbedChords` (Algorithm 4.4),
- `CanonicalForm` (Algorithm 4.5),

and, finally, `AugRotSystem` also includes the following method:

- `GetEmbeddings( AdjMatrix` inM, `int` max_genus, `boolean` *orientable* `)`: this method is static, and returns the number of combinatorial embeddings found for the input graph inM of Euler genus at most max_genus that are orientable when *orientable* is set to `true` and nonorientable otherwise.

Algorithm 4.4 generates one representative from each class of switch-flip equivalent embeddings of an augmented rotation system. Embeddings of low genus are desired, and such embeddings must have all their subembeddings of the same genus or smaller. An intuitive place to start with Algorithm 4.3 is to construct embeddings that are obviously planar by first embedding a depth-first-search spanning tree in the plane in all possible ways. For each way that the spanning tree is embedded, try to insert the remaining edges one at a time while maintaining a low enough genus. For a remaining edge uv, for each angle of u, and for each angle of v from the current subembedding, consider possibly inserting uv into the subembedding. If inserting uv for these angles does not increase the genus over the threshold, then the edge is inserted. Otherwise, the edge is not inserted and another pair of unique angles for the edge is chosen. Once all edges are inserted, the graph is embedded. At each stage of the backtrack, the algorithm chooses an edge e that is not yet embedded. It tries all possible placements for e that do not increase the genus so that it is larger than the desired genus. Note that embeddings of a graph whose genus is smaller than the target genus are also constructed. Some duplicate permute-equivalent embeddings are obviously repeated in the output of Algorithm 4.4, and it depends on which automorphisms of G maintain properties of faces in an embedding. The canonical form generated by Algorithm 4.5 is used to test for these duplicates in an effort to have exactly one representative from each class of isomorphic embeddings.

Algorithm 4.3. *Tree Embeddings*

INPUT:

- a `ChordNode` pointer *chords* to the start of a linked list corresponding to the remaining edges to be inserted in this `AugRotSystem`,
- an integer upper-bound `MAX_GENUS`,
- a boolean variable *orientable* that when `true` output embeddings are orientable, and when `false` output embeddings are nonorientable,
- an integer *level* to track the depth of recursion.

ACTION:

- each embedding of the spanning tree T of this `AugRotSystem` is visited.

```
TreeEmbeddings(chords, MAX_GENUS, orientable, level) {
 1 ) AdjNode ptr;
 2 ) Permutation q;
 3 ) boolean more;
 4 ) int num_embeddings = 0;
 5 ) if ( level == n ) {
 6 )   GetTwinLinks();
 7 )   num_embeddings = EmbedChords( chords, MAX_GENUS,
       orientable, level );
 8 )   return num_embeddings;
 9 ) }
10 ) // only one permutation to consider
11 ) if ( degree[level] ≤ 2 ) {
12 )   num_embeddings += TreeEmbeddings( chords, MAX_GENUS,
       orientable, level + 1 );
13 )   return num_embeddings;
14 ) }
15 ) // go through the cyclic permutations of the vertex level
16 ) q = new Permutation( degree[level] − 1 );
17 ) ptr = V[level];
18 ) // permute all adjacencies except the first one
19 ) for (int i = 0; i < degree[level] − 1; i++) {
20 )   ptr = ptr.next;
21 )   q.p[i] = ptr.u;
22 ) }
23 ) more = true;
24 ) while ( more ) {
25 )   num_embeddings += TreeEmbeddings( chords, MAX_GENUS,
       orientable, level + 1 );
26 )   // note that the last permutation visited is the
       identity which restores the adjacency list of level
       before returning
27 )   more = q.LexNextPerm();
28 )   ptr = V[level];
29 )   for (int i = 0; i < degree[level] − 1; i++) {
30 )     ptr = ptr.next;
31 )     ptr.u = q.p[i];
32 )   }
```

```
33 ) }
34 ) return num_embeddings;
}
```

Algorithm 4.4. *Embed Chords*

INPUT:

- a `ChordNode` pointer *chords* to the start of a linked list corresponding to the remaining edges to be inserted in this `AugRotSystem`,
- an integer upper-bound `MAX_GENUS`,
- a boolean variable *orientable* that when `true` output embeddings are orientable, and when `false` output embeddings are nonorientable,
- an integer *level* to track the depth of recursion.

OUTPUT:

- prints at least one representative from each class of switch-flip-equivalent embeddings of the graph corresponding to this `AugRotSystem` of genus at most `MAX_GENUS`,
- returns the number of embeddings printed.

```
EmbedChords( chords, MAX_GENUS, orientable, level ) {
 1 ) ChordNode temp;
 2 ) AdjNode ptr1, ptr2;
 3 ) int u, v, last_sign, new_g;
 4 ) int num_embeddings;
 5 ) // for use in code defined in Chapter 6
 6 ) Depiction d;
 7 ) if ( chords == NULL ) {
 8 )   if ( !orientable and num_negative == 0 )
 9 )     return 0;
10 )   ptr1 = V[0].next;
11 )   ptr2 = V[0].prev;
12 )   // is this embedding canonical?
13 )   if ( ptr2.u < ptr1.u )
14 )     return 0;
```

```
15 )   d = new Depiction( this );
16 )   return 1;
17 ) }
18 ) num_embeddings = 0;
19 ) // remove edge from front of list of chords
20 ) u = chords.u;
21 ) v = chords.v;
22 ) temp = chords;
23 ) chords = chords.next;
24 ) // for all possible ways to select two angles
25 ) if ( orientable )
26 )   last_sign = 1;
27 ) else
28 )   last_sign = -1;
29 ) ptr1 = V[u];
30 ) for (int i = 0; i < degree[u]; i++) {
31 )   ptr2 = V[v];
32 )   for (int j = 0; j < degree[v]; j++) {
33 )     for (int new_sign = 1; new_sign ≥ last_sign;
        new_sign -= 2) {
34 )       new_g = NewGenus( ptr1, ptr2, new_sign );
35 )       if ( new_g ≤ MAX_GENUS ) {
36 )         AddEdge( ptr1, ptr2, new_sign );
37 )         num_embeddings += EmbedChords( level + 1,
            chords, MAX_GENUS, orientable );
38 )         RemoveEdge( ptr1.next );
39 )       }
40 )     }
41 )     ptr2 = ptr2.next;
42 )   }
43 )   ptr1 = ptr1.next;
44 ) }
45 ) chords = temp;
46 ) return num_embeddings;
}
```

Algorithm 4.5. *Canonical Form*

OUTPUT:

- an `AugRotSystem` A with lexicographical minimum embedding sequence among all CBFS labellings of this `AugRotSystem` and its flip.

```
CanonicalForm() {
 1 ) AugRotSystem A, H;
 2 ) boolean first = true;
 3 ) for (int r = 0; r < n; r++) {
 4 )   AdjNode ptr = V[r];
 5 )   for (int j = 0; j < degree[r]; j++) {
 6 )     CBFS( this, r, ptr.u, H );
 7 )     if ( first or embedding sequence of H < embedding
         sequence of A ) {
 8 )       A = a copy of H;
 9 )       first = false;
10 )     }
11 )     ptr = ptr.next;
12 )   }
13 ) }
14 ) Let F = a copy of the flip of this AugRotSystem;
15 ) for (int r = 0; r < n; r++) {
16 )   AdjNode ptr = F.V[r];
17 )   for (int j = 0; j < F.degree[r]; j++) {
18 )     CBFS( F, r, ptr.u, H );
19 )     if ( embedding sequence of H < embedding sequence
         of A ) {
20 )       A = a copy of H;
21 )     }
22 )     ptr = ptr.next;
23 )   }
24 ) }
25 ) return A;
}
```

One application of an implementation of Algorithm 4.3 confirms that the nonorientable genus is not additive over blocks of a graph. The nonorientable genus g' and the orientable genus g of G must satisfy $g' \leq 2g + 1$ [Arc96]. Consider G the one-vertex identification of two copies of K_7, and note that K_7 does not embed on the Klein bottle, but embeds on $\mathbb{N}_3$. In this case, the

sum of the nonorientable genera of each block of G is 6, yet K_7 embeds on the torus, so that the orientable genus of G is 2. Therefore, the nonorientable genus of G must be less than or equal to 5. An embedding of G on the double torus is given in Figure 4.1. Note that any one of the edges in the embedding of G in Figure 4.1 can be assigned negative signature, which shows that G can embed on $\mathbb{N}_5$. The implementation was used to confirm that there are no embeddings of G on the surface $\mathbb{N}_4$.

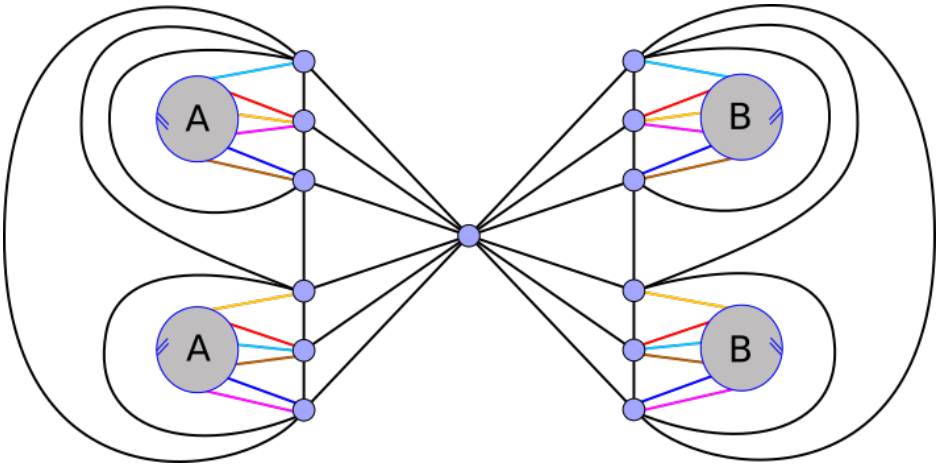


Figure 4.1: An image of an embedding of two blocks of K_7 in $\mathbb{S}_2$ with two handles shown as disks A and B with orientations marked as indicated in blue arrows. The edges using the handles are matched by colour.

Chapter 5

Obstructions

This chapter considers theoretical evidence for the Successive Surface Scaffolding Conjecture involving the case of the torus (as well as surfaces of higher Euler genus). First, definitions are given in Section 5.1. Then a literature review of obstructions is presented in Section 5.2. Lastly, in Section 5.3, a general proof for a property of embeddings of any obstructions of any surface is given. For a surface $\mathbb{S}$, an embedding of a graph in a subset of the minor-order minimal obstructions of $\mathbb{S}$ is given for the surface $\mathbb{S}$ with a crosscap added. Some special classes of obstructions for the torus are also given embeddings on the surface $\mathbb{N}_3$.

5.1 Definitions

A *topological obstruction* G for a surface $\mathbb{S}$ is a graph with neither isolated nor degree two vertices that does not embed in $\mathbb{S}$, yet $G - e$ does embed for any edge e. A *minor* obstruction has the added property that for any single edge contraction, the resulting graph embeds on $\mathbb{S}$. Bodendiek and Wagner [BW89b] proved that the set of topological, and hence minor, obstructions for an orientable surface is finite. Archdeacon and Huneke [AH89] proved that the set of topological obstructions for a nonorientable surface is finite. Robertson and Seymour proved the same for orientable and nonorientable surfaces together [RS90].

To *subdivide* an edge $e = uv$ of a graph, remove e, introduce a new vertex x, and add two new edges ux and vx. A *subdivision* of a graph G is simply either equal to G, or obtained from G by successively subdividing some of

the edges of G. Two graphs H and G are *homeomorphic* if a subdivision of H is isomorphic to a subdivision of G.

Bodendiek and Wagner [BW89a] define a useful set of relations to characterize obstruction sets by smaller subsets of graphs. First, we need to introduce the graph operations listed in Table 5.1.

Type of Operation	Operation
R_0	an edge or isolated vertex of G is deleted.
R_1	an edge e of G is contracted, where e is incident with at least one vertex of degree two.
R_2	an edge e of G is contracted, where both endpoints of e have degree at least three, and if multiple edges appear between some pair of vertices after contracting e, all but one of these edges are removed.
R_3	a vertex of degree three v is removed and its neighbours are joined with edges (informally, replacing Y with a Δ, hence R_3 is sometimes called $Y\Delta$), and if multiple edges appear between some pair of vertices, all but one of these edges are removed.
R_4	let u, v be two adjacent vertices both of degree 3, with $N(u) \cap N(v) = \emptyset$. Subdivide the edge uv and then perform R_3 on u and v separately.

Table 5.1: Operations which reduce the order of a graph.

It is trivial to see when any of these R_i operations are applied to an embedding $\widetilde{G}$ of genus g that the genus of the resulting embedding cannot exceed g, because each operation involves either removing or replacing one planar subgraph with another.

Bodendiek and Wagner then define relations $\geq_i$ for $i = 1, \ldots, 4$ (note that $i \neq 0$), where we have for two graphs $G \geq_i H$ if and only if either $G = H$ or there is some finite sequence of graphs starting with G that are successively related by any of R_j, $j \in \{0, \ldots, i\}$, that terminates with H. If we look at the partial order associated with each relation on the set of graphs $\mathcal{F}$ that do not embed on a surface $\mathbb{S}$, then $M_i(\mathbb{S})$ is the set of $\geq_i$-minimal graphs of $\mathcal{F}$, and we sometimes refer to these sets as the *i^{th}-order minimal basis* for the surface $\mathbb{S}$. Note that each minimal basis M_i for $i \geq 1$ does not include any

graphs with vertices of degree zero, one, or two, due to the relations R_0 and R_1. Also, directly from these definitions, for any surface $\mathbb{S}$, we conveniently have $M_4(\mathbb{S}) \subseteq M_3(\mathbb{S}) \subseteq M_2(\mathbb{S}) \subseteq M_1(\mathbb{S})$. The sets $M_1(\mathbb{S})$, $M_2(\mathbb{S})$, $M_3(\mathbb{S})$ and $M_4(\mathbb{S})$ are also referred to as the *topological*, *minor*, *wye-delta* and *double-wye-delta* obstructions of $\mathbb{S}$, respectively.

Graph theory in computer science typically includes the topic of *planar* graphs, those that can be embedded in $\mathbb{S}_0$, for its implications in designing graph algorithms. A combinatorial embedding of a planar graph G in $\mathbb{S}_0$ is also called a *plane* graph. Kuratowski's Theorem [Kur30], asserts that a graph G is planar if and only if G contains no subgraph homeomorphic to a graph in $M_1(\mathbb{S}_0) = \{K_5, K_{3,3}\}$. Any graph homeomorphic to K_5 or $K_{3,3}$ is dubbed a *Kuratowski* graph. Similarly, a *projective-planar* graph is a graph that can be embedded in $\mathbb{N}_1$, a *toroidal* graph can be embedded in $\mathbb{S}_1$, and a *Klein* graph can be embedded in $\mathbb{N}_2$. Also, Wagner's [Wag37] Theorem characterizes the minor-order obstructions of the plane as $M_2(\mathbb{S}_0) = \{K_5, K_{3,3}\}$; that is, a graph G is planar if and only if K_5 and $K_{3,3}$ are excluded minors of G. Note that $M_3(\mathbb{S}_0) = M_4(\mathbb{S}_0) = \{K_5\}$.

A *vertex split* of a vertex v in a graph G partitions the neighbours of v into two sets V_1 and V_2 and replaces v with two new vertices v_1 and v_2, joined by an edge v_1v_2, where the neighbours of v_1 are $V_1 \cup \{v_2\}$ and the neighbours of v_2 are $V_2 \cup \{v_1\}$.

Let v be a vertex of an embedding $\widetilde{G}$ and the rotation system of v be partitioned into two rotation systems V_1 and V_2 such that π_v is the concatenation of V_1 and V_2. Then the *embedded vertex split* of $\widetilde{G}$ that corresponds to this partition (V_1, V_2) is the embedding $\widetilde{H}$ which has $V(H) = \{v_1, v_2\} \cup V(G) \setminus \{v\}$,

$$\begin{aligned} E(H) = E(G) \setminus \{uv \mid u \text{ is a neighbour of } v\} \\ \cup \{xv_1 \mid x \in V_1\} \cup \{xv_2 \mid x \in V_2\} \\ \cup \{v_1v_2\}, \end{aligned}$$

and the signatures of any edge incident on some vertex u and either v_1 or v_2 is assigned the same signature as edge uv in $\widetilde{G}$, with edge v_1v_2 assigned positive signature, and the vertices v_1 and v_2 have rotation systems V_1v_2 and V_2v_1, respectively.

5.2 Literature Review for Obstructions

Glover, Huneke, and Wang [GHW79] constructed a list of the 103 topological obstructions $M_1(\mathbb{N}_1)$ of the projective plane. Archdeacon [Arc81] showed that this list is complete. Bodendiek, Schumacher, and Wagner were the first to show the set of double-delta-wye-order obstructions $M_4(\mathbb{N}_1)$ has cardinality twelve. Flötotto [Flö10] lists the 103 obstructions and to which minimal basis each belongs. Flötotto also gives a hierarchy chart of how the 103 obstructions relate, but the chart is missing the graph D_9 [Flö10]. See Table 5.2 below for counts of all the obstructions of the projective plane.

n	M_1	M_2	M_3	M_4	total
7	0	0	0	2	2
8	3	1	1	5	10
9	9	5	3	3	20
10	23	6	3	2	34
11	20	1	2	0	23
12	11	0	1	0	12
13	2	0	0	0	2
total	68	13	10	12	103

Table 5.2: Number of obstructions on the projective plane, $\mathbb{N}_1$. The column labelled M_i gives the number of obstructions in M_i, but not in M_{i+1}, for $i = 1, \ldots, 3$.

Gagarin, Myrvold, and Chambers [GMC09] established that there are four minor-order and eleven topological obstructions for the torus when restricted to the class of those that contain no $K_{3,3}$ subdivision as a subgraph. Skoda [Sko12] proved that there are 68 and 668 topological obstructions of connectivity two for the torus and the Klein bottle, respectively. Flötotto [Flö10] constructed 83 obstructions of the Klein bottle, those in $M_4(\mathbb{N}_2)$, by identifying one, two, or three vertices between an obstruction in $M_4(\mathbb{N}_1)$ and a copy of K_5. The 83 constructed graphs are not a complete set of obstructions of the Klein bottle.

By combining the work of Fiedler, Huneke, Richter, Robertson [FHRR95], and Randby [Ran97], a torus obstruction that is projective-planar has been shown to be $\Delta Y\Delta$-equivalent to the 4×4 projective-planar grid. Juvan [Juv95] has found the complete set of 270 projective-planar torus obstructions.

Chambers [Cha02], Neufeld [Neu94], and Woodcock [Woo06], each supervised by Myrvold for their master's theses, performed and rechecked an exhaustive computer search of the 3-regular topological obstructions of the torus having at most 24 vertices. See Table 5.3 for the number of 3-regular topological obstructions of the torus of each order.

n	number of obstructions
12	1
14	9
16	20
18	133
20	39
22	2
24	2
total	206

Table 5.3: The number of 3-regular topological obstructions of the torus.

Work of Battle, Harary, and Kodama [BHK62] have shown the orientable genus of a graph is the sum of the orientable genera of its blocks. Additivity of orientable genus is also across components of a graph.

5.3 Results

The Successive Surface Scaffolding Conjecture implies that it should be possible for each wye-delta obstruction of a surface $\mathbb{S}$ for at least one embedding on $\mathbb{S}$ with a crosscap added to insert edges without increasing the Euler genus so that an irreducible triangulation is obtained.

Conjecture 5.1. *(Campbell) Each obstruction in $M_3(\mathbb{S})$ for a surface $\mathbb{S}$ of Euler genus g has some embedding in $\mathbb{N}_{g+1}$ such that it is possible to insert edges and vertices to obtain an irreducible triangulation of $\mathbb{N}_{g+1}$.*

Theorem 5.2 shows that any embedding of an obstruction must have each edge on an essential cycle. Perhaps it may be useful towards proving Conjecture 5.1.

Theorem 5.2. *Suppose G is an obstruction of some surface. If $\widetilde{G}$ is an embedding of G on some surface, then every edge of G is part of an essential cycle of $\widetilde{G}$.*

Proof. Suppose there is an obstruction G with an embedding $\widetilde{G}$. Consider the edge e in G and suppose that e is not in any essential cycle of $\widetilde{G}$. Let F be a facial walk of $\widetilde{G}$ containing e. The proof is broken down into two cases: either F contains some repeated vertex v, or all vertices of F are distinct.

Case 1. *Suppose F contains a repeated vertex v.*

Let F_1 be a smallest subwalk of F which starts and ends with v that contains e such that F_1 is a cycle. Note that F_1 cannot be a walk of a face different from F, because F_1 contains at least 3 vertices (there are no multiple edges) and the degree of each vertex of F_1 is at least 3. Further, if F_1 were not essential, then v would be a cut vertex of a planar block with outer face boundary F_1, which contradicts G as an obstruction. Thus, e sits on an essential cycle F_1.

Otherwise, no subwalk of F containing v and e exists such that F_1 has all unique vertices. Then let F_1 simply be a smallest subwalk of F starting and ending with v not containing e. As before, F_1 must be an essential cycle. Consider the subwalk $F_2 = F - F_1$ that starts and ends with v. Since F_2 contains e and e cannot be part of a subwalk cycle with a repeated vertex of F, it must be that F_2 contains some other repeated vertex u. Let F_3 be a smallest subwalk of F starting and ending with u. Again, F_3 must be essential, but by supposition cannot contain e. Partition all such subwalk cycles this way so that, without loss of generality, u and v are minimum distance away from e in either direction along F—regardless of whether F_1 or F_3 are walked along F at a distance greater than the minimum away from e; i.e., u and v may appear more than twice along F.

If $u = v$, the minimum subwalk cycle of F starting and ending at u and including e must be essential, else there is a planar block as before. Therefore, it is assumed $u \neq v$.

Consider the subwalk of F from u to v that includes e and call it P. It must be that P is a path by property of minimum distance u and v from e. Let P_1 be the subpath of P that is of minimum length connecting F_1 and F_3. Let the ends of P_1 be called s in F_1 and t in F_3. Again, for the same reason $u \neq v$, it must be $s \neq t$.

Subcase 1.1. *Suppose F_1 and F_3 have a vertex in common.*

Without loss of generality, let $w \neq v$ be a common vertex of minimum distance from t in F_3. Then a path P_2 from w to s along F_1 cannot contain any vertices of a minimum path P_3 from w to t along F_3. Note that P_3 may be trivial, but not P_2. If the cycle $P_1 \cup P_2 \cup P_3$ is essential, there is no need to continue. If not, then the path $P_1 \cup P_3$ can be continuously deformed to P_2, so that the cycle $(F_1 - P_2) \cup P_1 \cup P_3$ that contains e must be essential.

Subcase 1.2. *Suppose F_1 and F_3 do not share a vertex in common. Suppose there exists a path P_4 internally disjoint from P_1 that connects a vertex, without loss of generality, $w \neq s$ of F_1 to a vertex of F_3.*

Then there exists such a path P_4 of minimum length. Then a similar argument to Subcase 1.1 with adjustments to include the path P_4 concludes that e is in some essential cycle.

To continue, make use of a partition of P_1 into paths P_s, the path from s to e, the path e itself, and P_t the path from e to t. Note that either P_s or P_t can be trivial.

Subcase 1.3. *Suppose F_1 and F_3 do not share a vertex in common, and no such path P_4 exists. Suppose there exists a path P_5 internally disjoint from $P_s \cup e$ that connects a vertex, without loss of generality, $w \neq s$ of F_1 to a vertex $x \neq t$ of P_t (if P_t is trivial, then this reverts to Subcase 1.2).*

Let P_5 be a shortest such path. Then a similar argument to Subcase 1.1 with adjustments to include the path P_5 concludes that e is in some essential cycle.

Subcase 1.4. *Suppose F_1 and F_3 do not share a vertex in common, and no such paths P_4 nor P_5 exist. Then e must be part of a block B separated from G by two cut vertices, say y on P_s and z on P_t.*

Let B be such that y and z are minimum distance from e. There is a subwalk F_4 of F from y to z that includes e. Because F is a facial walk and y and z are cut vertices, another subwalk F_5 of F exists from z to y. Since y and z are minimum distance from e, it must be that $F_4 \cup F_5$ is either a cycle including e or e is an edge-cut of G. If $F_4 \cup F_5$ is a cycle, it must be essential, because an obstruction cannot contain a planar block. Neither can an obstruction contain an edge-cut of size one.

Case 2. *Suppose F contains no repeated vertex.*

Say some other edge $q \neq e$ of F was part of an essential cycle, and that C is such a cycle containing q where $C \cap F$ has a minimum number of components. Let $P_1, P_2, \ldots, P_{2j}$ be a partition of F into paths where $C \cap F = \{P_{2i} \mid 1 \leq i \leq j\}$, and let P_2 contain q. Note that $e \notin C \cap F$. Also, let $C \setminus F = \{Q_{2i-1} \mid 1 \leq i \leq j\}$ be the remaining set of paths of C not on F. Name the paths P_{2i-1} to correspond with the paths Q_{2i-1} so that their ends match vertices.

Now if $j = 1$, it must be that $P_1 \cup Q_1$ is an essential cycle containing e. This is because the path P_2 can be continuously deformed to the path P_1.

If $j > 1$, then see that any Q_{2i-1} cannot be continuously deformed to the path P_{2i-1}, because C is minimum with respect to the number j of components in $C \cap F$. Let r be such that P_{2r-1} does not contain e. But then if $L = F \setminus P_{2r-1}$ is used to form a cycle $D = L \cup Q_{2r-1}$, it must be that D is essential and contains e. This is because P_{2r-1} can be continuously deformed to L; i.e., the face walk F is not essential. Because D contains e, ultimately, no edge of F can be part of an essential cycle. Extending the argument above to the other faces of $\widetilde{G}$ sharing edges with F, these faces cannot contain edges that are part of essential cycles. Repeating until all faces of $\widetilde{G}$ are dealt with as above leads to a contradiction if one of the faces contains a repeated vertex as in Case 1, or ultimately shows that no edge of G can be part of an essential cycle. This final situation erroneously concludes $\widetilde{G}$ is a plane embedding.

□

Lemma 5.3. *If an obstruction G of a surface $\mathbb{S}$ contains a triangle, then there exists a nonorientable embedding $\widetilde{G}$ of Euler genus $\gamma(\mathbb{S}) + 1$.*

Proof. Suppose the triangle of G is $u_1u_2u_3$. Then there exists an embedding $\widetilde{D_1}$ of $D_1 = G_1 - u_1u_2$ on the surface $\mathbb{S}$. Let $X = (x_1, x_2, \ldots, x_j)$ be one of the two sequences of vertices between u_1 and u_2, non-inclusive, of the adjacency list of u_3 in $\widetilde{D_1}$, and let $Y = (y_1, y_2, \ldots, y_k)$ be the other sequence. Without loss of generality, one of X or Y is nonempty, because u_3 is at least degree 3, so let $j \geq 1$. Let f_1 be the face of $\widetilde{D_1}$ with boundary containing u_3, x_1, and u_1, and let f_2 be the face of $\widetilde{D_1}$ with boundary containing u_3, x_j and u_2. Note that $f_1 \neq f_2$, since otherwise the edge u_1u_2 could be inserted and G would not be an obstruction.

Construct embedding $\widetilde{D_2}$ from $\widetilde{D_1}$ by changing the signature of each edge x_iu_3, for $x_i \in X$, and replacing X with X^R in the adjacency list of u_3. Let the counterclockwise facial walk of f_1 be $u_1u_3x_1W$ for some sequence of

vertices W, and let the counterclockwise facial walk of f_2 be $u_3u_2Zx_j$ for some sequence of vertices Z. Then the construction of $\widetilde{D}_2$ merges f_1 and f_2 into a new face f with facial walk $u_1u_3x_jZ^Ru_2u_3x_1W$. All the other faces in the construction retain the same facial walks. Therefore, there is one less face in $\widetilde{D_2}$ than in $\widetilde{D_1}$. Hence, the Euler genus of $\widetilde{D_2}$ is $\gamma((S))+1$. Finally, let $\widetilde{D_3}$ be the embedding obtained from $\widetilde{D_2}$ by inserting the edge u_1u_2 with a -1 signature. Since the ends of u_1u_2 are inserted into the face f where the angles must have opposite parity, by Theorem 3.1, the Euler genus of $\widetilde{D_3}$ must be the same as that of $\widetilde{D_2}$. □

Corollary 5.4. *The torus obstructions containing no $K_{3,3}$ subdivision each have an embedding in $\mathbb{N}_3$.*

Proof. The torus obstructions with no $K_{3,3}$ subdivision are listed in a paper by Gagarin, Myrvold and Chambers [GMC09], and these obstructions each have a triangle. By Lemma 5.3, each of these obstructions has an embedding in $\mathbb{N}_3$. □

Of the 249348 known topological obstructions of the torus, there are 136955 that each have a triangle, and therefore these each have an embedding on $\mathbb{N}_3$. Algorithm 4.4 was used to obtain embeddings of the known wye-delta obstructions of the torus. Each of them has at least one embedding in $\mathbb{N}_3$, so no counterexample is known yet for Conjecture 1.3 in the case of the torus.

Theorem 5.5. *For an arbitrary surface $\mathbb{S}$, obstructions in $M_2(\mathbb{S}) \setminus M_4(\mathbb{S})$ have a nonorientable embedding of Euler genus $\gamma(\mathbb{S})+1$.*

Proof. Suppose there is an obstruction $G \in M_2(\mathbb{S}) \setminus M_4(\mathbb{S})$. Then there exists a graph $H \in M_4(\mathbb{S})$ such that $G >_4 H$, and there are some sequence of the graph operations R_j with $j \in \{0,1,2,3,4\}$ that relates G to H. The first such operation cannot have $j < 3$, because $G \in M_2$.

Case 1. *The first operation on G is R_3.*

Therefore, there exists at least one vertex u of degree 3 in G. Let G_1 be the graph resulting from an R_3 operation to replace u with a triangle $u_1u_2u_3$. The R_3 operation cannot be followed by an R_0 operation of deleting some edge of the triangle in G_1, because this is the same as originally contracting an edge incident with u in G and contradicts the first operation with $j \geq 3$. Similarly, an R_2 operation of contracting an edge of $u_1u_2u_3$ in G_1 is the same as originally contracting two edges incident with u in G. Contracting or

deleting some edge other than in the triangle $u_1u_2u_3$ is the same as performing such an operation on G before the R_3 operation, which would result in an embedding of G_1 in the surface $\mathbb{S}$, and hence such an embedding for H as well. Therefore, G_1 is at least an obstruction in M_2.

By Lemma 5.3, there is a nonorientable embedding $\widetilde{D_3}$ of G_1 of Euler genus $\gamma(\mathbb{S}) + 1$.

The following leads to an Euler genus $\gamma(\mathbb{S})+1$ embedding of G. Obtain $\widetilde{D_4}$ by subdividing the edge u_1u_2 of $\widetilde{D_3}$ to include a new vertex u and give the edge uu_1 signature -1 and uu_2 signature $+1$. From the construction for Lemma 5.3, the signature of the edge (u_1, u_2) is -1, and therefore, $\gamma(\widetilde{D_3}) = \gamma(\widetilde{D_4})$. Next, obtain $\widetilde{D_5}$ by inserting the edge uu_3 with signature $+1$ such that it forms a plane triangle with u_2 and u_3. Obviously, $\gamma(\widetilde{D_4}) = \gamma(\widetilde{D_5})$. Finally, delete the edges u_1u_3 and u_2u_3 to obtain an embedding $\widetilde{D_6}$ of G. Note that in constructing $\widetilde{D_6}$ from $\widetilde{D_5}$ that two faces are lost as well as two edges, so $\gamma(\widetilde{D_5}) = \gamma(\widetilde{D_6})$.

Case 2. *The first operation on G is R_4.*

Then there are two adjacent vertices u and v both of degree 3. Let the neighbours of u be u_1,u_2, and v, and the neighbours of v be v_1, v_2, and u. Let G_1 be the graph resulting from replacing u and v with triangles $u_1u_2u_3$ and $v_1v_2u_3$, respectively, and note that u_3 belongs to both triangles and has degree 4. It cannot be that this R_4 operation is followed by deleting one of the edges. Suppose to the contrary that a graph G_2 is obtained by deleting, without loss of generality, u_1u_3 of G_1. Consider the graph G_3 obtained from G by contracting the edge uu_2. Note that G_3 is isomorphic to $G_2 - v_1v_2$, and since an embedding of G_3 in $\mathbb{S}$ where v is of degree 3, it is always possible to insert v_1v_2 into this embedding, so that G_2 embeds in $\mathbb{S}$. Now suppose instead that G_2 is obtained by deleting an edge, without loss of generality, u_1u_2 of G_1. Yet G_2 is isomorphic to the graph obtained from G by an R_3 operation on the vertex v. However, it is already dealt with in Case 1. Ultimately, G_1 must be an obstruction in M_1.

Then perform the exact same construction as in Case 1 on the triangle $u_1u_2u_3$ of G_1 to obtain an embedding $\widetilde{D_6}$ of Euler genus $\gamma(\mathbb{S}) + 1$ where the edges of the triangle $u_1u_2u_3$ are removed and replaced by a vertex u adjacent to u_1, u_2, and u_3. Also note that in this construction, either Y must be empty, or both X and Y have one vertex each, without loss of generality, $X = \{v_1\}$ and $Y = \{v_2\}$.

If Y is empty, then $X = \{v_1v_2\}$. Obtain a new embedding $\widetilde{D_7}$ from $\widetilde{D_6}$ by subdividing the edge v_1v_2 with a new vertex v, insert edge vu_3 with -1 signature into the angles v_1vv_2 and $v_1u_3v_2$, and delete the edges v_1u_3 and v_2u_3. Next get the embedding $\widetilde{D_8}$ from $\widetilde{D_7}$ by contracting the edge uu_3. This results in an embedding $\widetilde{D_8}$ of G with Euler genus $\gamma(\mathbb{S}) + 1$.

If Y is not empty, obtain a new embedding $\widetilde{D_7}$ from $\widetilde{D_6}$ by removing the edge v_1u_3 and subdividing v_2u_3 with a new vertex v. Notice that $\widetilde{D_7}$ would be an embedding in $\mathbb{S}$ except for one edge of negative signature uu_1. Then see that v_1 and v are in the boundary of a shared face in $\widetilde{D_7}$ and are corners of angles with opposite parity so that an edge vv_1 can be inserted with signature -1 to obtain an embedding $\widetilde{D_8}$. The embedding $\widetilde{D_9}$ obtained from $\widetilde{D_8}$ by contracting the edge u_3v results in an embedding of G with Euler genus $\gamma(\mathbb{S}) + 1$. □

Chapter 6

Depictions of Embeddings

Working with orientable and nonorientable embeddings of various Euler genus graphs is complex and this work is eased by having a way to visualize embeddings. Therefore, this chapter defines in Section 6.1 a depiction as a plane embedding that uses cycles to represent crosscaps, handles, and twisted handles of some higher Euler genus embedding. It is obvious that there can be more than one depiction for a given nonorientable embedding of Euler genus greater than zero, but some interesting examples are considered in Section 6.2. Lastly, in Section 6.3, a $\mathcal{O}(n^2)$ time algorithm is given for finding a depiction.

6.1 Definitions

First, consider the following example construction involving an embedding that has a crosscap. Suppose that we have an embedding $\widetilde{G}$ on the projective plane, with edges Q that use the crosscap in cyclic order

$$(u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k).$$

First subdivide each edge of Q twice so that edge (u_i, v_i) becomes the path $u_i a_i b_i v_i$. Next remove the edges (a_i, b_i) for $i = 1, \ldots, k$. Then add edges to create a cycle $a_1 a_2 \cdots a_k b_1 b_2 \cdots b_k$. When adding the cycle, the resulting rotation systems should be:

a_1: u_1, a_2, b_k,

a_i: u_i, a_{i+1}, a_{i-1}, for $i = 2, 3, \ldots, k-1$,

a_k: u_k, b_1, a_{k-1},

b_1: v_1, b_2, a_k,

b_i: v_i, b_{i+1}, b_{i-1}, for $i = 2, 3, \ldots, k-1$,

b_k: v_k, a_1, b_{k-1}.

An example is given in Figures 6.1 and 6.2 where five edges crossing in cyclic order are replaced as described. Note that during the construction, before inserting the cycle edges, the face involving the (u_i, a_i) and (v_i, b_i) edges could be walked to obtain the angles where the cycle edges should be inserted.

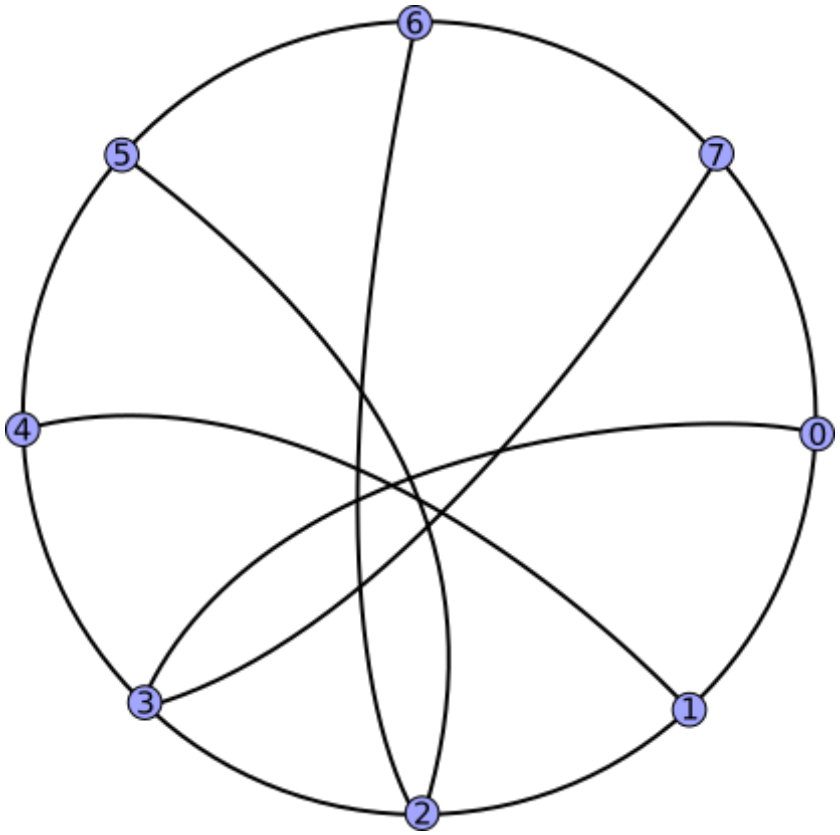


Figure 6.1: An example of an embedding drawn with 5 edges crossing in a cyclic order.

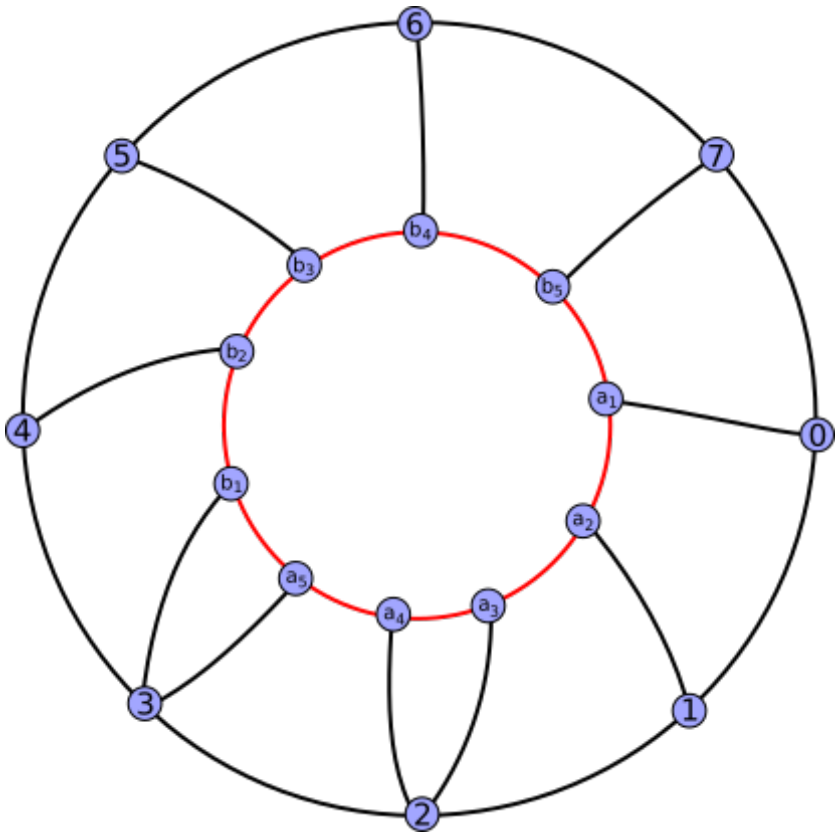


Figure 6.2: An example of an embedding where 5 crossing edges in the cyclic order presented in Figure 6.1 are subdivided and modified as discussed. A cycle C representing a crosscap is indicated with red edges, and note that the resulting embedding is planar.

This example involving a projective-plane obstruction motivates a construction for embeddings in general. Define a *feature* to be any one of a crosscap, twisted handle, or handle. Suppose that $\widetilde{G}$ is an embedding with t features $F_1, F_2, \ldots, F_t$. If for feature F_i, the cyclic order of edges going over the feature from one side to the other is $(u_1, v_1), \ldots, (u_k, v_k)$ then the feature is represented by adding two new sets of vertices A_i and B_i where $|A_i| = |B_i| = k$. Edge (u_j, v_j) is subdivided twice to be the path $u_j a_j b_j v_j$ and the edge (a_j, b_j) is removed. Then edges are added to the embedding as follows:

- for a crosscap, insert the edges in the cycle $a_1 a_2 \cdots a_k b_1 b_2 \cdots b_k$;

- for a twisted handle, insert the edges in the cycle $a_1a_2\cdots a_k$ and the cycle $b_1b_2\cdots b_k$;
- for a handle, insert the edges in the cycle $a_1a_2\cdots a_k$ and the cycle $b_kb_{k-1}\cdots b_2b_1$.

The rotation systems for the cycles are chosen so that the interiors of these new cycles are empty. For the crosscap, the rotation systems were already given in the previous example. For a handle, the rotation systems should be:

a_1: u_1, a_2, a_k,

a_i: u_i, a_{i+1}, a_{i-1}, for $i = 2, 3, \ldots, k-1$,

a_k: u_k, a_1, a_{k-1},

b_1: v_1, b_k, b_2,

b_i: v_i, b_{i-1}, b_{i+1}, for $i = 2, 3, \ldots, k-1$,

b_k: v_k, b_{k-1}, b_1.

For a twisted handle, the rotation systems should be:

a_1: u_1, a_2, a_k,

a_i: u_i, a_{i+1}, a_{i-1}, for $i = 2, 3, \ldots, k-1$,

a_k: u_k, a_1, a_{k-1},

b_1: v_1, b_2, b_k,

b_i: v_i, b_{i+1}, b_{i-1}, for $i = 2, 3, \ldots, k-1$,

b_k: v_k, b_1, b_{k-1}.

To visualize the features in 3D, vertex a_j is identified with b_j and the new cycle edges are removed. For this reason, vertices a_j and b_j are called a *pair*.

Formally, for an embedding $\widetilde{G}$, a *depiction* consists of a planar embedding $\widetilde{H}$ with vertex set partitioned into $2t+1$ parts where t is the number of features:

$$V(H) = V(G) \cup A_1 \cup A_2 \cup \cdots \cup A_t \cup B_1 \cup B_2 \cup \cdots \cup B_t.$$

Each A_i and B_i together represents some feature. If k edges use this feature, then $|A_i| = |B_i| = k$. Vertices $a_j \in A_i$ and $b_j \in B_i$ are degree three. When the faces of the resulting planar embedding are traversed in a common direction, clockwise or counterclockwise, then for:

- a crosscap, the cycle $C : a_1 a_2 \dots a_k b_1 b_2 \dots b_k$ is a face, or C^R is a face;
- a handle, then cycles $C_1 : a_1 a_2 \dots a_k$ and $C_2 : b_{k-1} \dots b_2 b_1$ are faces, or C_1^R and C_2^R are faces;
- a twisted-handle, then cycles $C_1 : a_1 a_2 \dots a_{k-1}$ and $C_2 : b_1 b_2 \dots b_{k-1}$ are faces, or C_1^R and C_2^R are faces.

Define the vertices of A_i and B_i as *feature vertices*. Further define $\widetilde{H}$ to *depict* an embedding $\widetilde{G}$. The edges uv of $\widetilde{H}$ with $u, v \in V(G)$ correspond to edges uv in $\widetilde{G}$ with $+1$ signature. Delete all edges (u, v) where $u, v \in A_i \cup B_i$, identify all vertices a_i with b_i and label the new vertex as a_i, and define the *sign* of a_i to be $+1$ if a_i corresponds to a handle face, and -1 if a_i corresponds to a crosscap or twisted-handle face of $\widetilde{H}$. Then for *half edges* ua, where $u \in V(G)$ and a_{t_1} is a feature vertex, find the vertex v at the end of the path $P : ua_{t_1}a_{t_2} \dots a_{t_p}v$ with each a_{t_i} a feature vertex. Then the corresponding edge uv in $\widetilde{G}$ has signature equal to $\prod_{j=1}^{p} sign(a_{t_j})$. Only simple graphs G are considered, so that there cannot be more than one such path P. Intuitively, each a_j corresponds to the edge uv using a twist if a_j lies on a crosscap or twisted-handle cycle, and uv using a handle if a_j lies on a handle cycle. In Figure 6.3, an embedding of a torus obstruction with 15 vertices is depicted by a planar graph with blue handle faces, a red crosscap face, and feature vertices labelled 16 to 28.

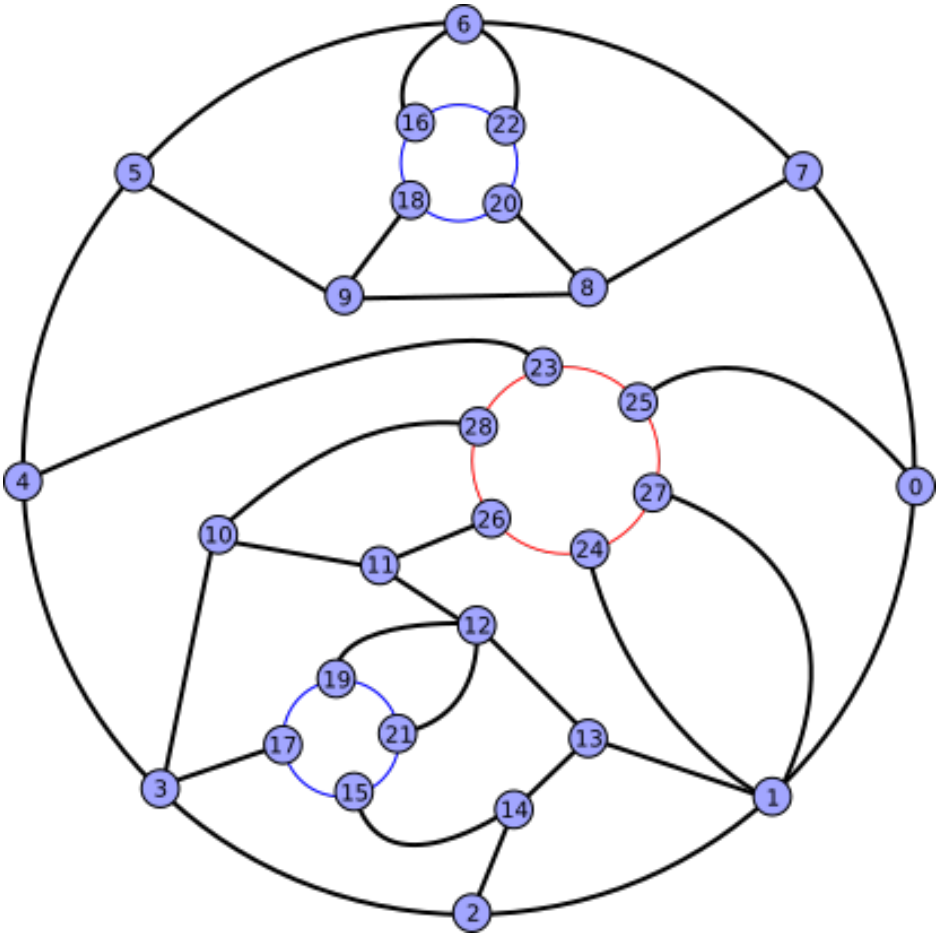


Figure 6.3: An example of a depiction of a torus obstruction with 15 vertices, labelled 0 to 14, embedded in $\mathbb{N}_3$ where the handle faces are indicated in blue, the crosscap face is indicated in red, and the feature vertices are labelled 15 to 28, with the first of each pair numbered odd $2k-1$ and the second in the pair numbered $2k$, for $k = 8, 9, \ldots, 16$.

6.2 Different Depictions of the Same Embedding

A signed combinatorial embedding $\widetilde{G}$ does not explicitly assign surface points to graph vertices and edges. However, a vertex in $V(G)$ of any depiction must

have the same rotation system as that in $\widetilde{G}$, where the appearance of some vertices in adjacency lists are replaced by appearances of feature vertices. However, depictions $\widetilde{H}$ of $\widetilde{G}$ may be drastically different depending on how the feature vertices are setup corresponding to edges in $\widetilde{G}$. Further, any subset S of the surface can be continuously deformed, so to categorize parts of $\widetilde{G}$ for a depiction by assigning sets A_i and B_i then leads to the following questions:

(1) Is the embedding nonplanar?

(2) If yes, which combination of features will be used to help depict the surface?

(3) Which edges in the graph will use each feature?

The Classification Theorem of Surfaces [Kos80] implies that for the Klein bottle two crosscaps are homeomorphic to a twisted handle, and for any surface with three or more crosscaps, any two of them are homeomorphic to a twisted handle, or exclusively, a handle while using the third crosscap. So which features should be used? Altogether, surfaces of nonorientable genus two or more will involve different depictions potentially using completely different feature vertices. On such surfaces, one may choose some combination of twisted handles, crosscaps, and handles to get a depiction, but a poor choice may force edges of the graph to use a handle, crosscap, or twisted handle more than once, as seen in Figure 6.7 with the dashed edges inside the orange and green coloured faces. However, for some embeddings, edges using a feature more than once may be necessary no matter what features are used to create a depiction. Perhaps there are algorithms for finding a depiction with a minimal number of edges using features more than once.

When depicting an embedding on any nonorientable surface of genus greater than one, we must consider what combination of features will be used. Obviously, the projective plane must be depicted with a crosscap, and any orientable surface depicted with handles. It should be possible to find polynomial-time algorithms to convert between the different depictions without any switching of vertices for these situations and this remains an open problem.

Figures 6.4, 6.5 show examples of different depictions of an embedding first represented with one handle and a crosscap, and second, with three crosscaps. Figures 6.6, 6.7 give the reverse situation, showing examples of

different depictions of an embedding first represented with three crosscaps, and second, with one handle and a crosscap.

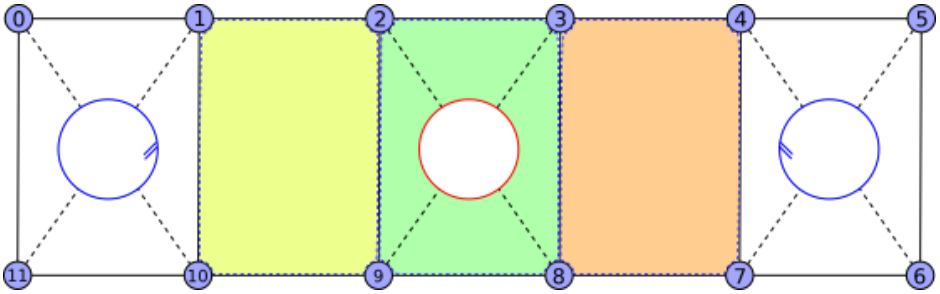


Figure 6.4: A depiction of an embedding $\widetilde{G_1}$ with 12 vertices embedded in the surface $\mathbb{N}_3$ with one crosscap (outlined red circle) and one handle (two outlined blue circles identified as indicated). Three separate faces are coloured yellow, green, and orange to help indicate the same faces in Figure 6.5. Note that feature vertices are omitted.

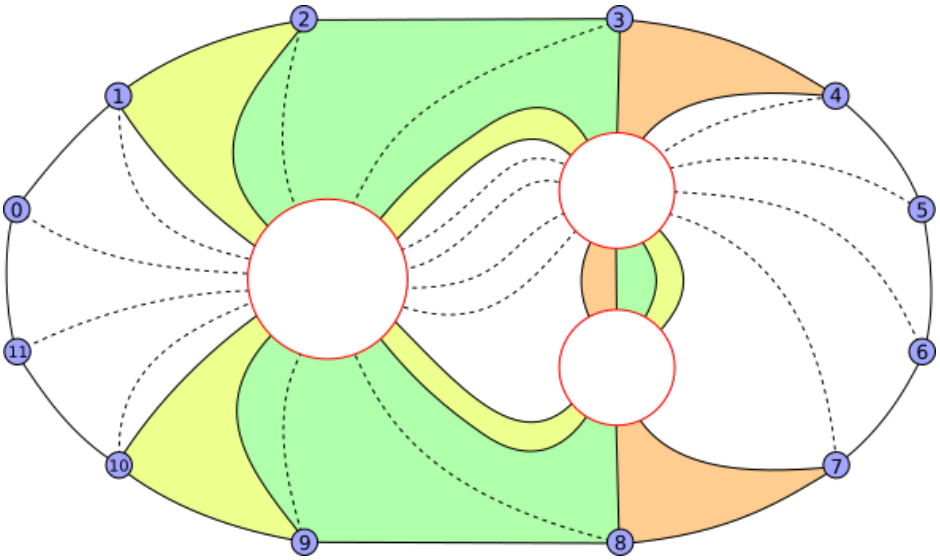


Figure 6.5: Another depiction of the embedding $\widetilde{G_1}$ from Figure 6.4 with 12 vertices embedded in the surface $\mathbb{N}_3$ with three crosscaps (outlined red circles), and with three separate faces coloured yellow, green, and orange to help indicate the same faces in Figure 6.4.

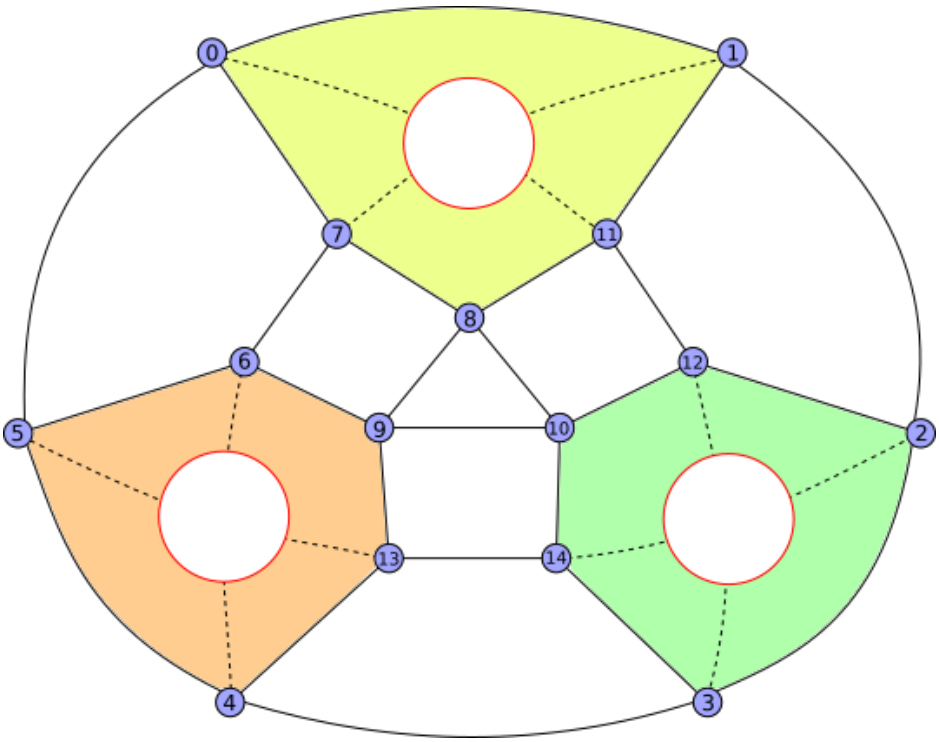


Figure 6.6: A depiction of an embedding $\widetilde{G_2}$ with 15 vertices embedded in the surface $\mathbb{N}_3$ with three crosscaps (outlined red circles), and with three separate faces coloured yellow, green, and orange to help indicate the same faces in Figure 6.7.

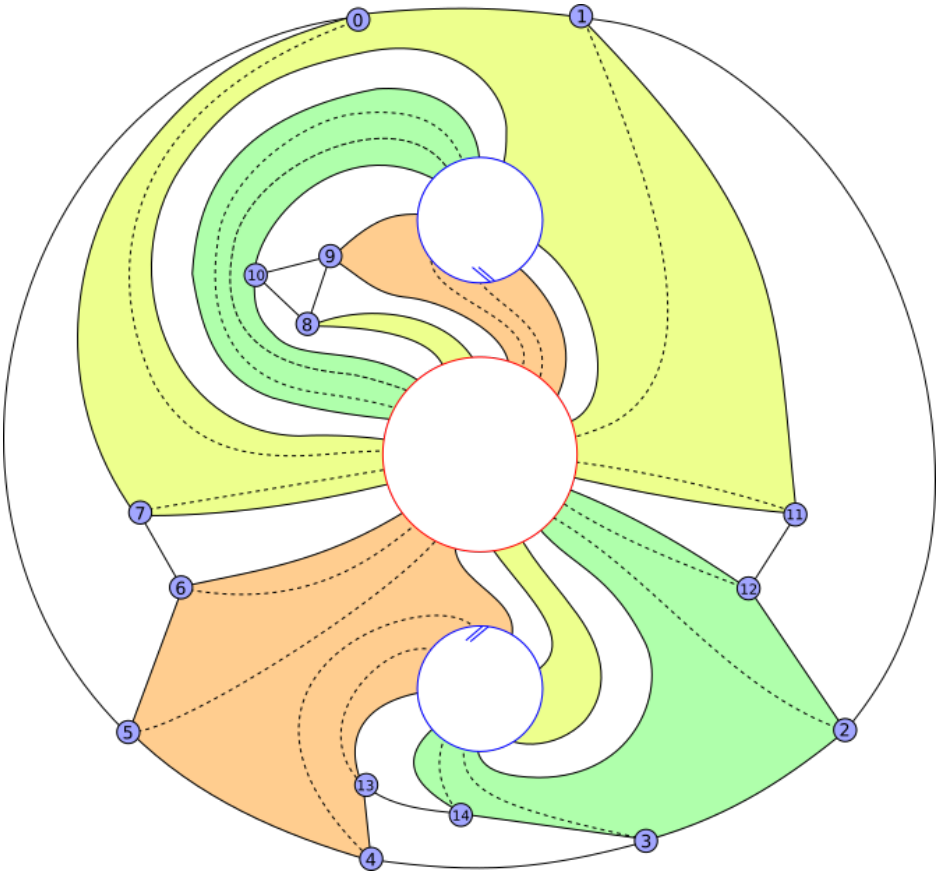


Figure 6.7: A depiction of the embedding $\widetilde{G_2}$ from Figure 6.6 with 15 vertices embedded in the surface $\mathbb{N}_3$ with one crosscap (outlined red circle) and one handle (two outlined blue circles identified as indicated). Three separate faces are coloured yellow, green, and orange to help indicate the same faces in Figure 6.6. Note that the dashed edges in the green and orange faces use both the crosscap and handle.

6.3 Algorithm to Obtain a Depiction

Visualizing graphs is itself a huge area of research. A practical approach is sought in order to develop an algorithm that will construct a plane embedding depiction using the augmented rotation systems data structure. Then any visualization techniques for plane embeddings others may wish to apply can do so as needed. It should be emphasized here that the resulting plane

embedding may include loops and multiple edges, but this should not pose any problems since twin pointers are used.

Suppose a depiction of some combinatorial embedding $\widetilde{G}$ is desired. An obvious first step for constructing a planar embedding $\widetilde{D}$ from $\widetilde{G}$ is to set $V(D) = V(G)$ and insert $+1$ signature edges of a spanning tree T of $\widetilde{G}$ while keeping the rotation systems of each vertex in the same order. Switches on the vertices of $\widetilde{G}$ can be performed to obtain such a tree if necessary. Use a subembedding $\widetilde{H}$ of $\widetilde{G}$ identical to $\widetilde{D}$ at this point. Use $\widetilde{H}$ to guide the construction of $\widetilde{D}$ by inserting into $\widetilde{H}$ each remaining edge of $\widetilde{G}$ not in the tree T. Suppose there is such an edge $e = uv$ at an arbitrary stage of iteration. By Theorem 3.1 there are three possible changes in genus to $\widetilde{H}$ after inserting e, $\Delta = 0, 1$, or 2. Together with the signature s of e, this dictates:

- for $\Delta = 2$ and $s = 1$, add a handle feature to $\widetilde{D}$ (Algorithm 6.8),
- for $\Delta = 2$ and $s = -1$, add a twisted-handle feature to $\widetilde{D}$ (Algorithm 6.8),
- for $\Delta = 1$ add a crosscap feature to $\widetilde{D}$ (Algorithms 6.9),
- for $\Delta = 0$ no features are added (this case is handled by Algorithms 6.10, 6.11, and 6.4).

A face walk in $\widetilde{D}$ may involve feature vertices, so that Algorithm 3.2 cannot be applied, because when a feature vertex is visited during the walk, then the walk must jump to the other vertex of its pair in the feature, and continue the walk from there. The walk must also use each twist feature vertex pair to switch between clockwise and counterclockwise direction when jumping from the first visited feature vertex visited in the pair to the second.

The following methods are needed in the `AugRotSystem` object:

- `AugRotSystem( AugRotSystem` G `)`: a constructor that sets up rotation systems from a spanning tree T of the input embedding G with *face_num* of each `AdjNode` set to 0 and *angle_parity* set to $+1$,
- `FindPlace` (Algorithm 6.1),
- `Subdivide` (Algorithm 6.2),
- `InsertLoop` (Algorithm 6.3),

- `WalkHalfFace` (Algorithm 6.4).

Other supporting data structures are also needed:

- `PathNode` with members:
  - *angle_ptr*: an `AdjNode` pointer corresponding to an angle,
  - *next*: a `PathNode` pointer to the next node in a list,
  - *prev*: a `PathNode` pointer to the previous node in a list,

  and a constructor method that takes an `AdjNode` pointer as input to assign to *angle_ptr* and appends the new `PathNode` at the end of a list,
- `PathList` with members:
  - *path_start*: a `PathNode` pointer to the start of this list,
  - *path_rear*: a `PathNode` pointer to the last node of this list,
  - *size*: the number of nodes in this list,

  and methods:
  - `append( AdjNode` *ptr* `)`: inserts *ptr* at the end of this `PathList`, updates *path_rear*, and increments *size* by one,
  - `del_front()`: removes the first `PathNode` in this `PathList` and decrements *size* by one,
  - `del_rear()`: removes the last `PathNode` in this `PathList` and decrements *size* by one.

The new data structure `Depiction` has members:

- G: an `AugRotSystem` representing the embedding to be depicted,
- H: an `AugRotSystem` representing a subembedding of G,
- D: an `AugRotSystem` that is the depiction of the embedding G,
- *original_n*: an integer indicating the original number of vertices in G,
- *feature_number*: an integer array keeping track of which feature vertices belong to which features,

- *feature_type*: an integer array keeping track of which feature vertices belong to which type of feature,
- `NMAX`: a static integer constant for the maximum number of vertices available,
- `UNDEFINED`: a static integer sentinel value -1 for a vertex with undefined feature number,
- `ORIGINAL`: a static integer constant 0 corresponding to non-feature vertices,
- `HANDLE`: a static integer constant 1 corresponding to feature vertices of a handle,
- `CROSSCAP`: a static integer constant 2 corresponding to feature vertices of a crosscap,
- `TWISTED_HANDLE`: a static integer constant 3 corresponding to feature vertices of a twisted handle,

and methods:

- `Depiction` (Algorithm 6.5),
- `InsertDepictionChords` (Algorithm 6.6),
- `GetPair` (Algorithm 6.7),
- `AddHandle` (Algorithm 6.8),
- `AddCrosscap` (Algorithm 6.9),
- `SubdivideFeatureEdges` (Algorithm 6.10),
- `AddEdgesBetweenFeatures` (Algorithm 6.11).

Algorithm 6.1. *Find Place*

INPUT:

- an integer vertex label u,

- an integer *pos* corresponding to a position number to compare with *my_position* values in the adjacency list of u.

ACTION:

- returns an AdjNode pointer to the desired position in the adjacency list of u.

```
FindPlace( u, pos ) {
1 ) AdjNode ptr;
2 ) // check if new node belongs at end of list
3 ) if ( V[u].prev.my_position < pos )
4 )   return V[u].prev;
5 ) // otherwise, it belongs somewhere in the middle
6 ) ptr = V[u];
7 ) while ( ptr.next.my_position < pos ) {
8 )   ptr = ptr.next;
9 ) }
10 ) return ptr;
}
```

Algorithm 6.2. *Subdivide*

INPUT:

- an AdjNode pointer *ptr* corresponding to some adjacency in a rotation system,
- only meant for use with a planar depiction, so the edge corresponding to *ptr* has a different face on either side.

ACTION:

- inserts a new vertex connected to the ends of the edge corresponding to *ptr*,
- updates the faces either side of the edge corresponding to *ptr*.

```
Subdivide( ptr ) {
1 ) AdjNode rear, ptr_v;
2 ) ptr_v = ptr.twin;
```

```
3 ) WalkOneFace( ptr );
4 ) WalkOneFace( ptr_v );
5 ) ptr.u = n;
6 ) ptr_v.u = n;
7 ) V[n] = new AdjNode(n, ptr.x, 1, NULL, NULL);
8 ) rear = new AdjNode(n, ptr_v.x, 1, V[n], V[n]);
9 ) V[n].twin = ptr;
10 ) ptr.twin = V[n];
11 ) rear.twin = ptr_v;
12 ) ptr_v.twin = rear;
13 ) degree[n] = 2;
14 ) m++;
15 ) n++;
}
```

Algorithm 6.3. *Insert Loop*

INPUT:

- a AdjNode pointer *ptr* corresponding to the angle where one end of an edge is to be inserted.

ACTION:

- a new vertex labelled n with a loop on n is added to the augmented rotation system corresponding to *ptr* and an edge inserted between $ptr.twin.u$ and n.

```
InsertLoop( ptr ) {
1 ) AdjNode rear, n1, n2;
2 ) int u;
3 ) // remove the old face
4 ) WalkOneFace( ptr );
5 ) f--;
6 ) // add the loop
7 ) V[n] = new AdjNode(n, n, 1, NULL, NULL);
8 ) rear = new AdjNode(n, n, 1, V[n], V[n]);
9 ) V[n].twin = rear;
10 ) rear.twin = V[n];
11 ) // add edge from new vertex to u
```

```
12 ) u = ptr.twin.u;
13 ) n1 = new AdjNode(n, u, 1, rear, V[n]);
14 ) n2 = new AdjNode(u, n, 1, ptr, ptr.next);
15 ) n1.twin = n2;
16 ) n2.twin = n1;
17 ) degree[n] = 3;
18 ) degree[u]++;
19 ) // update new faces
20 ) WalkOneFace( V[n] );
21 ) f++;
22 ) WalkOneFace( V[n].next );
23 ) f++;
24 ) n++;
25 ) m = m + 2;
26 ) g = 2 - n + m - f;
}
```

Algorithm 6.4. *Walk Half Face*

INPUT:

- a Depiction d,
- an AdjNode pointer ptr_u for the first angle of d to begin the walk,
- an AdjNode pointer ptr_v for the last angle of d to finish the walk.

ACTION:

- returns a PathList $path$, starting with ptr_u, followed by pointers to edges of features visited during the walk, and ending with ptr_v.

```
WalkHalfFace( d, ptr_u, ptr_v) {
1 ) boolean done;
2 ) AdjNode ptr;
3 ) int pair, dir;
4 ) PathList path = new PathList();
5 ) path.append(ptr_u);
6 ) ptr = ptr_u;
7 ) dir = 1;
```

```
 8 ) do {
 9 )   // walking on original vertex, nothing is saved
10 )   if ( ptr.twin.u < d.G.n) {
11 )     if ( dir == 1 )
12 )       ptr = ptr.next.twin;
13 )     else
14 )       ptr = ptr.prev.twin;
15 )   } else {
16 )     // add the edge pointer to subdivide later
17 )     if ( dir == 1 ) {
18 )       // at non-feature edge, want edge CW from this
19 )       path.append( ptr.next );
20 )     } else {
21 )       // at non-feature edge, want edge CCW from this
22 )       path.append( ptr.prev );
23 )     }
24 )     // find the paired vertex
25 )     pair = d.GetPair( ptr.twin.u );
26 )     ptr = V[pair];
27 )     if ( d.feature_type[ptr.twin.u] ≠ Depiction.HANDLE )
28 )       dir = −dir;
29 )     // move to adjacency that is not on this feature
30 )     while (
        d.feature_number[ptr.twin.u] == d.feature_number[ptr.u] )
31 )       ptr = ptr.next;
32 )     if ( dir == 1 )
33 )       path.append( ptr.prev );
34 )     else
35 )       path.append( ptr.next );
36 )     ptr = ptr.twin;
37 )   } // end of walking a feature
38 )   if ( path.size ≥ 5 ) {
39 )     int x1, x2;
40 )     x1 = path.path_rear.angle_ptr.twin.u;
41 )     x2 = path.path_rear.prev.prev.angle_ptr.twin.u;
42 )     if ( d.feature_number[x1] == d.feature_number[x2] ) {
43 )       for (int i = 0; i < 4; i++)
44 )         path.del_rear();
```

```
45 )      }
46 )    }
47 )    done = false;
48 )    if ( ptr == ptr_v and dir == 1 )
49 )      done = true;
50 )    if ( ptr.prev == ptr_v and dir == -1 )
51 )      done = true;
52 ) } while ( !done );
53 ) path.append( ptr_v );
54 ) return path;
}
```

Algorithm 6.5. *Depiction*

INPUT:

- an augmented rotation system *oldG* representing an embedding.

ACTION:

- creates a plane embedding depiction D of *oldG*.

```
Depiction( oldG ) {
 1 ) G = oldG;
 2 ) G.InitPositions();
 3 ) H = new AugRotSystem(G);
 4 ) D = new AugRotSystem(G);
 5 ) feature_type = new int[NMAX];
 6 ) feature_number = new int[NMAX];
 7 ) num_feature = 0;
 8 ) for (int i = 0; i < G.n; i++) {
 9 )    feature_type[i] = ORIGINAL;
10 )    feature_number[i] = UNDEFINED;
11 ) }
12 ) InsertDepictionChords();
}
```

Algorithm 6.6. *Insert Depiction Chords*

ACTION:

- inserts the edges of G not in T into D in the order given in the adjacency lists of G,
- any features needed during edge insertion are added to D.

```
InsertDepictionChords() {
 1 ) PathList p1, p2, p;
 2 ) AdjNode h_ptr1, h_ptr2;
 3 ) AdjNode d_ptr1, d_ptr2;
 4 ) int u, v, pu, pv;
 5 ) boolean need_crosscap;
 6 ) boolean need_handle;
 7 ) int new_sign;
 8 ) int new_g;
 9 ) for (u = 0; u < G.n; u++) {
10 )   AdjNode ptr = G.V[u];
11 )   for (int j = 0; j < G.degree[u]; j++) {
12 )     v = ptr.u;
13 )     if ( u < v and G.T.A[u][v] == 0 ) {
14 )       pu = ptr.my_position;
15 )       pv = ptr.twin.my_position;
16 )       new_sign = ptr.sign;
17 )       h_ptr1 = H.FindPlace( u, pu );
18 )       h_ptr2 = H.FindPlace( v, pv );
19 )       d_ptr1 = D.FindPlace( u, pu );
20 )       d_ptr2 = D.FindPlace( v, pv );
21 )       new_g = H.NewGenus( h_ptr1, h_ptr2, new_sign );
22 )       if ( new_g - H.g == 1 )
23 )         need_crosscap = true;
24 )       else
25 )         need_crosscap = false;
26 )       if ( new_g - H.g == 2 )
27 )         need_handle = true;
28 )       else
29 )         need_handle = false;
30 )       H.AddEdge( h_ptr1, h_ptr2, new_sign );
31 )       h_ptr1.next.my_position = pu;
32 )       h_ptr2.next.my_position = pv;
```

```
33 )        if ( need_handle )
34 )          AddHandle( d_ptr1, d_ptr2, new_sign );
35 )        else {
36 )          p1 = D.WalkHalfFace( d_ptr1, d_ptr2, this );
37 )          p2 = D.WalkHalfFace( d_ptr2, d_ptr1, this );
38 )          if ( p1.size ≤ p2.size )
39 )            p = p1;
40 )          else
41 )            p = p2;
42 )          if ( need_crosscap ) {
43 )            AddCrosscap( p.path_start.angle_ptr );
44 )            // put angle of loop on path list instead of
                  first angle
45 )            p.path_start.angle_ptr = D.V[n − 1];
46 )          }
47 )          SubdivideFeatureEdges( p.path_start );
48 )          AddEdgesBetweenFeatures( p.path_start );
49 )        }
50 )        d_ptr1.next.my_position = pu;
51 )        d_ptr2.next.my_position = pv;
52 )      }
53 )      ptr = ptr.next;
54 )    }
55 ) }
}
```

Algorithm 6.7. *Get Pair*

INPUT:

- an integer vertex label v corresponding to one of a pair in a feature of this depiction.

ACTION:

- returns the label of the other vertex of the pair corresponding to v.

```
GetPair( v ) {
1 ) if ( v < G.n )
2 )   return -1; // not paired
```

```
3 ) if ( G.n % 2 == v % 2 )
4 )   return (v + 1);
5 ) else
6 )   return (v − 1);
}
```

Algorithm 6.8. *Add Handle*

INPUT:

- an AdjNode *d_ptr*1 corresponding to the angle where we want to insert one end of an edge in this depiction,
- an AdjNode *d_ptr*2 corresponding to the angle where we want to insert the other end of an edge in this depiction,
- an integer *new_sign* for the signature of the edge to be inserted.

ACTION:

- one part of a handle structure added to this depiction, consisting of a new vertex labelled n with a loop, and a new edge between $d\_ptr1.twin.u$ and n,
- the other part of a handle structure added to this depiction, consisting of a new vertex labelled $n + 1$ with a loop, and a new edge between $d\_ptr2.twin.u$ and $n + 1$.

```
AddHandle( d_ptr1, d_ptr2, new_sign ) {
1 ) D.InsertLoop( d_ptr1 );
2 ) D.InsertLoop( d_ptr2 );
3 ) if ( new_sign == −1 )
4 )   type = TWISTED_HANDLE;
5 ) else
6 )   type = HANDLE;
7 ) feature_type[D.n − 2] = type;
8 ) feature_type[D.n − 1] = type;
9 ) feature_number[D.n − 2] =num_feature;
10 ) feature_number[D.n − 1] =num_feature;
11 ) num_feature++;
}
```

Algorithm 6.9. *Add Crosscap*

INPUT:

- an AdjNode *angle_ptr* corresponding to the angle where we want to insert an edge in this depiction.

ACTION:

- a crosscap structure added to this depiction, consisting of a new vertex labelled n with a loop that is subdivided to have a new vertex labelled $n+1$, and a new edge between *angle_ptr.twin.u* and n.

```
AddCrosscap( angle_ptr ) {
1 ) D.InsertLoop( angle_ptr );
2 ) D.Subdivide( D.V[D.n − 1] );
3 ) feature_type[D.n − 2] = CROSSCAP;
4 ) feature_type[D.n − 1] = CROSSCAP;
5 ) feature_number[D.n − 2] = num_feature;
6 ) feature_number[D.n − 1] = num_feature;
7 ) num_feature++;
}
```

Algorithm 6.10. *Subdivide Feature Edges*

INPUT:

- a PathNode pointer *p_ptr* corresponding to the start of a sequence of feature edges in this depiction, but note that the first and last item in the list correspond to angles of this depiction that are not on features.

ACTION:

- each feature edge in the list of *p_ptr* is subdivided, and the node in the list of *p_ptr* is replaced with a pointer to the new angle to be used for inserting edges later.

```
SubdivideFeatureEdges( p_ptr ) {
1 ) AdjNode sub_ptr;
2 ) int dir = 1;
3 ) // from second entry to second last one
```

```
4 ) while ( p_ptr.next ≠ NULL ) {
5 )   p_ptr = p_ptr.next;
6 )   if ( p_ptr.next ≠ NULL ) {
7 )     sub_ptr = p_ptr.angle_ptr;
8 )     boolean first_side_of_feature = ((G.n % 2) == (D.n % 2));
9 )     feature_type[D.n] = feature_type[sub_ptr.twin.u];
10 )    feature_number[D.n] = feature_number[sub_ptr.twin.u];
11 )    D.Subdivide( sub_ptr );
12 )    // replace subdivide edge pointer with angle of feature
13 )    if ( dir == 1 and first_side_of_feature )
14 )      p_ptr.angle_ptr = D.V[D.n − 1];
15 )    else if ( dir == 1 )
16 )      p_ptr.angle_ptr = D.V[D.n − 1].next;
17 )    else if ( dir == −1 and first_side_of_feature )
18 )      p_ptr.angle_ptr = D.V[D.n − 1].next;
19 )    else
20 )      p_ptr.angle_ptr = D.V[D.n − 1];
21 )    // walk over a twist?
22 )    if ( feature_type[p_ptr.angle_ptr.twin.u] ≠
        Depiction.HANDLE and first_side_of_feature )
23 )      dir = −dir;
24 )  }
25 ) }
}
```

Algorithm 6.11. *Add Edges Between Features*

INPUT:

- a PathNode pointer *p_ptr* for the start of a sequence of angles we want to add edges between.

ACTION:

- an edge inserted to this depiction for each $(2i)^{th}$ and $(2i+1)^{th}$ angles in the sequence of *p_ptr*.

```
AddEdgesBetweenFeatures( p_ptr ) {
1 ) while ( p_ptr ≠ NULL ) {
2 )   D.AddEdge( p_ptr.angle_ptr, p_ptr.next.angle_ptr, 1 );
```

```
3 )     p_ptr = p_ptr.next.next;
4 ) }
}
```

Chapter 7

Irreducible Triangulations

This chapter begins by summarizing a survey of basic results for irreducible triangulations in Section 7.1. Section 7.2 collects observations made, where the main result gives the different possible subembeddings of a region of an irreducible triangulation bound by homotopic cycles. Evidence towards the truth of the Successive Surface Scaffolding Conjecture is organized according to surface into Subsections 7.2.1 to 7.2.3.

7.1 Introduction

A *triangulation* of a surface $\mathbb{S}$ is an embedding on $\mathbb{S}$ such that every face is bound by exactly three edges. An edge e of a triangulation is *contractible* if contracting e and removing multiple edges results in another triangulation of the surface. An example of edges which are contractible is seen in Figure 7.7 of Section 7.2.2. An *irreducible* triangulation has no contractible edges, and it also has the property that any two triangles share at most one edge. Note that the condition limiting the number of shared edges between triangles forces an embedding of K_4 in the plane to be isomorphic to the only irreducible triangulation of the plane, as opposed to an embedding of K_3 which has two triangular faces sharing all three edges. One can use the unlabelled embedding of K_4 to generate all other triangulations of the plane that are not K_3 by vertex splitting and edge insertions [LN97]. Note that after an embedded vertex split on v is performed in some triangulation T so that new vertices v_1 and v_2 are obtained, the resulting graph has two faces f_1 and f_2 of size four incident on both v_1 and v_2, and inserting an edge into f_1 and

another edge into f_2 in any of the four possible ways will result in a larger triangulation than T.

Barnette proved there are exactly two irreducible triangulations of the projective plane [Bar82]. These are pictured in Figure 7.5 found in Section 7.2.1.

Lawrencenko and Negami [LN97] generated 25 irreducible triangulations for the Klein bottle. Their list is not complete. We explain what they missed after presenting some of their basic arguments. A foundational lemma is used which they refer to in establishing their results. The proof of Lemma 7.1 is also beneficial to observe.

Lemma 7.1. *[LN97] A triangulation T on any surface other than the sphere is irreducible if and only if every edge of T lies in an essential cycle of length three.*

Proof. Suppose T is an irreducible triangulation of some surface other than the sphere, and that there exists some edge e that is not in an essential 3-cycle. Then the end vertices of e could only have two common neighbours, those which complete the two triangles on either side of e. Hence, contracting e and removing any redundant edges results in a smaller triangulation, which contradicts T being irreducible.

On the other hand, suppose a triangulation T has every edge in an essential 3-cycle. Then for any edge e let one of its essential 3-cycles be $C = eu$ for some third vertex u. Then contracting e results in an embedding with multiple edges incident on u and removing any of them increases the size of some face to be larger than a triangle. Hence, T with e contracted is not a triangulation. Therefore, T is an irreducible triangulation. □

It is because of Lemma 7.1 that an edge e is called *essential* if e is in an essential 3-cycle of an embedding. Further, for any edge uv, without loss of generality for the vertex u, there are two triangular faces on either side of uv which give three incidences on u and an edge completing an essential 3-cycle with uv gives a fourth incidence on u. Therefore, Lemma 7.1 implies every vertex must have degree at least four.

Lawrencenko and Negami [LN97] incorrectly claim that there are only 25 irreducible triangulations for the Klein bottle. The correct total is 29 irreducible triangulations of the Klein bottle, the extra four were found by Sulanke [Sul06c]. One omission error by Lawrencenko and Negami arises from assuming there could not be a chord outside a certain planar region as

they attempt to build irreducible triangulations in all possible ways, but such a chord is actually possible. The other omission error is from missing possible ways of inserting edges into a partially triangulated embedding. Lawrencenko and Negami have structural properties that hold true for all the irreducible Klein-bottle triangulations, reconfirmed by Sulanke, that are aesthetic and concise, some are emphasized in the following thereom.

Theorem 7.2. *[LN97, Sul06c] Every irreducible triangulation of the Klein bottle has all of the following properties.*

(a) a disjoint pair of longitudes and a meridian which crosses each of the longitudes only once,

(b) a meridian and an equator which crosses each other at precisely two vertices,

(c) a Hamilton cycle which is planar on the Klein bottle,

(d) a Hamilton cycle which is a meridian,

(e) a Hamilton cycle which is a longitude, and

(f) a Hamilton cycle which is an equator.

Also, Theorem 12 of [LN97] states that "*a triangulation of the Klein bottle includes two disjoint meridians if and only if it does not include an equator of length* 3," and this is again reconfirmed by Sulanke [Sul06c]. Lemma 4 of [LN97] states that, *the irreducible triangulations of the Klein bottle can be classified into two disjoint classes, handle type and crosscap type.* The former, *handle type* irreducible triangulations have a meridian 3-cycle and no equator 3-cycle, while the latter, *crosscap type* irreducible triangulations have an equator 3-cycle and no meridian 3-cycle. There are 25 irreducible triangulations of handle type and 4 of crosscap type. The argument to obtain the classification is not trivial, and involves cutting along essential cycles to break up the surface into regions to force where the crosscaps must reside. There is only one irreducible triangulation of the Klein bottle that is also embeddable on the torus which they call $Kh1$ (where K denotes the Klein bottle and $h1$ denotes the first among handle types). A depiction of $Kh1$ is drawn flat with vertical boundary identified in antiparallel and horizontal boundary identified in parallel in the centre of Figure 7.15. Because every triangulation

of the Klein bottle can be constructed from some irreducible triangulation of the Klein bottle and a sequence of vertex splits, it follows immediately from Theorem 7.2 that properties (a)–(f) apply to every triangulation of the Klein bottle. See Table 7.4 for counts of irreducible triangulations of the Klein bottle based on their orders and minimum degrees.

Lawrencenko proved there are exactly 21 irreducible triangulations of the torus [Law87]. See Table 7.1 for counts of the torus irreducible triangulations based on their orders and minimum degrees.

Barnette and Edelson [BE88, BE89] proved that the number of irreducible triangulations for any surface is finite. They also state that all other triangulations can be generated by sequences of vertex splits plus inserting edges.

Sulanke [Sul06b] computed the irreducible triangulations of the surfaces of genus, 0, 1, and 2, for the orientable case, and 1, 2, 3, and 4, for the nonorientable case. See Tables 7.1 – 7.6. Computer files of the irreducible triangulations for $\mathbb{S}_2$, $\mathbb{N}_3$, and $\mathbb{N}_4$ are available online [Sul06a]. Also in Table 4 of [Sul06b], Sulanke establishes that any irreducible triangulation of $\mathbb{N}_3$ has a noncontractible separating cycle of length at most 6.

Joret and Wood [JW10] derived an upper bound on the order of any irreducible triangulation of a surface S to be $13g-4$, where g is the Euler genus of S. This evaluates to 22 for the Klein bottle, yet the largest irreducible Klein triangulation has only 11 vertices.

There are other results by Sulanke in [Sul06a] to consider, such as the following. Take any vertex v in an irreducible triangulation. Let $u_1, u_2, \ldots, u_d$ be the cyclic order of the neighbours about vertex v. Define the $link(v)$ as the cycle $u_1u_2 \ldots u_d$. Every vertex must lie on at least two edge-disjoint nonseparating essential cycles. An essential cycle C through v has two other vertices $u, w \in link(v)$, and let P be a shortest path between u and w on $link(v)$. A sufficient condition for C to be nonseparating is for P to be of minimum length among all other such P for each essential cycle through v.

7.2 Results

With the motivation to explore the Successive Surface Scaffolding Conjecture as much as possible, any further information about irreducible triangulations may enlighten directions toward its proof. This section presents new theorems with respect to irreducible triangulations.

Lemma 7.3. *Suppose $\widetilde{G}$ is an orientable embedding of genus g with $3n+$*

n	irreducible	$\delta \geq 3$	$\delta \geq 4$	$\delta \geq 5$	$\delta \geq 6$
7	1	1	1	1	1
8	4	7	6	4	1
9	15	112	75	24	2
10	1	2109	887	112	1

Table 7.1: ([Sul06a]) Number of triangulations of the torus, $\mathbb{S}_1$. For all tables of triangulation counts δ denotes minimum degree of the triangulations. There is a total of 21 irreducible triangulations.

n	irreducible	$\delta \geq 3$	$\delta \geq 4$	$\delta \geq 5$	$\delta \geq 6$	$\delta \geq 7$
10	865	865	865	750	298	3
11	26276	113506	93684	53270	7044	4
12	117047	7085444	4377179	1470379	64820	6
13	159205	290085272	126901868	23973881	356293	–
14	54527	9022585751	2712461256	281502783	1429448	–
15	38195	231102712868	47018397869	2652648134	4644258	–
16	664	–	700632975127	–	–	–
17	5	–	9322057619556	–	–	–

Table 7.2: ([Sul06a]) Number of triangulations of the double torus, $\mathbb{S}_2$.

$6(g-1)$ edges, or a nonorientable embedding of genus k with $3(n+k-2)$ edges. Then $\widetilde{G}$ is an irreducible triangulation if and only if for every edge uv in $\widetilde{G}$, together u and v have at least three common neighbours; i.e., $|N(u) \cap N(v)| \geq 3$.

Proof. Suppose $\widetilde{G}$ is an irreducible triangulation. Lemma 7.1 gives an essential 3-cycle C for any edge $e = uv$ of $\widetilde{G}$, so that u and v have at least three common neighbours: in the triangle on one side of e, in the triangle on the other side of e, and the one in C.

If $\widetilde{G}$ is not an irreducible triangulation, then there exists some contractible, non-essential edge $e = uv$. Suppose the triangular faces on either side of e are formed with vertex t and vertex w. Then the only multiple edges formed by contracting e are from edges tu, tv, and wu and wv. Therefore, it cannot be that u and v share a third neighbour.

□

Without Lemma 7.3, in order to check that an embedding $\widetilde{G}$ is an ir-

n	irreducible	$\delta \geq 3$	$\delta \geq 4$	$\delta \geq 5$
6	1	1	1	1
7	1	3	2	0

Table 7.3: ([Sul06a]) Number of triangulations of the projective plane, $\mathbb{N}_1$.

n	irreducible	$\delta \geq 3$	$\delta \geq 4$	$\delta \geq 5$	$\delta \geq 6$
8	6	6	6	5	–
9	19	187	133	38	1
10	2	4462	1971	250	1
11	2	86968	23541	1246	0

Table 7.4: ([Sul06a]) Number of triangulations of the Klein bottle, $\mathbb{N}_2$, with total 29 irreducible triangulations.

reducible triangulation, one would need to find essential 3-cycles for every edge. However, Lemma 7.3 provides a simple and quick way to check if a triangluation is in fact an irreducible triangulation. The following new theorem characterizes the triangulation subembeddings that are possible between homotopic cycles.

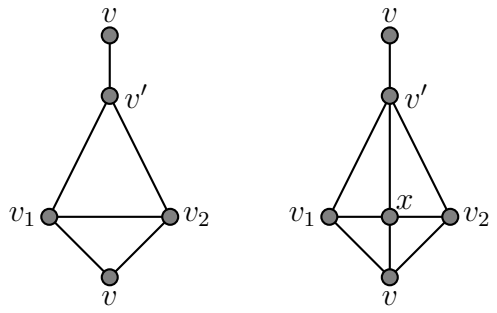


Figure 7.1: Case 1 for the triangulated region between homotopic essential 3-cycles $C_1 = vv_1v'$ and $C_2 = vv_2v'$.

Theorem 7.4. *The number of possible triangulations of a region R bound by two homotopic essential 3-cycles in an irreducible triangulation T is fourteen. See Figures 7.1–7.4.*

n	irreducible	$\delta \geq 3$	$\delta \geq 4$	$\delta \geq 5$	$\delta \geq 6$
9	133	133	133	111	23
10	2521	11784	9385	4523	190
11	4638	530278	298323	74307	758
12	1320	16306649	6162345	777348	2037
13	946	392973078	97905411	6255683	4574
14	93	8001174073	1308857389	42769083	8929
15	50	144075560093	15516720575	262073923	16026
16	7	–	168565283562	–	–

Table 7.5: ([Sul06a]) Number of triangulations of the surface $\mathbb{N}_3$, with 9708 total irreducible triangulations.

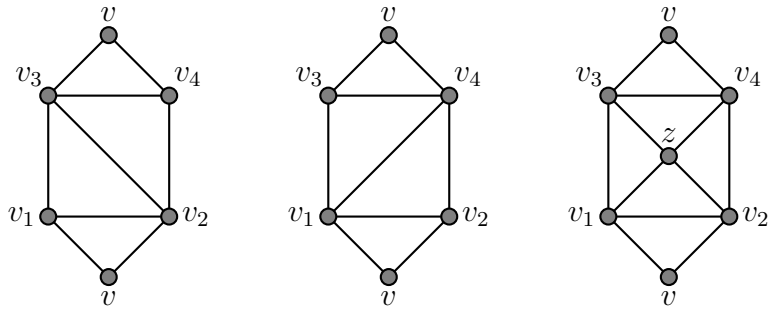


Figure 7.2: Case 2 for the triangulated region between homotopic essential 3-cycles $C_1 = vv_1v_3$ and $C_2 = vv_2v_4$.

Proof. Take any vertex v in an irreducible triangulation T of some surface $\mathbb{S}$. Consider the essential cycles through v. Suppose two of these essential cycles $C_1 \neq C_2$ are homotopic. Then $\mathbb{S} - C_1 - C_2$ has two regions one of which, say R, corresponds to a continuous deformation between C_1 and C_2. It must be that all other essential cycles completely contained in R are homotopic to C_1 and C_2, and go through v. Note that R is a pinched cylinder when C_1 and C_2 are two-sided, and R is a pinched Möbius strip when C_1 and C_2 are one-sided, with the pinched point at v in both instances. Also, it is trivial to see that the cycles C_1 and C_2 must not be separating. This is because some edges inside of R can be in an essential 3-cycle that partly consists of other edges outside of R. These edges are typically the edges between neighbours of v in R.

n	irreducible	$\delta \geq 3$	$\delta \geq 4$	$\delta \geq 5$	$\delta \geq 6$	$\delta \geq 7$
9	37	37	37	37	34	9
10	10347	13657	13067	10845	3864	36
11	370170	1628504	1314000	735766	87396	18
12	1891557	99694693	61111294	20568278	818072	28
13	2067817	4076362798	1787036930	341762288	4611407	–
14	956967	4076362798	38628806361	4088768693	18933909	–
15	700733	–	–	–	–	–
16	186999	–	–	–	–	–
17	89036	–	–	–	–	–
18	19427	–	–	–	–	–
19	3975	–	–	–	–	–
20	832	–	–	–	–	–
21	79	–	–	–	–	–
22	6	–	–	–	–	–

Table 7.6: ([Sul06a]) Number of triangulations of the surface $\mathbb{N}_4$.

Case 1. *Cycles C_1 and C_2 share a vertex at v' of $link(v)$.*

Let the edge e_1 of C_1 not be either of the two edges incident with v, and similarly, let e_2 be the edge of C_2 not incident with v. Necessarily, e_1 and e_2 share the common vertex v', and are chords of $link(v)$, with vertices v_1 and v_2, respectively, so that $e_1 = v'v_1$ and $e_2 = v'v_2$. Consider the path P on $link(v)$ from v_1 to v_2 that lies in the region R. For an internal vertex x on this path, there must be an edge vx and an essential cycle C_3 for vx which can only have v' as the third vertex of C_3, forcing C_3 to be homotopic to C_1 and C_2. Further, the vertex x must have another essential cycle $C_4 = v_1xv_2$ not homotopic to C_1 nor C_2. There can be at most one internal vertex on P, for if there were two internal vertices x and y, then the edge xy could not lie on any essential 3-cycle. If there are no internal vertices for P, then all the vertices and edges given so far form the triangulation of R. This case, therefore, contributes to two possible triangulations of R. See Figure 7.1.

Case 2. *Cycles C_1 and C_2 share no vertex of $link(v)$.*

Let $P_1 = v_1 \dots v_2$ and $P_2 = v_3 \dots v_4$ be the two paths that lie on $link(v)$ and intersect R. By the same argument as in the previous case, both P_1 and

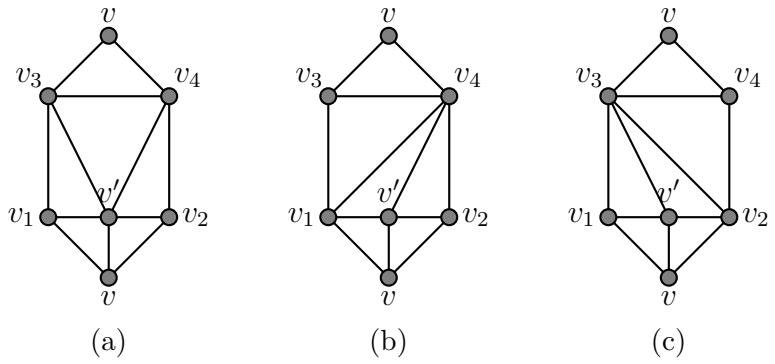


Figure 7.3: Case 3 for the triangulated region between homotopic essential 3-cycles $C_1 = vv_1v_3$ and $C_2 = vv_2v_4$.

P_2 may have at most one internal vertex. Continue to break into further cases, dealing with the case for no internal vertices here.

If there are no internal vertices on P_1 and P_2 and let $e_1 = v_1v_3$ and $e_2 = v_2v_4$ be the edges on C_1 and C_2, respectively. Then the region bound by P_1, P_2, e_1, and e_2 is a cycle f of size four which is temporarily called a face and is intended to be triangulated. Either there is a diagonal in f and the triangulation is completed, or there is only one vertex that sits in f. There cannot be more than one vertex x and y in f, since an edge xy in f cannot be part of an essential 3-cycle, and without the edge xy any triangulation of f must use a diagonal which forces both x and y to have degree 3, which contradicts every vertex of an irreducible triangulation having degree at least 4. Thus, if there is one vertex z in f, it must be adjacent to all four vertices of $f = v_1v_2v_4v_3$ and there must be at least two essential cycles on z not homotopic to C_1 nor C_2. Note that the only chords outside of R to satisfy this can be the edges v_1v_4 and v_2v_3. This case, therefore, contributes three possible triangulations of R. See Figure 7.2.

Case 3. *Path P_1 has an internal vertex $v' \in link(v)$, but P_2 does not.*

Let $f = v_1v'v_2v_4v_3$ be the region in R same as before, but now with another vertex v'. The vertex v' cannot be incident to a vertex z inside f, because $v'z$ cannot be part of an essential 3-cycle. Therefore, v' must be incident with an edge e that is incident at its other end on a vertex of P_2, either v_3 or v_4, so that $vv'e$ is an essential cycle and the region R breaks apart into previous cases. However, the region inside, for example $v'v_2v_4v_3$,

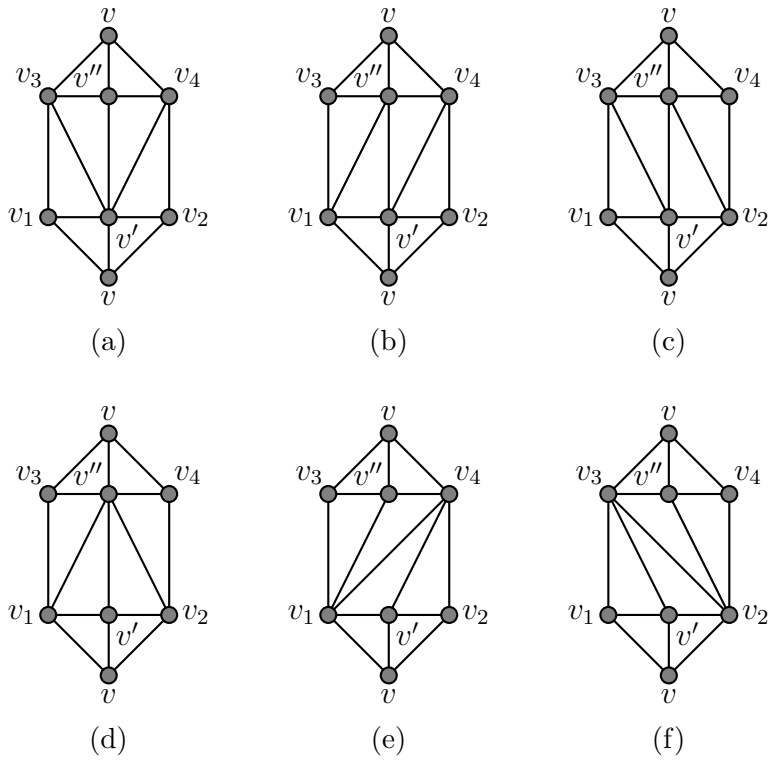


Figure 7.4: Case 4 for the triangulated region between homotopic essential 3-cycles $C_1 = vv_1v_3$ and $C_2 = vv_2v_4$.

cannot have an internal vertex, as it would have to be incident on v', and there cannot be a vertex inside of f incident on an internal vertex of P_1, as shown previously. Thus, this case can only have chords in the cycle f. Again, there must be an essential 3-cycle through path P_1 not homotopic to C_1 nor C_2. This case leads to three possible triangulations of R. See Figure 7.3.

Case 4. *Both P_1 and P_2 have internal vertices v' and v'', respectively.*

Let $f = v_1v'v_2v_4v''v_3$ be as before but now including the vertices v' and v''. There can be no vertex x inside f, since such a vertex must unavoidably be adjacent to either v' or v'', but this leads to a contradiction as in previous cases. Then six triangulations of f are possible, those which do not insert edges to connect the endpoints of P_1 or connect the endpoints of P_2. Notice how one can add an edge to obtain previous cases. This last case, therefore, contributes six possible triangulations of R. See Figure 7.4.

□

Now that the possible structures between homotopic 3-cycles are characterized, we consider the conditions that trivial cycles must satisfy in an irreducible triangulation. This leads to another characterization of irreducible triangulations.

Lemma 7.5. *A triangulation T of a surface $\mathbb{S}$ is irreducible if and only if for every edge e of T there is no trivial cycle C vertex-disjoint from e such that e is in the region bound by C homeomorphic to a disc, and every pair of triangular faces of T share at most one edge.*

Proof. First, note that the statement holds true on the sphere for the only irreducible triangulation K_4, since any edge and cycle of K_4 cannot be vertex-disjoint with each other. The condition that every pair of triangular faces share at most one edge is included to ensure that K_3 is not an irreducible triangulation on the sphere. Consider any surface other than the sphere for the remainder of the proof.

Suppose T is irreducible and there exists such an edge e vertex disjoint with planar cycle C. Any two edges incident with e must also lie in the region homeomorphic to a disc, and could then not be part of an essential 3-cycle with e. But then e is contractible, which contradicts that T is irreducible by Lemma 7.1.

On the other hand, if T is not irreducible, then there is an edge $e = (v_1, v_2)$ which when v_1 and v_2 are identified as vertex v and multiple edges are removed, the result is a new triangulation T' of $\mathbb{S}$. Then $N(v)$ is a planar cycle C, since T' is a triangulation. Obviously, C is also a planar cycle in T and is vertex disjoint from e. □

A corollary to Lemma 7.5 helps us decide which embeddings of torus obstructions are not worth trying to extend to irreducible triangulations of $\mathbb{N}_3$.

Corollary 7.6. *An embedding $\widetilde{G}$ in a surface $\mathbb{S}$ other than the sphere, cannot be extended to an irreducible triangulation of $\mathbb{S}$ if $\widetilde{G}$ has an edge e and a trivial cycle C vertex-disjoint from e such that e is in the planar region bound by C.*

Call an edge of an embedding which satisfies Corollary 7.6 *planar bound*, embeddings with such edges *unextendable* and otherwise *favourable*. Then

does every connected obstruction in $M_3(\mathbb{S}_1)$ have at least one favourable embedding? If an embedding of such an obstruction is favourable, is it always possible to extend it to an irreducible triangulation of $\mathbb{N}_3$? To this end, there must still be devised an algorithm for selecting which edges to add and in which faces of such favourable obstruction embeddings.

7.2.1 Plane Obstructions in $\mathbb{N}_1$ Irreducible Triangulations

Figures 7.5 and 7.6 prove that the Successive Surface Scaffolding Conjecture is true for $k = 1$ because the plane obstructions are found as subgraphs of the irreducible triangulations of the projective plane.

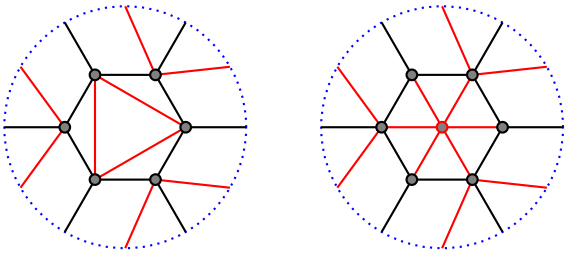


Figure 7.5: Irreducible triangulations of the projective plane obtained from adding red edges to embeddings of $K_{3,3}$.

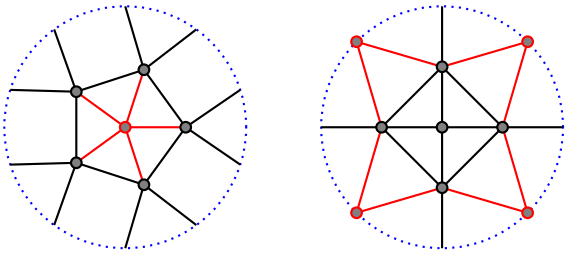


Figure 7.6: Irreducible triangulations of the projective plane obtained by adding one or two red vertices to the two embeddings of K_5. The red vertices are made adjacent to every edge of the face they are placed inside.

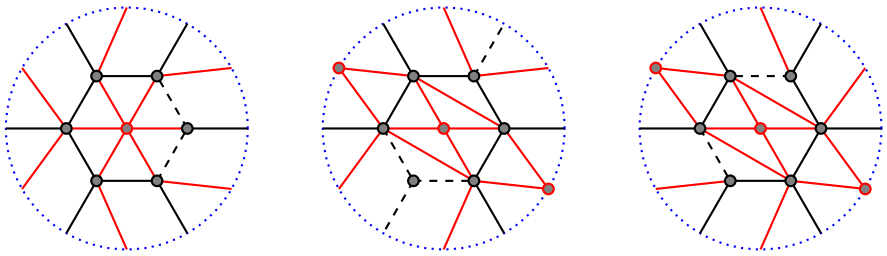


Figure 7.7: Triangulations of the projective plane obtained by adding red edges to embeddings of $K_{3,3}$. Some red vertices are also added. Edges which are contractible are dashed.

7.2.2 $\mathbb{N}_1$ Obstructions in $\mathbb{N}_2$ Irreducible Triangulations

Figures 7.8 – 7.25 show by brute force that the Successive Surface Scaffolding Conjecture is true for the connected wye-delta-order projective-plane obstructions. At the right of each figure is a representation of each obstruction as a subgraph for clarity. The figures which have triangulations of $\mathbb{N}_2$ represented with a square boundary have their horizontal sides identified in parallel and vertical sides identified in antiparallel, as explicitly shown in Figures 7.8 and 7.9. The figures which have triangulations of $\mathbb{N}_2$ represented with a boundary of two joined hexagons have vertices on the boundary corners of each hexagon in antipodal identification. Each obstruction is named as found in Appendix A of Graphs and Surfaces [MT01].

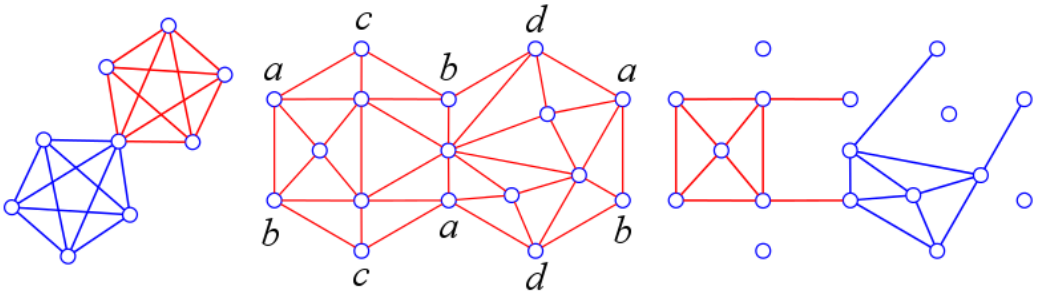


Figure 7.8: A_1 obstruction of $\mathbb{N}_1$ — $Kc4$ triangulation of $\mathbb{N}_2$

Consider the disconnected obstruction of two disjoint copies of K_5, which could only possibly be subgraphs of the Klein-bottle irreducible triangulations $Kh25, Kc2, Kc3$, or $Kc4$. For $Kh25$ depicted in Figure 7.13, there are

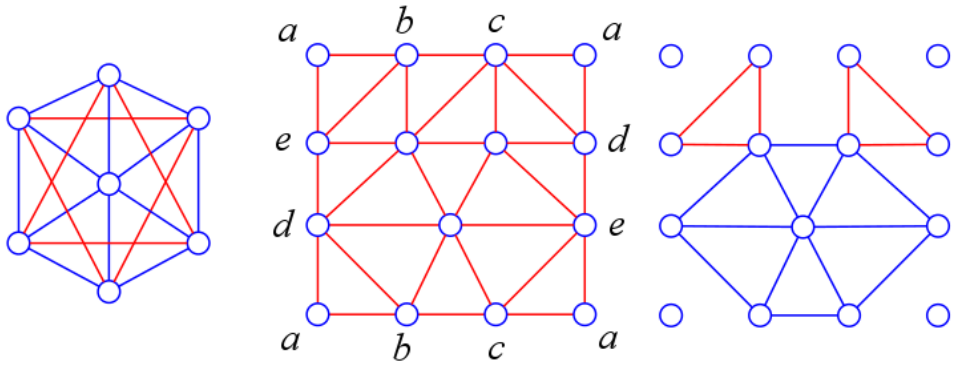


Figure 7.9: A_2 obstruction of $\mathbb{N}_1$ — $Kh6$ triangulation of $\mathbb{N}_2$

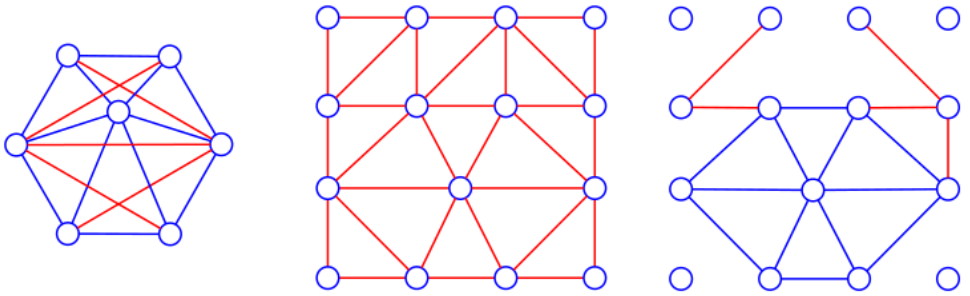


Figure 7.10: B_1 obstruction of $\mathbb{N}_1$ — $Kh6$ triangulation of $\mathbb{N}_2$

two vertices of degree four which share only two neighbours, so that two disjoint copies of K_5 as a subgraph is not possible. The irreducible triangulations $Kc2, Kc3$, and $Kc4$ can be obtained by taking two irreducible triangulations of the projective plane and identifying a triangle from both. Since there is only at most 7 vertices in a projective-plane irreducible triangulation, and only four other vertices apart from a triangle, none of $Kc2, Kc3$, and $Kc5$ contain two disjoint copies of K_5 as a subgraph.

Next, consider the graph E_2, shown in Figure 7.26, which is not in the set of wye-delta-order projective-plane obstructions, but is instead among the minor-order projective-plane obstructions. Note that E_2 is bipartite and has 11 vertices. The only irreducible triangulations of the Klein bottle that could possibly contain E_2 are $Kc3$ and $Kc4$ (see Figure 7.8). The argument for $Kc3$ is the same as for $Kc4$. Remove the three vertices a, b, and the vertex of degree 8 at the centre of the figure, so that two copies of K_4 minus

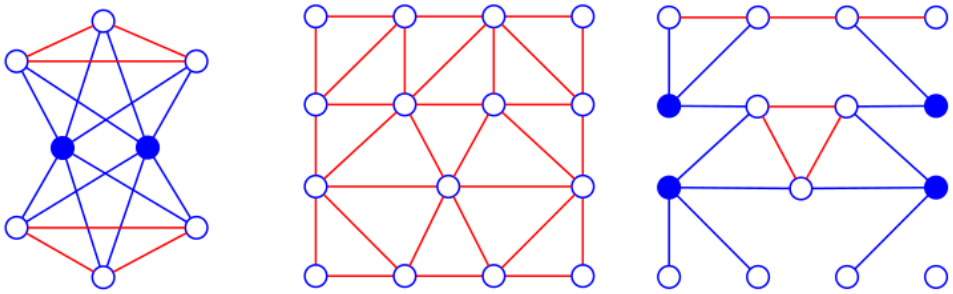


Figure 7.11: B_3 obstruction of $\mathbb{N}_1$ — $Kh6$ triangulation of $\mathbb{N}_2$

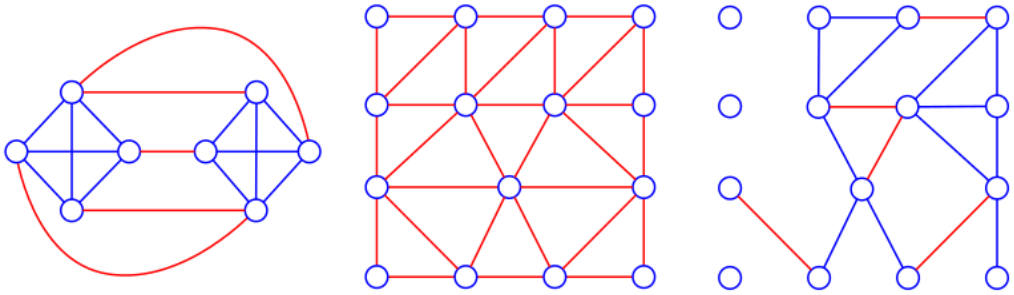


Figure 7.12: C_7 obstruction of $\mathbb{N}_1$ — $Kh4$ triangulation of $\mathbb{N}_2$

an edge remain as components. Since E_2 is bipartite, the only subgraphs that should appear if E_2 has three vertices removed are paths of length 4 or cycles of length 4, but E_2 always has a vertex of degree larger than 2 when any three vertices are removed. Therefore, E_2 cannot possibly be a subgraph of either $Kc3$ or $Kc4$. This establishes an example showing that the Successive Surface Scaffolding Conjecture is not always applicable for minor-order obstructions, nor topological obstructions of surfaces.

A complete search was done for obstructions of the projective plane within irreducible triangulations of the Klein bottle, either as a subgraph, or a subdivision. The algorithm used was a recursive search that considered marking each edge of an irreducible triangulation as either used or deleted at a given level of recursion. Pruning strategies for the recursive decision tree to speed up computation applied simple rules:

- the minimum number of edges a projective-plane obstruction can have; i.e., at least 15 edges;

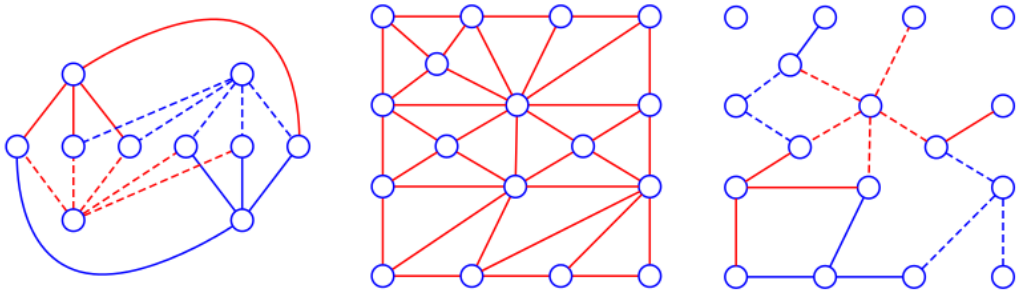


Figure 7.13: D_9 obstruction of $\mathbb{N}_1$ — $Kh25$ triangulation of $\mathbb{N}_2$

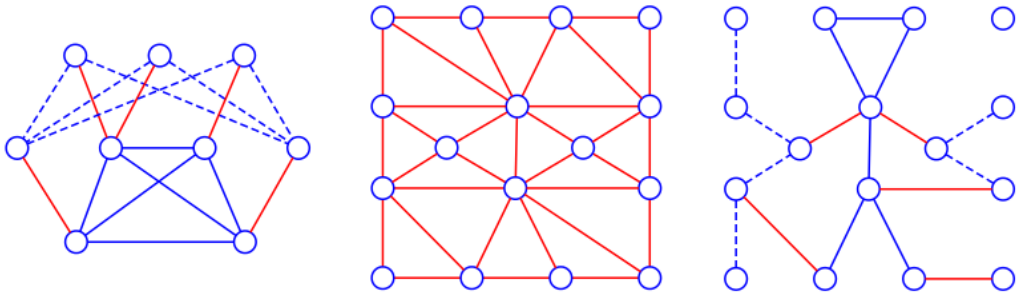


Figure 7.14: D_{12} obstruction of $\mathbb{N}_1$ — $Kh7$ triangulation of $\mathbb{N}_2$

- limiting the number of degree two vertices in a subgraph;
- limiting the number of isolated vertices in a subgraph;
- and not allowing degree one vertices.

The results were obtained by two different programmers each using different software where the results of both separate computations match. Tables 7.7 to 7.12 give the results with some rows omitted where every entry is zero. The obstructions are numbered in order of canonical form by `nauty` [MP14], and a correspondence with their names as listed in Graphs on Surfaces [MT01] is given in Appendix A. Also, the Klein-bottle irreducible triangulations are numbered as listed in the file `genus-2.alpha` available from Sulanke [Sul06a] within any version of a `surftri` download, and a correspondence with their names as listed in the work of Lawrencenko, Negami, and Sulanke [LN97, Sul06c] is given in Appendix B. For reference, the 103 projective-plane obstructions categorized into four types are partitioned as follows:

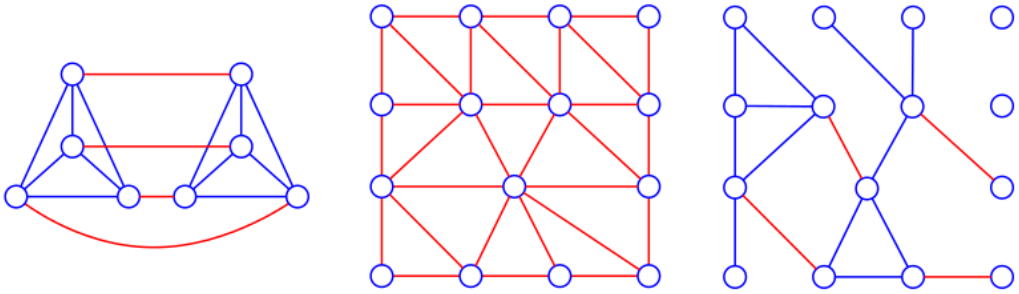


Figure 7.15: D_{17} obstruction of $\mathbb{N}_1$ — $Kh1$ triangulation of $\mathbb{N}_2$

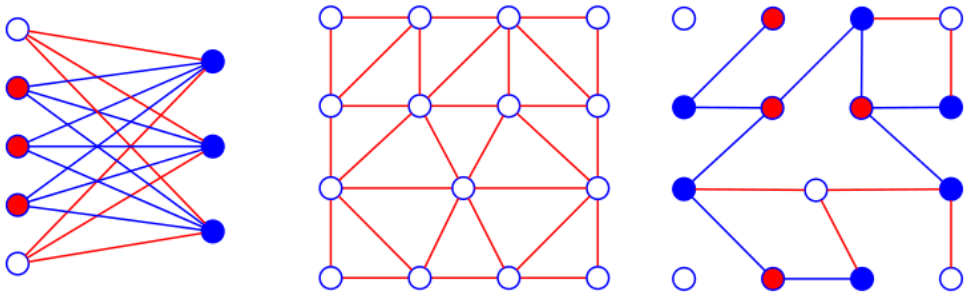


Figure 7.16: E_3 obstruction of $\mathbb{N}_1$ — $Kh6$ triangulation of $\mathbb{N}_2$

- topological, $\{3, 8, 10, 12, 15, 20, 22, 24, \ldots, 27, 29, 34, 35, 38, \ldots, 41, 44, 45, 47, \ldots, 52, 54, 55, 57, 58, 59, 61, 62, 63, 65, 66, 67, 70, \ldots, 85, 87, \ldots, 99, 101, 102\}$;
- minor, $\{7, 17, 18, 21, 28, 30, 33, 36, 42, 43, 46, 53, 68, 100\}$;
- wye-delta, $\{5, 16, 19, 23, 37, 56, 60, 69, 86\}$;
- double-wye-delta, $\{0, 1, 2, 4, 6, 9, 11, 13, 14, 31, 32, 64\}$;

and note that 64, 86, and 100 are each disconnected. For consideration of Conjecture 1.2, all connected double-wye-delta and wye-delta projective-plane obstructions are found as either a subgraph or subdivision of the irreducible triangulations of the Klein bottle. Note that 68, also referred to as E_2, is not found as a subgraph or subdivision, so that not all the minor obstructions of the projective plane can be found. The topological and minor obstructions that can be found as a subgraph or subdivision are:

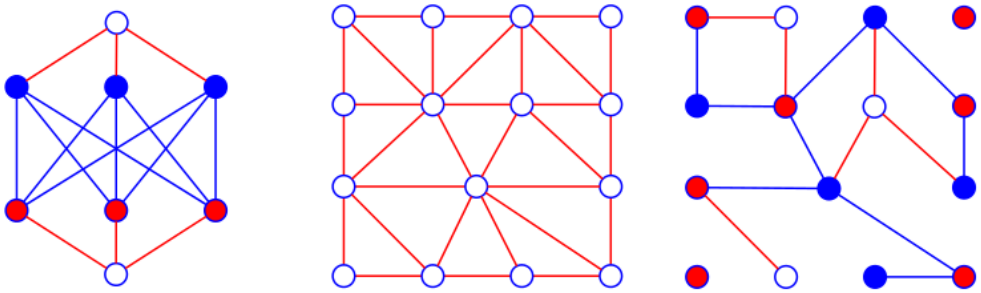


Figure 7.17: E_{18} obstruction of $\mathbb{N}_1$ — $Kh2$ triangulation of $\mathbb{N}_2$

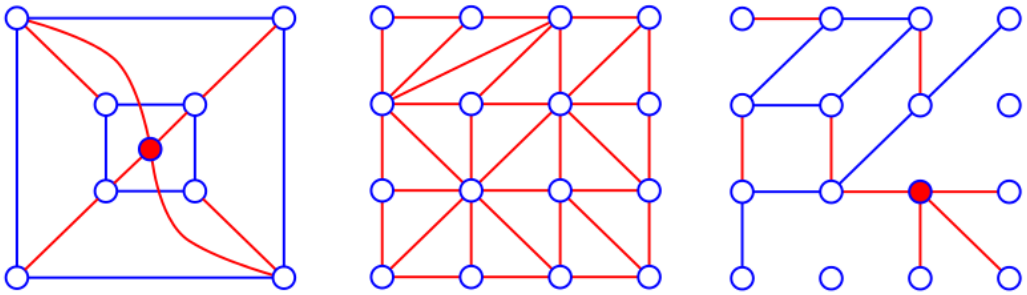


Figure 7.18: E_{22} obstruction of $\mathbb{N}_1$ — $Kh13$ triangulation of $\mathbb{N}_2$

- topological, $\{3, 8, 10, 12, 15, 20, 22, 24, \ldots, 27, 29, 35, 38, \ldots, 41, 44, 48, 57, 58, 59, 61, 62, 65, 72, 73, 74, 88\}$;
- minor, $\{7, 17, 18, 21, 28, 30, 33, 42, 43, 46, 53\}$;

and the topological and minor obstructions that cannot be found as a subgraph or subdivision are:

- topological, $\{34, 45, 47, 49, 50, 51, 52, 54, 55, 63, 66, 67, 70, 71, 75, \ldots, 85, 87, 89, \ldots, 99, 101, 102\}$;
- minor, $\{36, 68, 100\}$.

For consideration of Conjecture 1.1, all the connected double-wye-delta and wye-delta projective-plane obstructions are found as subgraphs only. The topological and minor obstructions that can be found as a subgraph are:

- topological, $\{3, 8, 10, 12, 15, 20, 22, 24, \ldots, 27, 29, 35, 38, \ldots, 41, 44, 48, 57, 58, 59, 61, 65, 72, 73, 74, 88\}$;

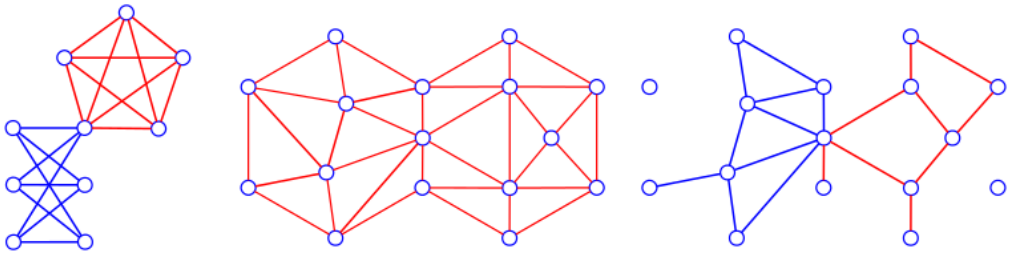


Figure 7.19: C_1 obstruction of $\mathbb{N}_1$ — $Kc2$ triangulation of $\mathbb{N}_2$

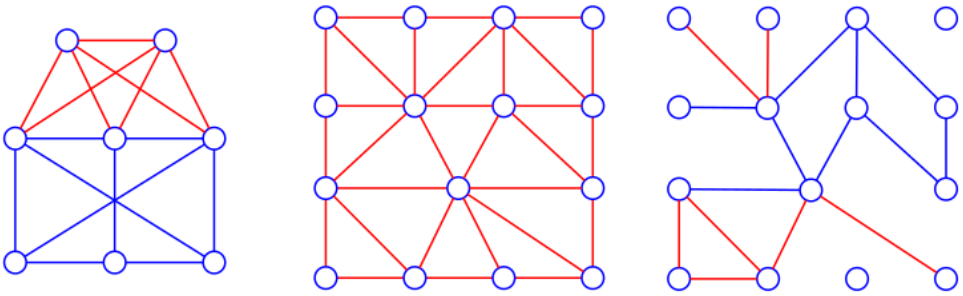


Figure 7.20: D_3 obstruction of $\mathbb{N}_1$ — $Kh2$ triangulation of $\mathbb{N}_2$

- minor, $\{7, 17, 18, 21, 28, 30, 33, 42, 43, 46, 53\}$;

and the topological and minor obstructions that cannot be found as a subgraph are:

- topological, $\{34, 45, 47, 49, 50, 51, 52, 54, 55, 62, 63, 66, 67, 70, 71, 75, \ldots, 85, 87, 89, \ldots, 99, 101, 102\}$;
- minor, $\{36, 68, 100\}$.

There are also results for those obstructions that are found as subdivisions only. All of the connected double-wye-delta obstructions of the projective-plane obstructions are found as only subdivisions of the irreducible triangulations of the Klein bottle. All connected wye-delta obstructions except 69 can be found as a subdivision only. The topological and minor obstructions that can be found as a subdivision only are:

- topological, $\{3, 8, 10, 12, 15, 20, 22, 24, 26, 29, 38, \ldots, 41, 48, 57, 59, 61\}$;

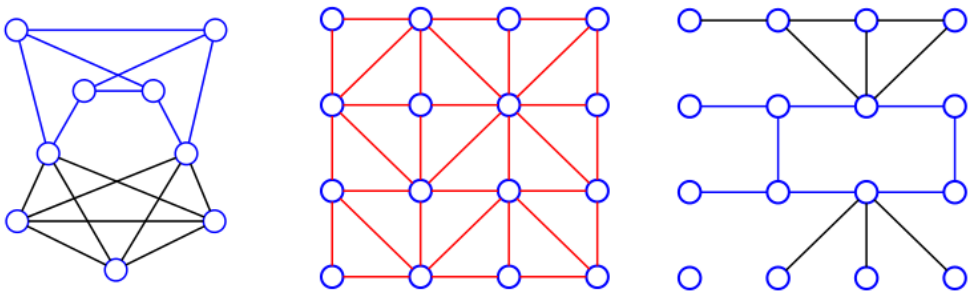


Figure 7.21: D_4 obstruction of $\mathbb{N}_1$ — $Kh19$ triangulation of $\mathbb{N}_2$

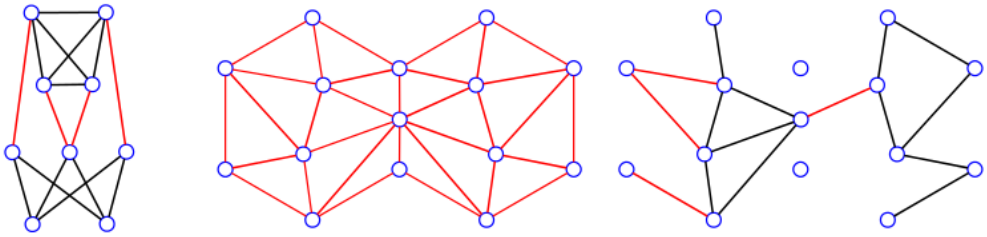


Figure 7.22: E_{19} obstruction of $\mathbb{N}_1$ — $Kc1$ triangulation of $\mathbb{N}_2$

- minor, $\{7, 17, 28, 30, 33, 43\}$;

and the topological and minor obstructions that cannot be found as a subdivision only are:

- topological, $\{25, 27, 34, 35, 44, 45, 47, 49, \ldots, 52, 54, 55, 58, 62, 63, 65, 66, 67, 70, \ldots, 85, 87, \ldots, 99, 101, 102\}$;
- minor, $\{18, 21, 36, 42, 46, 53, 68, 100\}$.

7.2.3 Torus Obstructions in $\mathbb{N}_3$ Irreducible Triangulations

In efforts towards finding evidence for a way to prove Conjecture 1.1, a set of 6313 obstructions of the torus in $M_3(\mathbb{S}_1)$ have been calculated. These graphs were found by using R_3 operations on 16683 minor-order obstructions of the torus found by Chambers [Cha02], and Woodcock [Woo06]. All unique

obs/tri	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	248	169	169	124	208	90	487	741	570	464	244	230	548	312
1	0	66	66	96	96	113	112	0	0	117	176	233	0	121
2	4	1	1	2	1	3	17	42	22	24	52	20	4	13
3	48	102	102	100	72	96	310	120	152	200	384	320	303	351
4	16	8	8	22	6	24	41	26	100	64	104	136	30	50
5	120	96	96	130	78	142	403	624	492	498	564	654	331	490
6	12	3	3	4	3	4	5	42	14	14	4	6	10	6
7	0	72	72	72	36	72	100	0	0	76	232	92	40	178
8	48	102	102	116	60	120	294	120	80	172	420	324	194	384
9	120	72	72	112	51	136	90	312	164	224	116	194	158	268
10	72	45	45	58	30	68	136	246	140	152	396	246	88	189
11	24	18	18	28	12	32	22	48	32	48	28	48	38	61
12	0	0	0	0	0	0	23	36	24	42	48	26	4	26
13	0	0	0	0	0	0	6	0	0	6	16	12	4	18
14	0	0	0	0	0	0	34	30	28	64	58	50	19	64
15	0	0	0	0	0	0	34	54	28	56	60	50	18	56
16	0	0	0	0	0	0	36	72	76	100	142	94	24	86
17	0	0	0	0	0	0	19	33	26	36	38	29	9	20
18	0	0	0	0	0	0	0	0	0	0	12	0	2	10
19	0	0	0	0	0	0	8	48	30	44	68	41	20	27
20	0	0	0	0	0	0	98	48	32	48	256	208	76	160
21	0	0	0	0	0	0	14	0	0	0	20	12	10	18
22	0	0	0	0	0	0	32	24	16	24	80	62	22	52
23	0	0	0	0	0	0	0	12	6	8	6	5	3	3
24	0	0	0	0	0	0	34	30	64	72	132	110	20	56
25	0	0	0	0	0	0	10	0	0	0	48	24	4	24
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	8	0	0	0	32	16	6	10
28	0	0	0	0	0	0	3	9	2	4	2	2	0	1
29	0	0	0	0	0	0	11	30	26	34	64	32	7	25
30	0	0	0	0	0	0	24	24	24	40	76	56	22	45

Table 7.7: Columns 1 to 14 of the total occurrences of projective-plane obstructions within the irreducible triangulations of the Klein bottle as either a subgraph or as a subdivision.

obs/tri	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
0	455	289	132	36	601	1062	398	404	627	630	627	1899	1706	1717	1657
1	118	103	126	90	102	0	0	0	0	81	0	0	0	0	0
2	7	6	12	72	42	42	48	8	21	42	42	35	70	0	0
3	300	357	222	198	372	0	176	288	318	354	270	0	184	0	0
4	35	58	84	180	62	0	32	112	30	28	30	0	54	0	0
5	446	539	684	1062	444	972	1004	744	354	366	306	3090	2468	7948	8340
6	16	10	21	21	0	237	121	48	0	0	0	791	196	1876	2912
7	98	126	210	108	96	0	0	0	104	180	72	0	0	0	0
8	282	354	474	252	372	0	192	256	300	354	234	0	192	0	0
9	196	356	726	972	0	1026	1140	548	0	0	0	2670	1078	6536	6128
10	142	192	162	198	216	486	470	252	84	252	108	753	546	1032	1352
11	46	90	192	270	0	0	176	120	0	0	0	0	128	0	0
12	10	10	36	108	30	18	38	16	24	24	36	134	218	304	408
13	6	12	30	36	24	0	0	0	24	0	0	0	0	0	0
14	22	41	114	162	36	0	44	36	60	12	18	0	128	0	0
15	30	62	102	144	72	108	140	92	60	30	18	588	196	1728	3704
16	43	60	156	234	66	54	90	88	54	36	0	576	722	3012	2128
17	17	29	30	90	42	27	31	34	18	21	27	204	210	796	1676
18	2	6	18	18	24	0	0	0	14	0	0	0	0	0	0
19	23	40	84	162	0	108	168	70	0	15	9	642	316	3100	1540
20	118	184	348	360	0	0	128	144	0	120	36	0	224	0	0
21	12	24	36	0	0	0	0	0	0	18	18	0	0	0	0
22	34	52	114	108	0	0	64	64	0	36	18	216	176	816	688
23	4	3	6	0	0	36	32	14	0	0	0	360	112	2024	1160
24	29	50	114	252	48	0	36	76	24	30	18	0	80	0	0
25	10	24	84	252	0	0	24	16	0	12	0	0	0	0	0
26	0	0	0	0	0	42	30	0	0	0	0	184	0	1204	0
27	9	8	12	0	0	0	16	8	0	18	0	0	0	0	0
28	3	3	0	6	6	24	14	8	3	3	3	194	56	492	1620
29	11	25	42	162	42	54	60	26	27	18	0	210	160	284	908
30	26	52	120	180	24	0	96	80	22	24	24	0	96	0	0
31	0	0	0	0	0	6	2	0	0	0	0	34	0	289	0
32	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0
33	0	0	0	0	0	0	0	0	0	0	0	28	20	336	272
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37	0	0	0	0	0	0	0	0	0	0	0	12	0	170	68
38	0	0	0	0	0	0	0	0	0	0	0	84	100	952	848
39	0	0	0	0	0	0	0	0	0	0	0	12	24	56	280
40	0	0	0	0	0	0	0	0	0	0	0	18	24	168	64
41	0	0	0	0	0	0	0	0	0	0	0	48	0	468	312
42	0	0	0	0	0	0	0	0	0	0	0	0	32	0	0
43	0	0	0	0	0	0	0	0	0	0	0	8	6	32	172
44	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0
45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
46	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0
47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
48	0	0	0	0	0	0	0	0	0	0	0	24	48	304	96
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
53	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
56	0	0	0	0	0	0	0	0	0	0	0	24	16	337	270
57	0	0	0	0	0	0	0	0	0	0	0	32	0	368	112
58	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0
59	0	0	0	0	0	0	0	0	0	0	0	6	12	24	128
60	0	0	0	0	0	0	0	0	0	0	0	18	12	200	200
61	0	0	0	0	0	0	0	0	0	0	0	2	0	34	0
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
65	0	0	0	0	0	0	0	0	0	0	0	0	0	80	64
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
69	0	0	0	0	0	0	0	0	0	0	0	0	0	27	18
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
72	0	0	0	0	0	0	0	0	0	0	0	0	0	12	12
73	0	0	0	0	0	0	0	0	0	0	0	0	0	55	50
74	0	0	0	0	0	0	0	0	0	0	0	0	0	10	4
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
88	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Table 7.8: Columns 15 to 29 of the total occurrences of projective-plane obstructions within the irreducible triangulations of the Klein bottle as either a subgraph or as a subdivision.

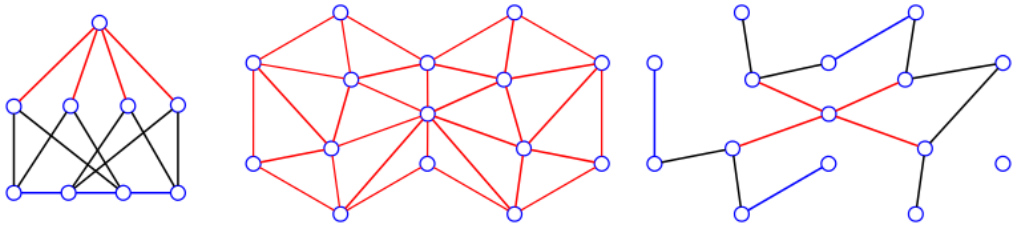


Figure 7.23: F_1 obstruction of $\mathbb{N}_1$ — $Kc1$ triangulation of $\mathbb{N}_2$

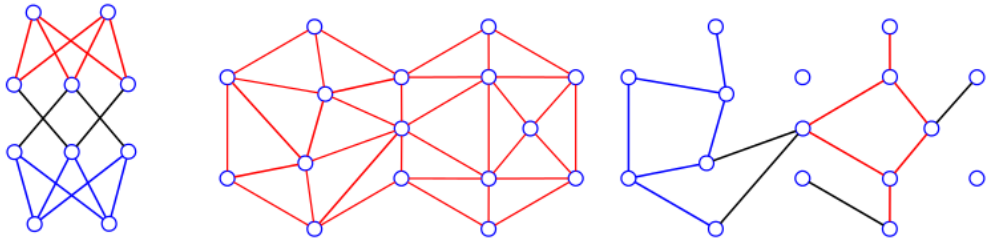


Figure 7.24: F_6 obstruction of $\mathbb{N}_1$ — $Kc2$ triangulation of $\mathbb{N}_2$

embeddings up to isomorphism of nonorientable genus less than four for the 6310 connected obstructions from this set have been generated. See Table 7.13 for counts partitioned by order and number of edges. Also, of the 6313 wye-delta order obstructions, 3149 were calculated as double-wye-delta by performing R_4 operations. Faster algorithms are still needed to find torus obstructions within irreducible triangulations of $\mathbb{N}_3$. Part of the issue is deciding which edges to keep from the irreducible triangulation when there are $3n+3$ edges to choose from. We conjecture that for a torus obstruction G, if an irreducible triangulation T of $\mathbb{N}_3$ contains G, then the graph $T - E(G)$ must be planar.

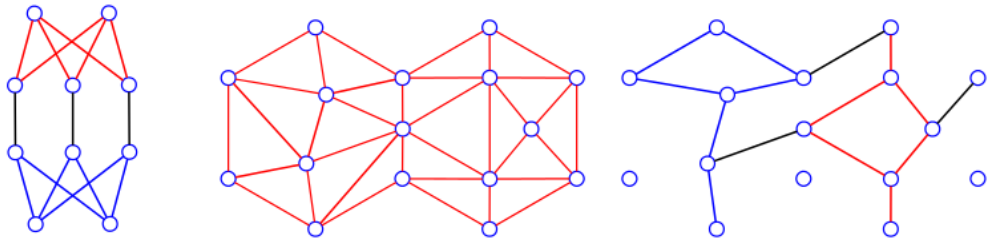


Figure 7.25: G_1 obstruction of $\mathbb{N}_1$ — $Kc2$ triangulation of $\mathbb{N}_2$

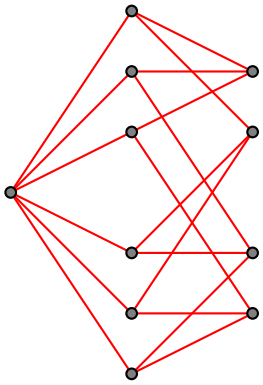


Figure 7.26: The minor-order projective-plane obstruction E_2.

obs/tri	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	228	157	157	118	182	88	480	731	558	458	244	230	535	311
1	0	63	63	90	81	108	109	0	0	114	175	230	0	120
2	0	0	0	0	0	0	15	36	18	22	46	17	3	12
3	0	0	0	0	0	0	262	96	120	164	356	284	252	311
4	0	0	0	0	0	0	34	22	80	58	98	114	24	44
5	0	0	0	0	0	0	336	531	412	430	532	565	270	429
6	0	0	0	0	0	0	4	36	12	12	4	6	8	5
7	0	0	0	0	0	0	82	0	0	56	212	84	34	152
8	0	0	0	0	0	0	246	96	64	136	392	288	160	336
9	0	0	0	0	0	0	68	252	132	180	116	176	126	219
10	0	0	0	0	0	0	114	207	116	132	360	214	70	166
11	0	0	0	0	0	0	16	36	24	36	28	44	30	49

Table 7.9: Columns 1 to 14 of the occurrences of projective-plane obstructions within the irreducible triangulations of the Klein bottle as a subdivision only.

obs/tri	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
0	449	289	132	36	594	1053	398	402	612	622	612	1896	1702	1716	1656
1	115	102	126	90	99	0	0	0	0	81	0	0	0	0	0
2	6	5	12	66	36	36	42	6	18	36	36	34	68	0	0
3	256	319	222	198	312	0	160	256	258	312	216	0	176	0	0
4	29	50	78	168	52	0	28	92	24	26	24	0	52	0	0
5	379	478	630	990	378	864	906	646	285	327	243	3030	2404	7920	8312
6	14	9	18	18	0	216	112	42	0	0	0	780	192	1872	2904
7	82	108	198	108	84	0	0	0	84	150	54	0	0	0	0
8	238	308	444	252	312	0	176	224	240	312	180	0	184	0	0
9	164	312	642	918	0	918	1038	476	0	0	0	2616	1038	6512	6104
10	121	170	162	198	180	432	426	218	66	219	90	738	532	1028	1348
11	38	78	168	252	0	0	160	104	0	0	0	0	120	0	0
12	0	0	0	0	0	0	0	0	0	0	0	112	176	296	400
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	96	0	0
15	0	0	0	0	0	0	0	0	0	0	0	480	156	1656	3584
16	0	0	0	0	0	0	0	0	0	0	0	480	588	2916	2048
17	0	0	0	0	0	0	0	0	0	0	0	168	168	764	1628
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	528	252	2992	1480
20	0	0	0	0	0	0	0	0	0	0	0	0	176	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	192	144	800	672
23	0	0	0	0	0	0	0	0	0	0	0	312	96	1980	1128
24	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	156	0	1176	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	168	48	480	1584
29	0	0	0	0	0	0	0	0	0	0	0	168	126	272	880
30	0	0	0	0	0	0	0	0	0	0	0	0	72	0	0
31	0	0	0	0	0	0	0	0	0	0	0	30	0	285	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	276	216
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
37	0	0	0	0	0	0	0	0	0	0	0	0	0	150	60
38	0	0	0	0	0	0	0	0	0	0	0	0	0	760	672
39	0	0	0	0	0	0	0	0	0	0	0	0	0	40	232
40	0	0	0	0	0	0	0	0	0	0	0	0	0	136	48
41	0	0	0	0	0	0	0	0	0	0	0	0	0	396	264
42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
43	0	0	0	0	0	0	0	0	0	0	0	0	0	24	144
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
48	0	0	0	0	0	0	0	0	0	0	0	0	0	240	64
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
56	0	0	0	0	0	0	0	0	0	0	0	0	0	276	216
57	0	0	0	0	0	0	0	0	0	0	0	0	0	312	96
58	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
59	0	0	0	0	0	0	0	0	0	0	0	0	0	16	104
60	0	0	0	0	0	0	0	0	0	0	0	0	0	160	160
61	0	0	0	0	0	0	0	0	0	0	0	0	0	30	0

Table 7.10: Columns 15 to 29 of the occurrences of projective-plane obstructions within the irreducible triangulations of the Klein bottle as a subdivision only.

obs/tri	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	20	12	12	6	26	2	7	10	12	6	0	0	13	1
1	0	3	3	6	15	5	3	0	0	3	1	3	0	1
2	4	1	1	2	1	3	2	6	4	2	6	3	1	1
3	48	102	102	100	72	96	48	24	32	36	28	36	51	40
4	16	8	8	22	6	24	7	4	20	6	6	22	6	6
5	120	96	96	130	78	142	67	93	80	68	32	89	61	61
6	12	3	3	4	3	4	1	6	2	2	0	0	2	1
7	0	72	72	72	36	72	18	0	0	20	20	8	6	26
8	48	102	102	116	60	120	48	24	16	36	28	36	34	48
9	120	72	72	112	51	136	22	60	32	44	0	18	32	49
10	72	45	45	58	30	68	22	39	24	20	36	32	18	23
11	24	18	18	28	12	32	6	12	8	12	0	4	8	12
12	0	0	0	0	0	0	23	36	24	42	48	26	4	26
13	0	0	0	0	0	0	6	0	0	6	16	12	4	18
14	0	0	0	0	0	0	34	30	28	64	58	50	19	64
15	0	0	0	0	0	0	34	54	28	56	60	50	18	56
16	0	0	0	0	0	0	36	72	76	100	142	94	24	86
17	0	0	0	0	0	0	19	33	26	36	38	29	9	20
18	0	0	0	0	0	0	0	0	0	0	12	0	2	10
19	0	0	0	0	0	0	8	48	30	44	68	41	20	27
20	0	0	0	0	0	0	98	48	32	48	256	208	76	160
21	0	0	0	0	0	0	14	0	0	0	20	12	10	18
22	0	0	0	0	0	0	32	24	16	24	80	62	22	52
23	0	0	0	0	0	0	0	12	6	8	6	5	3	3
24	0	0	0	0	0	0	34	30	64	72	132	110	20	56
25	0	0	0	0	0	0	10	0	0	0	48	24	4	24
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	8	0	0	0	32	16	6	10
28	0	0	0	0	0	0	3	9	2	4	2	2	0	1
29	0	0	0	0	0	0	11	30	26	34	64	32	7	25
30	0	0	0	0	0	0	24	24	24	40	76	56	22	45

Table 7.11: Columns 1 to 14 of the occurrences of projective-plane obstructions within the irreducible triangulations of the Klein bottle as a subgraph only.

obs/tri	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
0	6	0	0	0	7	9	0	2	15	8	15	3	4	1	1
1	3	1	0	0	3	0	0	0	0	0	0	0	0	0	0
2	1	1	0	6	6	6	6	2	3	6	6	1	2	0	0
3	44	38	0	0	60	0	16	32	60	42	54	0	8	0	0
4	6	8	6	12	10	0	4	20	6	2	6	0	2	0	0
5	67	61	54	72	66	108	98	98	69	39	63	60	64	28	28
6	2	1	3	3	0	21	9	6	0	0	0	11	4	4	8
7	16	18	12	0	12	0	0	0	20	30	18	0	0	0	0
8	44	46	30	0	60	0	16	32	60	42	54	0	8	0	0
9	32	44	84	54	0	108	102	72	0	0	0	54	40	24	24
10	21	22	0	0	36	54	44	34	18	33	18	15	14	4	4
11	8	12	24	18	0	0	16	16	0	0	0	0	8	0	0
12	10	10	36	108	30	18	38	16	24	24	36	22	42	8	8
13	6	12	30	36	24	0	0	0	24	0	0	0	0	0	0
14	22	41	114	162	36	0	44	36	60	12	18	0	32	0	0
15	30	62	102	144	72	108	140	92	60	30	18	108	40	72	120
16	43	60	156	234	66	54	90	88	54	36	0	96	134	96	80
17	17	29	30	90	42	27	31	34	18	21	27	36	42	32	48
18	2	6	18	18	24	0	0	0	14	0	0	0	0	0	0
19	23	40	84	162	0	108	168	70	0	15	9	114	64	108	60
20	118	184	348	360	0	0	128	144	0	120	36	0	48	0	0
21	12	24	36	0	0	0	0	0	0	18	18	0	0	0	0
22	34	52	114	108	0	0	64	64	0	36	18	24	32	16	16
23	4	3	6	0	0	36	32	14	0	0	0	48	16	44	32
24	29	50	114	252	48	0	36	76	24	30	18	0	20	0	0
25	10	24	84	252	0	0	24	16	0	12	0	0	0	0	0
26	0	0	0	0	0	42	30	0	0	0	0	28	0	28	0
27	9	8	12	0	0	0	16	8	0	18	0	0	0	0	0
28	3	3	0	6	6	24	14	8	3	3	3	26	8	12	36
29	11	25	42	162	42	54	60	26	27	18	0	42	34	12	28
30	26	52	120	180	24	0	96	80	22	24	24	0	24	0	0
31	0	0	0	0	0	6	2	0	0	0	0	4	0	4	0
32	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0
33	0	0	0	0	0	0	0	0	0	0	0	28	20	60	56
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37	0	0	0	0	0	0	0	0	0	0	0	12	0	20	8
38	0	0	0	0	0	0	0	0	0	0	0	84	100	192	176
39	0	0	0	0	0	0	0	0	0	0	0	12	24	16	48
40	0	0	0	0	0	0	0	0	0	0	0	18	24	32	16
41	0	0	0	0	0	0	0	0	0	0	0	48	0	72	48
42	0	0	0	0	0	0	0	0	0	0	0	0	32	0	0
43	0	0	0	0	0	0	0	0	0	0	0	8	6	8	28
44	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0
45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
46	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0
47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
48	0	0	0	0	0	0	0	0	0	0	0	24	48	64	32
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
53	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
56	0	0	0	0	0	0	0	0	0	0	0	24	16	61	54
57	0	0	0	0	0	0	0	0	0	0	0	32	0	56	16
58	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0
59	0	0	0	0	0	0	0	0	0	0	0	6	12	8	24
60	0	0	0	0	0	0	0	0	0	0	0	18	12	40	40
61	0	0	0	0	0	0	0	0	0	0	0	2	0	4	0
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
65	0	0	0	0	0	0	0	0	0	0	0	0	0	80	64
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
69	0	0	0	0	0	0	0	0	0	0	0	0	0	27	18
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
72	0	0	0	0	0	0	0	0	0	0	0	0	0	12	12
73	0	0	0	0	0	0	0	0	0	0	0	0	0	55	50
74	0	0	0	0	0	0	0	0	0	0	0	0	0	10	4
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
88	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Table 7.12: Columns 15 to 29 of the occurrences of projective-plane obstructions within the irreducible triangulations of the Klein bottle as a subgraph only.

n/m	18	19	20	21	22	23	24	25	26	27	28	29	30	total
8	0	0	0	0	1	0	1	1	0	0	0	0	0	3
9	0	2	5	2	7	13	4	1	4	0	0	0	0	38
10	0	14	3	15	16	71	76	88	65	17	1	0	1	367
11	5	2	0	31	93	287	735	569	154	24	7	3	1	1911
12	1	0	0	33	126	557	1006	719	161	40	14	1	0	2658
13	0	0	0	4	42	209	383	361	109	25	4	0	0	1137
14	0	0	0	0	2	10	45	95	38	2	0	0	0	192
15	0	0	0	0	0	0	1	3	3	0	0	0	0	7
total	6	18	8	85	287	1147	2251	1837	534	108	26	4	2	6313

Table 7.13: Number of known wye-delta-order obstructions on the torus by order (n) and size (m).

Chapter 8

Essential Cycles

This chapter shows in Section 8.1 how an embedding can be cut open along a cycle C so that the resulting embedding can be checked for properties that determine whether C is essential. After this is established by Theorem 8.1, Section 8.2 formalizes cutting along a cycle as a subroutine of Algorithm 8.3.

It should be emphasized that the problem of finding essential cycles in embeddings has been researched by de Verdiere [dV12]. For an arbitrary embedding, de Verdiere has a $\mathcal{O}(n^2)$ algorithm for finding a shortest essential cycle.

8.1 Cutting Along a Cycle

Consider a two-sided cycle C of an embedding $\widetilde{G}$. Remove C from $\widetilde{G}$ and replace it with two copies C_1 and C_2, where C_1 is connected to $\widetilde{G}$ with the adjacencies of one side of C and C_2 is connected to $\widetilde{G}$ with the adjacencies of the other side of C such that we preserve the order from the rotation systems of vertices on C. If the number of components of the underlying graph of this construction increased from the number of components in $\widetilde{G}$, then C is called *separating*, and otherwise C is *nonseparating*. Such a construction is called *cutting* $\widetilde{G}$ *along* C. This construction can be extended to any number of cycles by cutting along them in any sequence while adjusting for copies of vertices created in cutting along each cycle. One way to consider depicting an embedding on the plane can involve cutting along cycles of the embedding until only genus zero components remain. A cycle of an embedding on the Klein bottle is called a *longitude* if it is one-sided, a *meridan* if two-sided and

nonseparating, and an *equator* if two-sided and separating.

The same can be defined on the nonorientable surface $\mathbb{N}_3$, but with further requirements:

A cycle C of an embedding on $\mathbb{N}_3$ is called a *longitude* if it is one-sided and cutting along C results in an embedding on the Klein bottle. If cutting along C results in a torus embedding, then C is called a *latitude*.

A cycle C of an embedding on $\mathbb{N}_3$ is called a *meridian*, again, if it is two-sided and nonseparating.

A cycle C of an embedding on $\mathbb{N}_3$ is called an *equator* if it is two-sided, separating, and cutting along C leaves one projective-planar component plus a Klein component. If the components are projective-planar and toroidal, then C is called a *tropic*.

A pair C_1 and C_2 of one-sided cycles that are homotopic simple closed curves must cross each other an odd number of times. When C_1 and C_2 are two-sided and homotopic, they must cross an even number of times. To see the two-sided case, each consecutive pair of crossings between simple closed curves corresponding to C_1 and C_2 can be continuously deformed away, until all crossings are eliminated and the two curves form the boundaries of a cylinder. If a pair of homotopic one-sided cycles could do the same, then such a cylinder would erroneously show C_1 and C_2 each have two sides.

Two essential cycles C_1 and C_2 of an irreducible triangulation T are homotopic when cutting along both C_1 and C_2 results in a set of components that together contain a copy of both C_1 and C_2 that are altogether plane embeddings. To see this, note that the the segments of C_1 and C_2 between each consecutive pair of crossings can be continuously deformed to each other, which defines each planar component.

Consider an embedding $\widetilde{G}$ with oriented walk

$$C = u_1(s_1 = +1)u_2s_2 \dots u_ps_p(u_{p+1} = u_1)$$

with corresponding rotation systems π_{u_i} for $i = 1, 2, \dots, k$ where $k = p$ if C is two-sided, and $2k = p$ if C is one-sided. Construct two copies of cycle $u_1u_2 \dots u_k$ with $C_1 = v_1v_2 \dots v_k$ and $C_2 = w_1w_2 \dots w_k$ and obtain the corresponding rotation systems of the vertices of C_1 and C_2 from each π_{u_i} as follows. The edges $u_{i-1}u_i$ and u_iu_{i+1} modulo k determine two compound angles of π_{u_i}:

$$L_i = u_{i-1}, \pi_{u_i}^{1s_{i-1}}(u_{i-1}), \pi_{u_i}^{2s_{i-1}}(u_{i-1}), \dots, \pi_{u_i}^{(j-1)s_{i-1}}(u_{i-1}), \pi_{u_i}^{js_{i-1}}(u_{i-1})$$

such that $\pi_{u_i}^{js_{i-1}}(u_{i-1}) = u_{i+1}$, and

$$R_i = u_{i+1}, \pi_{u_i}^{1s_{i-1}}(u_{i+1}), \pi_{u_i}^{2s_{i-1}}(u_{i+1}), \ldots, \pi_{u_i}^{(q-1)s_{i-1}}(u_{i+1}), \pi_{u_i}^{qs_{i-1}}(u_{i+1})$$

such that $\pi_{u_i}^{qs_{i-1}}(u_{i+1}) = u_{i-1}$. Then the new rotation systems are set as

$$\pi_{v_i} = v_{i-1}, L_i \setminus \{u_{i-1}, u_{i+1}\}, v_{i+1},$$

and

$$\pi_{w_i} = w_{i+1}, R_i \setminus \{u_{i-1}, u_{i+1}\}, w_{i-1},$$

if $s_{i-1} = +1$, and

$$\pi_{v_i} = v_{i+1}, R_i \setminus \{u_{i-1}, u_{i+1}\}, v_{i-1},$$

and

$$\pi_{w_i} = w_{i-1}, L_i \setminus \{u_{i-1}, u_{i+1}\}, w_{i+1},$$

otherwise. The rotation systems for v_1, v_k, w_1, and w_k are defined as above, but the first and last vertex of the corresponding rotation systems must be set dependent on the sided property of C. If C is one-sided:

- π_{v_1} must start with w_k and end with v_2,
- π_{v_k} must start with v_{k-1} and end with w_1,
- π_{w_1} must start with v_k and end with w_2,
- π_{w_k} must start with w_{k-1} and end with v_1.

If C is two-sided:

- π_{v_1} starts with v_k and ends with v_2;
- π_{v_k} starts with v_{k-1} and ends with v_1;
- π_{w_1} starts with w_2 and ends with w_k;
- π_{w_k} starts with w_1 and ends with w_{k-1}.

This either results in two combinatorial embeddings $\widetilde{G}_1$ and $\widetilde{G}_2$ when C is separating, or a new combinatorial embedding $\widetilde{G}'$ when C is non-separating. Note that the edges between the v_i and w_i have signature the same as for their corresponding edges incident to u_i.

To see the effect that cutting along C has on the Euler genus of the embedding, one case is when C is separating, and the other is when C is not separating. This is explained in the following theorem.

Theorem 8.1. *Let g be the Euler genus of an embedding $\widetilde{G}$. If the cycle C in $\widetilde{G}$ is separating, then cutting along C results in two component embeddings $\widetilde{G}_1$ and $\widetilde{G}_2$ with Euler genus g_1 and g_2 such that $g = g_1 + g_2$. If the cycle C is nonseparating, then cutting along C results in a new embedding $\widetilde{G}'$ with Euler genus g' such that $g = g' + 1$ if C is one-sided, and $g = g' + 2$ if C is two-sided.*

Proof.

Case 1. *Suppose each component embedding $\widetilde{G}_1$ and $\widetilde{G}_2$ have Euler genus, respectively,*

$$g_1 = 2 - n_1 + m_1 - f_1 \text{ and } g_2 = 2 - n_2 + m_2 - f_2,$$

where $\widetilde{G}_1$ and $\widetilde{G}_2$ contain facial walks

$$v_1 s_1 v_2 s_2 \dots v_k s_k v_1 \text{ and } w_1 s_k w_k s_{k-1} w_{k-1} \dots s_2 w_2 s_1 w_1,$$

respectively.
By identifying $v_1 v_2 \dots v_k$ back with $w_1 w_2 \dots w_k$ to obtain the original embedding $\widetilde{G}$, then the Euler genus of $\widetilde{G}$ must be

$$\begin{aligned} g &= 2 - n + m - f \\ &= 2 - (n_1 + n_2 - k) + (m_1 + m_2 - k) - (f_1 + f_2 - 2) \\ &= (2 - n_1 + m_1 - f_1) + (2 - n_2 + m_2 - f_2) \\ &= g_1 + g_2. \end{aligned}$$

For the next cases $\widetilde{G}'$ is a new connected embedding corresponding to if C is nonseparable and has one facial walk

$$v_1 s_1 v_2 s_2 \dots v_k s_k w_1 s_{k+1} w_2 s_{k+2} \dots w_{k-1} s_{p-1} w_k s_p v_1,$$

if C is one-sided, and two facial walks

$$v_1 s_1 v_2 s_2 \dots v_k s_k v_1 \text{ and } w_1 s_k w_k s_{k-1} w_{k-1} \dots w_2 s_1 w_1,$$

if C is two-sided.

Case 2. *Suppose $\widetilde{G}'$ has Euler genus $g' = 2 - n' + m' - f'$, and that C is one-sided.*

Here, the identification of v_i and w_i results in one less face, so it must be that the Euler genus of $\widetilde{G}$ is

$$\begin{aligned} g &= 2 - n + m - f \\ &= 2 - (n' - k) + (m' - k) - (f' - 1) \\ &= 2 - n' + m' - f' + 1 \\ &= g' + 1. \end{aligned}$$

Intuitively, this case corresponds with removing a crosscap of $\widetilde{G}$.

Case 3. *Again, suppose $\widetilde{G}'$ has Euler genus $g' = 2 - n' + m' - f'$, but that now C is two-sided.*

Identifying v_i and w_i removes two faces of $\widetilde{G}'$. Then the Euler genus of $\widetilde{G}$ must be

$$\begin{aligned} g &= (2 - n + m - f) \\ &= 2 - (n' - k) + (m' - k) - (f' - 2) \\ &= (2 - n' + m' - f') + 2 \\ &= g' + 2. \end{aligned}$$

Intuitively, this last case corresponds with removing either a handle or a twisted handle of $\widetilde{G}$. □

8.2 Algorithm for Cutting

Algorithm 8.2 is a recursive method included in the `AugRotSystem` data structure that backtracks through all possible cycles of a graph and prints the cycles that are essential using `PrintCycle` on Line 27. Since Algorithm 8.2 backtracks through the adjacencies of each vertex to find cycles, it clearly runs in exponential time. The process of cutting along a cycle is given in Algorithm 8.3 and it is used in Algorithm 8.2 to determine all the essential cycles of the embedding corresponding to an instance of an `AugRotSystem` object. Algorithm 8.2 also takes a `ChordList` parameter, which has the same object structure as `PathList` from Chapter 6 except that the `PathNode` member types are replaced by `ChordNode` type, and that a `ChordNode` is modified to include a `ChordNode` *prev* member and a *parity* member to keep track of $+1$ clockwise or -1 counterclockwise orientation around each vertex visited.

Algorithm 8.2. *Print Essential Cycles*

INPUT:

- a `ChordList` *cycle_trace* that corresponds to edges that form a cycle of this `AugRotSystem`,
- a boolean array *visited* of size n for vertices used in the cycle,
- an integer *dir_parity* set to $+1$ for clockwise and -1 for counterclockwise orientation about a vertex,
- an integer *level* to track the depth of recursion.

OUTPUT:

- the essential cycles of the embedding corresponding to this `AugRotSystem`.

```
PrintEssentialCycles( cycle_trace, visited, dir_parity, level ) {
 1 ) if ( level == 0 ) {
 2 )   dir_parity = 1;
 3 )   for (i = 0; i < n; i++) {
 4 )     visited[i] = true;
 5 )     ptr = V[i];
 6 )     for (j = 0; j < degree[i]; j++) {
 7 )       cycle_trace.append( new ChordNode( i, ptr.u,
           cycle_trace.path_rear, ptr.sign, dir_parity );
 8 )       visited[ptr.u] = true;
 9 )       PrintEssentialCycles( cycle_trace, visited,
           dir_parity, level + 1 );
10 )       visited[ptr.u] = false;
11 )       cycle_trace.del_rear();
12 )       ptr = ptr.next;
13 )     }
14 )     visited[i] = false;
15 )   }
16 ) } else if ( level ≤ n ) {
17 )   u = cycle_trace.path_rear.u;
18 )   v = cycle_trace.path_rear.v;
19 )   ptr = V[v];
```

```
20 )   for (i = 0; i < degree[v]; i++) {
21 )     if (u == ptr.u) continue;
22 )     // close the path to a cycle
23 )     if (level > 1 and ptr.u == cycle_trace.path_start.u) {
24 )       dir_parity = dir_parity * cycle_trace.path_rear.prev.sign;
25 )       cycle_trace.append( new ChordNode( v, ptr.u,
           cycle_trace.path_rear, ptr.sign, dir_parity ) );
26 )       if ( IsEssential( cycle_trace, visited ) ) {
27 )         PrintCycle( cycle_trace, level + 1 );
28 )       }
29 )       cycle_trace.del_rear();
30 )       dir_parity = dir_parity * cycle_trace.path_rear.prev.sign;
31 )     }
32 )     // or extend the path
33 )     if ( !visited[ptr.u] ) {
34 )       dir_parity = dir_parity * cycle_trace.path_rear.prev.sign;
35 )       cycle_trace.append( new ChordNode( v, ptr.u,
           cycle_trace.path_rear, ptr.sign, dir_parity ) );
36 )       visited[ptr.u] = true;
37 )       PrintEssentialCycles( cycle_trace, visited,
           dir_parity, level + 1 );
38 )       visited[ptr.u] = false;
39 )       cycle_trace.del_rear();
40 )       dir_parity = dir_parity * cycle_trace.path_rear.prev.sign;
41 )     }
42 )     ptr = ptr.next;
43 )   } // end of for-loop i
44 ) }
}
```

Algorithm 8.3 is another method included in the `AugRotSystem` data structure, and uses the results of Theorem 8.1 to check whether a cycle is essential. Note that Lines 35 and 45 use the $\mathcal{O}(n)$ algorithm `SitsOnNewSide` that iterates through the adjacencies of $V[ptr.u]$ to see which side of the cycle *cycle_trace* the vertex u sits on with respect to $ptr.u$, which depends on the direction *parity* for each vertex visited in *cycle_trace*. Algorithm 8.3 also makes use of a `ChordList`. For each vertex v in *cycle_trace* Algorithm 8.3 iterates through the adjacencies of v, and makes a call to `SitsOnNewSide`,

so that altogether in it runs in $\mathcal{O}(n^3)$ time.

Algorithm 8.3. *Is Essential*

INPUT:

- a `ChordList` *cycle_trace* that corresponds to edges that form a cycle of this `AugRotSystem`,
- a boolean array *visited* of size n for vertices used in the cycle.

OUTPUT:

- returns `true` if the cycle input is essential and `false` otherwise.

```
IsEssential( cycle_trace, visited ) {
1 ) int num_sign = the number of -1 signature edges in cycle_trace;
2 ) if ( (num_sign % 2) == 1 ) {
3 )   // the cycle is nonorientable
4 )   return true;
5 ) }
6 ) // cut along the oriented cyclic walk of this AugRotSystem
7 ) // to create a new embedding H with new vertices for each
    vertex visited in cycle_trace
8 ) AugRotSystem H = a copy of this AugRotSystem;
9 ) n_H = n + cycle_trace.size;
10 ) new_labels = integer array of new vertex labels from n to
    n + cycle_trace.size - 1 in order of the vertices visited in
    cycle_trace;
11 ) // keep track of the local orientation direction as we
    iterate through the oriented cyclic walk
12 ) dir = +1;
13 ) ChordNode chord_ptr = cycle_trace.path_start;
14 ) for (i = 0; i < cycle_trace.size; i++) {
15 )   t = (i == 0) ? cycle_trace.path_rear.u : chord_ptr.prev.u;
16 )   u = chord_ptr.u;
17 )   v = chord_ptr.v;
18 )   // copies of t, u, v
19 )   x = new_labels[t];
20 )   y = new_labels[u];
```

```
21 )    z = new_labels[v];
22 )    L = the compound angle of V[u] from t to v;
23 )    R = the compound angle of V[u] from v to t;
24 )    if ( dir == +1 ) {
25 )      remove L − {t, v} from H[u];
26 )      insert x(L − {t, v})z into H[y];
27 )    } else {
28 )      remove R − {t, v} from H[u];
29 )      insert z(R − {t, v})x into H[y];
30 )    }
31 )    // iterate through H[y] − {x, z}
32 )    AdjNode ptr = H.V[y].next;
33 )    for (j = 1; j < H.degree[y] − 1; j++) {
34 )      if (visited[ptr.u]) {
35 )        if ( SitsOnNewSide( cycle_trace, new_label, ptr, u )
           ) {
36 )          ptr.u = new_label[ptr.u];
37 )        }
38 )      }
39 )      ptr = ptr.next;
40 )    }
41 )    // iterate through H[u] − {t, v}
42 )    ptr = H.V[u].next;
43 )    for (j = 1; j < H.degree[u] − 1; j++) {
44 )      if (visited[ptr.u]) {
45 )        if ( SitsOnNewSide( cycle_trace, new_label, ptr, u )
           ) {
46 )          ptr.u = new_label[ptr.u];
47 )        }
48 )      }
49 )      ptr = ptr.next;
50 )    }
51 )    if ( signature of edge uv is negative ) {
52 )      dir = dir ∗ (−1);
53 )    }
54 )  chord_ptr = chord_ptr.next;
55 )  } // end of for-loop on i
```

```
56 ) // embedding H is now a copy of this AugRotSystem cut along
     the oriented cyclic walk cycle_trace
57 ) sum_g = the sum of the Euler genus of each component of H̃;
58 ) if (g == sum_g) {
59 )   return false;
60 ) }
61 ) return true;
}
```

Chapter 9

Open Problems

This research exposes a plethora of new ideas for future work. The following is a list of open questions:

1. What is the complete set of topological obstructions of the torus?

2. All of the projective-planar obstructions of the torus have been found [Juv95]. What are the obstructions of the torus that also embed on the Klein bottle?

3. The Successive Surface Scaffolding Conjectures in both the strong and the weak forms have yet to be proved or disproved. What approaches could be used to prove it? What approaches could be used to disprove it?

4. Which projective-plane obstructions are contained in irreducible triangulations of the torus? (Mohar)

5. Does there exist a graph such that the deletion of any edge lowers the nonorientable genus by two? [Arc95]

6. We proved that the minor obstructions of the torus that are not double-wye-delta can be embedded in $\mathbb{N}_3$. Can all of the topological or double-wye-delta obstructions of the torus be embedded in $\mathbb{N}_3$?

7. Is there a practical polynomial-time algorithm for embedding on the torus?

8. Is there an efficient algorithm for taking a signed combinatorial embedding as input which outputs an equivalent signed embedding with a minimum number of -1 signature edges?

9. Is there an algorithm taking a depiction of an embedding as input that could output other depictions, perhaps more aesthetic given some criteria? For example, an output depiction with a minimum number of edges using features? Or any edge uses at most some minimum number of features?

10. Is there an algorithm for changing a depiction with two crosscaps into a depiction with one twisted handle, and vice versa? An algorithm for changing a depiction with one crosscap and one handle into a depiction with three crosscaps, and vice versa? An algorithm for changing a depiction with one crosscap and a twisted handle to one crosscap and a handle, and vice versa?

11. Is there a way to get a depiction to show the symmetries of an embedding?

12. Is there an efficient algorithm to change a depiction that simply moves a handle or a crosscap?

13. Consider an embedding of an obstruction G of the surface $\mathbb{S}$. Is it always possible to triangulate any embedding of G on the surface obtained from $\mathbb{S}$ with a crosscap added by inserting edges and vertices? Which such G and embeddings of G allow for a triangulation that is irreducible? Could properties be characterized for when this is possible? If such a triangulation of G is possible, what is the smallest number of vertices necessary to add to G in order to get an irreducible triangulation?

14. For an obstruction G of a surface $\mathbb{S}$ that is contained in an irreducible triangulation T of the surface $\mathbb{S}$ with a crosscap added, what properties does the graph $T - E(G)$ have? Is there a polynomial-time algorithm to decide which edges of an irreducible triangulation must be removed to obtain an obstruction?

15. Can the ideas behind the Successive Surface Scaffolding Conjecture be used to form more conjectures within the context of other graph minor problems?

Appendix A

Projective-Planar Obstruction Names

The 103 projective-plane obstructions numbered as in Tables 7.7–7.12 matched with their names as appearing in Appendix A of Graphs on Surfaces ([MT01]):

0 . . . . . B_1	21 . . . . . C_3	42 . . . . . F_4	63 . . . . . B_{10}	84 . . . . . E_{31}
1 . . . . . A_2	22 . . . . . D_6	43 . . . . . D_1	64 . . . . . A_5	85 . . . . . D_{19}
2 . . . . . E_3	23 . . . . . D_4	44 . . . . . E_{28}	65 . . . . . E_{13}	86 . . . . . C_{11}
3 . . . . . B_2	24 . . . . . E_{21}	45 . . . . . E_{23}	66 . . . . . E_{15}	87 . . . . . C_9
4 . . . . . E_{18}	25 . . . . . B_9	46 . . . . . E_{11}	67 . . . . . E_{14}	88 . . . . . A_4
5 . . . . . D_3	26 . . . . . B_8	47 . . . . . E_{26}	68 . . . . . E_2	89 . . . . . E_{36}
6 . . . . . B_3	27 . . . . . B_6	48 . . . . . F_3	69 . . . . . E_1	90 . . . . . E_{17}
7 . . . . . B_7	28 . . . . . C_2	49 . . . . . D_{14}	70 . . . . . F_{10}	91 . . . . . E_{16}
8 . . . . . B_5	29 . . . . . D_8	50 . . . . . E_{25}	71 . . . . . F_9	92 . . . . . E_{38}
9 . . . . . C_7	30 . . . . . E_{20}	51 . . . . . E_{24}	72 . . . . . F_8	93 . . . . . E_{37}
10 . . . . . B_4	31 . . . . . A_1	52 . . . . . D_{15}	73 . . . . . F_7	94 . . . . . E_{39}
11 . . . . . D_{17}	32 . . . . . D_9	53 . . . . . E_{27}	74 . . . . . C_8	95 . . . . . C_{10}
12 . . . . . E_4	33 . . . . . E_6	54 . . . . . D_{10}	75 . . . . . E_{32}	96 . . . . . F_{14}
13 . . . . . E_{22}	34 . . . . . E_{12}	55 . . . . . D_{11}	76 . . . . . D_{16}	97 . . . . . F_{13}
14 . . . . . D_{12}	35 . . . . . E_{10}	56 . . . . . F_6	77 . . . . . E_{35}	98 . . . . . F_{12}
15 . . . . . D_5	36 . . . . . D_2	57 . . . . . C_5	78 . . . . . E_{33}	99 . . . . . F_{11}
16 . . . . . F_1	37 . . . . . C_1	58 . . . . . F_5	79 . . . . . E_{34}	100 . . . . E_{42}
17 . . . . . E_5	38 . . . . . F_2	59 . . . . . E_7	80 . . . . . C_6	101 . . . . E_{41}
18 . . . . . C_4	39 . . . . . E_8	60 . . . . . G	81 . . . . . E_{30}	102 . . . . E_{40}
19 . . . . . E_{19}	40 . . . . . E_9	61 . . . . . A_3	82 . . . . . D_{18}	
20 . . . . . D_7	41 . . . . . D_{13}	62 . . . . . B_{11}	83 . . . . . E_{29}	

Appendix B

Klein-Bottle Irreducible-Triangulation Names

The 29 irreducible triangulations of the Klein bottle numbered as in Tables 7.7–7.12 matched with their names as appearing in the work of Lawrencenko and Negami ([LN97]), and Sulanke ([Sul06c]):

$1 \ldots\ldots Kh3$	$7 \ldots. Kh18$	$13 \ldots Kh17$	$19 \ldots Kh12$	$25 \ldots Kh16$
$2 \ldots\ldots Kh2$	$8 \ldots\ldots Kh7$	$14 \ldots Kh21$	$20 \ldots\ldots Kc1$	$26 \ldots\ldots Kc2$
$3 \ldots\ldots Kh5$	$9 \ldots\ldots Kh9$	$15 \ldots Kh19$	$21 \ldots Kh10$	$27 \ldots Kh25$
$4 \ldots\ldots Kh1$	$10 \ldots. Kh8$	$16 \ldots Kh15$	$22 \ldots Kh11$	$28 \ldots\ldots Kc4$
$5 \ldots\ldots Kh6$	$11 \ldots Kh22$	$17 \ldots Kh23$	$23 \ldots Kh13$	$29 \ldots\ldots Kc3$
$6 \ldots\ldots Kh4$	$12 \ldots Kh24$	$18 \ldots Kh14$	$24 \ldots Kh20$	

Bibliography

[AH89] Dan Archdeacon and Phil Huneke. A Kuratowski theorem for nonorientable surfaces. *Journal of Combinatorial Theory, Series B*, 46(2):173–231, 1989.

[AP61] Louis Auslander and Seymour V. Parter. On embedding graphs into the plane. *Journal of Mathematics and Mechanics*, 10:517–523, 1961.

[Arc81] Dan Archdeacon. A Kuratowski theorem for the projective plane. *Journal of Graph Theory*, 5(3):243–246, 1981.

[Arc95] Dan Archdeacon. The effect of edge deletion on the nonorientable genus. `http://www.cems.uvm.edu/TopologicalGraphTheoryProblems/decgenus.htm`, 1995. Accessed: 28 May 2017.

[Arc96] Dan Archdeacon. Topological graph theory. *A survey. Congressus Numerantium*, 115(5-54):18, 1996.

[Arm13] Mark Anthony Armstrong. *Basic Topology*. Springer Science & Business Media, 2013.

[Bad64] Wilhelm Bader. Das topologische problem der gedruckten schaltung und seine lösung. *Archiv für Elektrotechnik*, 49:2–12, 1964.

[Bar82] David Barnette. Generating the triangulations of the projective plane. *Journal of Combinatorial Theory, Series B*, 33(3):222–230, 1982.

[BE88] David Barnette and Allan Edelson. All orientable 2-manifolds have finitely many minimal triangulations. *Israel Journal of Mathematics*, 62(1):90–98, 1988.

[BE89] David Barnette and Allan Edelson. All 2-manifolds have finitely many minimal triangulations. *Israel Journal of Mathematics*, 67(1):123–128, 1989.

[BHK62] Joseph Battle, Frank Harary, and Yukihiro Kodama. Additivity of the genus of a graph. *Bulletin (New Series) of the American Mathematical Society*, 68(6):565–568, 1962.

[BL76] Kellogg S. Booth and George S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using *pq*-tree algorithms. *Journal of Computer and System Sciences*, 13(3):335–379, 1976.

[BM04] John M Boyer and Wendy J Myrvold. On the cutting edge: Simplified $\mathcal{O}(n)$ planarity by edge addition. *J. Graph Algorithms Appl.*, 8(2):241–273, 2004.

[Bra09] Ulrik Brandes. The left-right planarity test. *Manuscript submitted for publication*, 2009.

[BW89a] Rainer Bodendiek and Klaus Wagner. The fascination of minimal graphs. In G. Dirac and L.D. Andersen, editors, *Graph Theory in Memory of G.A. Dirac*, Annals of Discrete Mathematics, pages 39–51. North-Holland, 1989.

[BW89b] Rainer Bodendiek and Klaus Wagner. Solution to König's graph embedding problem. *Mathematische Nachrichten*, 140(1):251–272, 1989.

[Cha02] John Chambers. Hunting for torus obstructions. Master's thesis, Department of Computer Science, University of Victoria, Victoria, BC, 2002.

[CNAO85] Norishige Chiba, Takao Nishizeki, Shigenobu Abe, and Takao Ozawa. A linear algorithm for embedding planar graphs using¡ i¿ pq¡/i¿-trees. *Journal of Computer and System Sciences*, 30(1):54–76, 1985.

[dF08] Hubert de Fraysseix. Trémaux trees and planarity. *Electronic Notes in Discrete Mathematics*, 31:169–180, 2008.

[dFdM02] Hubert de Fraysseix and Patrice Ossona de Mendez. PIGALE-public implementation of a graph algorithm library and editor. *SourceForge project page http://sourceforge.net/projects/pigale*, 2002.

[dFdMR06] Hubert de Fraysseix, Patrice Ossona de Mendez, and Pierre Rosenstiehl. Trémaux trees and planarity. *International Journal of Foundations of Computer Science*, 17(05):1017–1029, 2006.

[DMP64] G. Demoucron, Y. Malgrange, and R. Pertuiset. Graphes planaires. *Rev. Fran caise Recherche Op erationnelle*, 8:33–47, 1964.

[DR91] Hristo Djidjev and John Reif. An efficient algorithm for the genus problem with explicit construction of forbidden subgraphs. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 337–347. ACM, 1991.

[dV12] Eric C. de Verdiere. Topological algorithms for graphs on surfaces. 2012.

[FHRR95] J.R. Fiedler, John Philip Huneke, R. Bruce Richter, and Neil Robertson. Computing the orientable genus of projective graphs. *Journal of Graph Theory*, 20(3):297–308, 1995.

[Fil78] I.S. Filotti. An efficient algorithm for determining whether a cubic graph is toroidal. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 133–142. ACM, 1978.

[Fil80] I.S. Filotti. An algorithm for imbedding cubic graphs in the torus. *Journal of Computer and System Sciences*, 20(2):255–276, 1980.

[Flö10] Anna Flötotto. *Embeddability of graphs into the Klein surface*. PhD thesis, Bielefeld (Germany): Bielefeld University, 2010.

[FMR79] I.S. Filotti, Gary L. Miller, and John Reif. On determining the genus of a graph in $O(v^{O(g)})$ steps (preliminary report). In *Proceedings of the eleventh annual ACM symposium on Theory of computing*, pages 27–37. ACM, 1979.

[GHW79] Henry H Glover, John P Huneke, and Chin San Wang. 103 graphs that are irreducible for the projective plane. *Journal of Combinatorial Theory, Series B*, 27(3):332–370, 1979.

[GK02] Andrei Gagarin and William Kocay. Embedding graphs containing k_5-subdivisions. *Ars Comb.*, 64:33, 2002.

[GKN03] Andrei Gagarin, William Kocay, and Daniel Neilson. Embeddings of small graphs on the torus. *Cubo*, 5:171–251, 2003.

[GMC09] Andrei Gagarin, Wendy Myrvold, and John Chambers. The obstructions for toroidal graphs with no $K_{3,3}$'s. *Discrete Mathematics*, 309(11):3625–3631, 2009.

[Gol63] A. J. Goldstein. An efficient and constructive algorithm for testing whether a graph can be embedded in a plane. In *Graph and Combinatorics Conference, Contract No. NONR 1858-(21), Office of Naval Research Logistics Proj., Dept. of Mathematics, Princeton University, May 16-18*, 1963.

[GT87] Jonathan L. Gross and Thomas W. Tucker. *Topological Graph Theory*. Courier Corporation, 1987.

[HT74] John Hopcroft and Robert Tarjan. Efficient planarity testing. *Journal of the ACM*, 21(4):549–568, 1974.

[JMM95] Martin Juvan, Jože Marinček, and Bojan Mohar. Embedding graphs in the torus in linear time. In Egon Balas and Jens Clausen, editors, *Integer Programming and Combinatorial Optimization*, volume 920 of *Lecture Notes in Computer Science*, pages 360–363. Springer Berlin / Heidelberg, 1995.

[Juv95] Martin Juvan. Algorithms and obstructions for embedding graphs in the torus. *Slovene, University of Ljubljana, Ph. D. Thesis*, 1995.

[JW10] Gwenaël Joret and David R. Wood. Irreducible triangulations are small. *Journal of Combinatorial Theory, Series B*, 100(5):446–455, 2010.

[Klo89] W. Klotz. A constructive proof of kuratowski?s theorem. *Ars Combinatoria*, 28:51–54, 1989.

[Kos80] Czes Kosniowski. *A first course in algebraic topology*. CUP Archive, 1980.

[Kur30] K. Kuratowski. Sur le probléme des courbes gauches en topologie. *Fundamenta Mathematicae*, 15:271–283, 1930.

[Law87] S. Lawrencenko. Irreducible triangulations of the torus. *Ukrain. Geom. Sb*, 30:52–62, 1987.

[LEC66] Abraham Lempel, Shimon Even, and Israel Cederbaum. An algorithm for planarity testing of graphs. *Theory of graphs*, 8(2):215–232, 1966.

[LN97] Serge Lawrencenko and Seiya Negami. Irreducible triangulations of the Klein bottle. *Journal of Combinatorial Theory, Series B*, 70(2):265–291, 1997.

[MK11] Wendy Myrvold and William Kocay. Errors in graph embedding algorithms. *Journal of Computer and System Sciences*, 77(2):430–438, 2011.

[MM96] K. Mehlhorn and P. Mutzel. On the embedding phase of the hopcroft and tarjan planarity testing algorithm. *Algorithmica*, 16(2):233–242, 1996.

[Moh93] Bojan Mohar. Projective planarity in linear time. *J. Algorithms*, 15(3):482–502, 1993.

[Moh96] Bojan Mohar. Embedding graphs in an arbitrary surface in linear time. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 392–397. ACM, 1996.

[Moh99] Bojan Mohar. A linear time algorithm for embedding graphs in an arbitrary surface. *SIAM Journal on Discrete Mathematics*, 12(1):6–26, 1999.

[MP14] Brendan D McKay and Adolfo Piperno. Practical graph isomorphism, ii. *Journal of Symbolic Computation*, 60:94–112, 2014.

[MT01] Bojan Mohar and Carsten Thomassen. *Graphs on Surfaces*. The John Hopkins University Press, 2715 North Charles Street, Baltimore, Maryland 21218-4363, first printing edition, 2001.

[Neu94] Eugene T. Neufeld. *Practical toroidality testing*. PhD thesis, University of Victoria, 1994.

[NM97] Eugene Neufeld and Wendy Myrvold. Practical toroidality testing. In *Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms*, SODA '97, pages 574–580, Philadelphia, PA, USA, 1997. Society for Industrial and Applied Mathematics.

[PD85] B. Perunicic and Z. Duric. An efficient algorithm for embedding graphs in the projective plane. In *Graph theory with applications to algorithms and computer science*, pages 637–650. John Wiley & Sons, Inc., 1985.

[Ran97] Scott P. Randby. Minimal embeddings in the projective plane. *Journal of Graph Theory*, 25(2):153–163, 1997.

[RM05] Jianping Roth and Wendy Myrvold. Simpler projective plane embedding. *Ars Combinatoria*, 75:135–156, 2005.

[RS90] Neil Robertson and P.D Seymour. Graph minors. viii. a Kuratowski theorem for general surfaces. *Journal of Combinatorial Theory, Series B*, 48(2):255–288, 1990.

[Sch13] Jens M. Schmidt. A planarity test via construction sequences. In *Mathematical Foundations of Computer Science 2013*, pages 765–776. Springer, 2013.

[Sko12] Petr Skoda. *Obstructions for embedding graphs into surfaces*. PhD thesis, Simon Fraser University, 2012.

[Sul06a] Thom Sulanke. Generating triangulations of surfaces. `http://hep.physics.indiana.edu/~tsulanke/graphs/surftri/index.html`, 2006. Accessed: 6 May 2015.

[Sul06b] Thom Sulanke. Irreducible triangulations of low genus surfaces. *Arxiv preprint math/0606690*, 2006.

[Sul06c] Thom Sulanke. Note on the irreducible triangulations of the klein bottle. *Journal of Combinatorial Theory, Series B*, 96(6):964 – 972, 2006.

[UAH74] Jeffrey D. Ullman, Alfred V. Aho, and John E. Hopcroft. *The design and analysis of computer algorithms*, volume 4. Addison-Wesley, Reading, 1974.

[Wag37] Klaus Wagner. Über eine eigenschaft der ebenen komplexe. *Mathematische Annalen*, 114(1):570–590, 1937.

[Woo06] Jennifer R. Woodcock. *A faster algorithm for torus embedding.* PhD thesis, University of Victoria, 2006.

[Yu14] Jiahua Yu. *A Practical Torus Embedding Algorithm and Its Implementation.* PhD thesis, Applied Sciences: School of Computing Science, 2014.