Hunting for Torus Obstructions

by

John Chambers

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

MASTER OF SCIENCE

in the Department of Computer Science

We accept this thesis as conforming
to the required standard

[redacted]

Dr. Wendy Myrvold, Supervisor (Dept. of Computer Science)

[redacted]

Dr. John Ellis, Departmental Member (Dept. of Computer Science)

[redacted]

Dr. Frank Ruskey, Departmental Member (Dept. of Computer Science)

[redacted]

Dr. Luis Goddyn, External Examiner (Dept. of Mathematics, Simon Fraser University)

Supervisor: Dr. Wendy Myrvold

ABSTRACT

A *torus* is a sphere with one handle. A graph is *toroidal* if it embeds in the torus with no crossing edges. A *topological obstruction* for the torus is a graph G with minimum vertex degree three that is not embeddable in the torus but for all edges e, $G - e$ embeds in the torus. A *minor order obstruction* has the additional property that for all edges e, G contract e embeds in the torus. Two algorithms to find torus obstructions are presented. Using these algorithms, a new lower bound on the number of torus obstructions is established at 239,451 topological obstructions, 16,682 of which are minor order. Also, an alternate approach to finding the torus obstructions is discussed. Implementation of this approach requires knowing the projective planar minor order torus obstructions. An additional contribution of this thesis is the determination of the complete set of 270 projective planar minor order obstructions for the torus.

Examiners:

[redacted]

Dr. Wendy Myrvold, Supervisor (Dept. of Computer Science)

[redacted]

Dr. John Ellis, Departmental Member (Dept. of Computer Science)

[redacted]

Dr. Frank Ruskey, Departmental Member (Dept. of Computer Science)

[redacted]

Dr. Luis Goddyn, External Examiner (Dept. of Mathematics, Simon Fraser University)

Contents

List of Tables

List of Figures

Acknowledgments

Thanks to the author's academic supervisor, Dr. Wendy Myrvold, for her direction and assistance.

Chapter 1

Introduction

A *graph* G consists of a finite set $V(G)$ of *vertices* and a finite set $E(G)$ of *edges*, where each edge in $E(G)$ is associated with an unordered pair (u, v) of elements of $V(G)$. The edge (u, v) has as *endpoints* vertices u and v, and edge e is *incident* to vertex u if u is an endpoint of e. We disallow *multiple edges* (more than one edge with the same endpoints) and *loops* (edges of the form (u, u)). The *size* of G is $|E(G)|$ and the *order* of G is $|V(G)|$. The *degree* of a vertex is the number of edges incident to it. A graph whose vertices are all degree three is called a *cubic* graph. Discussion of elementary graph theoretic concepts such as these and some that follow can be found in the introductory texts by West [26] and Bondy and Murty [7].

Surfaces are uniquely determined by the properties of *genus* and *orientability*. The genus may be any non-negative integer, and if the surface has genus greater than zero, it may be *orientable* or *non-orientable*. Genus may be thought of intuitively as describing the number of handles on the surface, and orientability as describing whether or not the surface has a well-defined sense of clockwise. A more detailed account of genus and orientability, as well as other fundamental topological concepts, can be found in the texts by Henle [13] and Kinsey [15].

The plane, equivalent to the sphere, is the only surface of genus zero, and is

orientable. After the plane, the remaining orientable surfaces are called the *torus* (the one-handled sphere), with genus one, and the *k-handled torus* (k greater than one), with genus k. The *projective plane* is a non-orientable surface of genus one and is equivalent to a disk with antipodal points identified.

A graph is *embeddable* in surface Σ if it can be drawn on Σ with no intersecting edges or overlapping vertices. The *orientable* (*non-orientable*) genus of a graph is equal to the minimum of the genera of the orientable (non-orientable) surfaces in which it embeds. A graph is *planar* if it embeds in the sphere, *toroidal* if it embeds in the torus, and *projective planar* if it embeds in the projective plane. A graph of orientable genus greater than one will here be called *non-toroidal.*

A *topological obstruction* for surface Σ is a graph G with minimum vertex degree three that is not embeddable in Σ but for all edges e, $G-e$ embeds in Σ. To *contract* edge (u, v) of graph G, remove edge (u, v), identify u and v, and replace multiple edges with a single edge. The graph created by contracting edge e of G is denoted by $G \circ e$. A *minor order obstruction* for surface Σ is a graph G that is a topological obstruction and, for all edges e, $G \circ e$ embeds in Σ. Reference to obstructions will imply topological obstructions, unless otherwise specified. The restriction on the degree of the vertices of an obstruction G follows from the observation that neither removing a vertex of degree two and replacing it with an edge nor removing a vertex of degree zero or one changes the genus of G.

As of this writing, complete sets of obstructions are known for only the plane and the projective plane. Kuratowski showed that K_5 and $K_{3,3}$ are the only topological obstructions for the plane, and Wagner showed them to be the only minor order ones [16, cited in [3]]. Glover, Huneke, and Wang discovered 103 topological projective plane obstructions, 35 of which are minor order [12]. Archdeacon proved that this set was complete [2].

The generalized Kuratowski theorem states that for any surface there exists a finite set of obstructions which characterize embeddability in that surface. Bodendiek and Wagner proved this result for orientable surfaces [6] and Archdeacon

and Huneke for non-orientable surfaces [4]. A more general result was proved by Robertson and Seymour which implies both the orientable and non-orientable cases [24].

Neufeld and Myrvold claimed to find a complete set of torus obstructions with up to ten vertices and a partial set of larger obstructions [21, 22]. In total they found 3884 2-connected topological obstructions, 2249 of which are minor order. Cattell, Dinneen, Downey, Fellows, and Langston conjecture that there are about 2000 minor order torus obstructions [8]. Archdeacon reports that Glover and Huneke conjecture that there are at least 10,000 minor order torus obstructions [3].

The primary goal of this research is to extend the work of Neufeld and Myrvold in their efforts to find a complete set of torus obstructions. An important tool in both their work and ours is the torus-embedding algorithm they constructed [21, 22]. Although the algorithm is exponential in the worst case, various optimizations have been added that increase its speed at least threefold.

Practical alternatives to the Neufeld-Myrvold torus embedding algorithm are unavailable. Filotti's algorithm, which embeds only cubic graphs in the torus, is too specialized for our purposes [10]. Mohar's linear-time embedding algorithm for arbitrary surfaces [19] has proven to be too complicated to implement. Juvan and Mohar have presented an algorithm for torus embedding that they claim is easier to implement than the linear version but that has cubic complexity [14]. As of this writing, it has not been successfully implemented.

In this thesis, two algorithms for finding torus obstructions are presented. Both algorithms use the same decision routine for determining whether a graph is an obstruction, and it will be described in Chapter 2. In Chapter 3, the two algorithms and the central results are presented and discussed. In Chapter 4 is presented the results of a project to generate a complete set of cubic torus obstructions. A strategy proposed by Archdeacon for finding all the torus obstructions is outlined in Chapter 5. Using results established by Randby [23] and Fiedler, Huneke, Richter, and Robertson [9], we show how we have taken the first step in Archdeacon's two-step

proposal, that of finding all the projective planar minor order torus obstructions. Chapter 6 concludes with suggestions for future research.

With the two obstruction finding algorithms, the central result of this thesis was established, namely, the discovery of a large number of hitherto unknown torus obstructions, and therefore a new lower bound on their number. In total, 239,451 topological obstructions were found, 16,682 of which were minor order. Size and order data for these results are presented in Tables 3.4 and 3.5.

Chapter 2

The Obstruction_Test algorithm

The Obstruction_Test algorithm, used to determine whether a graph is a torus obstruction, is described in Section 2.1. Section 2.2 explains the type and number of machines that were used throughout this project.

2.1 The Obstruction_Test algorithm

Central to our research is the torus embedding algorithm developed by Neufeld and Myrvold [21, 22]. Given a graph G, the algorithm determines whether there is a torus embedding of G, returning the embedding if it exists. Their algorithm can be described as a torus embedder combined with a preprocessor that quickly eliminates certain non-toroidal graphs before the exponential embedding process. The algorithm used for this research, called Torus_Test, is a modification of just the embedder part of their algorithm. The preprocessor part was not used. This is important to note, as the number of obstructions reported found using their algorithm differs from the number found using Torus_Test (see Section 3.1). The Torus_Test(G) routine returns the orientable genus of input graph G if it is one or less, two if the genus of G is greater than one.

The decision algorithm, Obstruction_Test(G), is built around Torus_Test and it

determines whether input graph G is a torus obstruction. Pseudocode for Obstruction_Test is shown in Figure 2.1. The design of Obstruction_Test follows naturally from the definitions of topological and minor order obstructions given in Chapter 1. If G is toroidal or has a vertex of degree less than three, Obstruction_Test returns zero; if G is a topological torus obstruction, it returns one; if G is a minor order torus obstruction, it returns two; and if G is non-toroidal and not an obstruction, it returns three.

```
function Obstruction_Test( G : graph )
begin
    if G has any vertices of degree at most two then return(0);
    if Torus_Test( G ) < 2 then return(0);
    for every edge e of G do
        if Torus_Test( G − e ) > 1 then return(3);
    for every edge e of G do
        if Torus_Test( G ∘ e ) > 1 then return(1);
    return(2);
end {Obstruction_Test}
```

Figure 2.1: The Obstruction_Test algorithm.

2.2 Hardware note

Almost all computation supporting our research was conducted on 41 700MHz PentiumIII-based personal computers, all using Sun's Solaris 8. Several 400MHz Pentium IIs were used as well, also using Solaris 8. Because of the large amount of computation involved in our research, work was distributed over these machines whenever possible.

Chapter 3

The Naive algorithm and the Split-Delete algorithm

In this chapter, both the Naive and Split-Delete algorithms are described. The Naive routine is a brute force approach to finding torus obstructions that is guaranteed to find every obstruction of a specified size and order. The completeness of its results is offset by the time cost it incurs. The Split-Delete algorithm does not necessarily generate all the torus obstructions of a certain size and order. However, it finds a significant number of obstructions, and it does so much more efficiently than Naive, especially as graph order increases. Most of the obstructions collected during this research resulted from combining these two approaches.

The Naive algorithm is discussed in Section 3.1, followed by a discussion of the Split-Delete algorithm in Section 3.2. Two applications of the algorithms are presented in Section 3.3, one using just Split-Delete and the other a combination of the two. Section 3.4 concludes with a description of the results of the applications.

3.1 The Naive algorithm

The Naive algorithm, shown in Figure 3.1, considers all the connected graphs of order n within a specific size range and retains those that are torus obstructions. The algorithm takes advantage of the following basic properties of torus obstructions to mitigate its time cost. Since K_7 is toroidal [21], any obstruction to toroidality has at least eight vertices. There are only three eight-vertex obstructions, all minor order [22]. Further, the size of any obstruction does not exceed $3n+1$. This follows from the fact that a toroidal graph can have no more than $3n$ edges (see Gibbons in [11]). It is also the case that the size of an obstruction will not be smaller than $\lceil 3n/2 \rceil$, otherwise the graph will have a vertex of degree less than three, thereby violating the degree restriction on obstructions. The Naive algorithm finds all the connected obstructions of order n by applying Obstruction_Test to all the graphs that fall between sizes $\lceil 3n/2 \rceil$ and $3n+1$ inclusive.

To generate the graphs to be tested by Obstruction_Test, Brendan McKay's algorithm geng is used [18]. The geng algorithm has various options which allow one to specify, among other things, the order of the graphs to be generated, whether to generate connected graphs, minimum and maximum vertex degrees, and minimum and maximum size. A typical invocation of geng follows:

$$\text{geng} \quad -cd3D10 \quad 11 \quad 24:25$$

where -c indicates that the generated graphs are connected, d3 and D10 indicate that no graph with a vertex of degree less than three or greater than ten is generated, 11 specifies the order of the output graphs, and 24:25 indicates that graphs of size no smaller than 24 and no larger than 25 are produced. A helpful option not shown allowed for graphs satisfying certain parameters to be split into k roughly equal subsets, where k is a parameter specified by the user and the subsets are numbered 0 to k-1.

Using Naive, the complete set of connected obstructions of order less than twelve was determined. There are only two disconnected obstructions in this range, one

```
function Naive( n : integer; Result : array )
local H : graph; ob_code : integer;
begin
    while geng is not finished do
        Get next graph H of order n;
        ob_code = Obstruction_Test( H );
        if ob_code = 1 or 2 then put H in Result;
    end while
end {Naive}
```

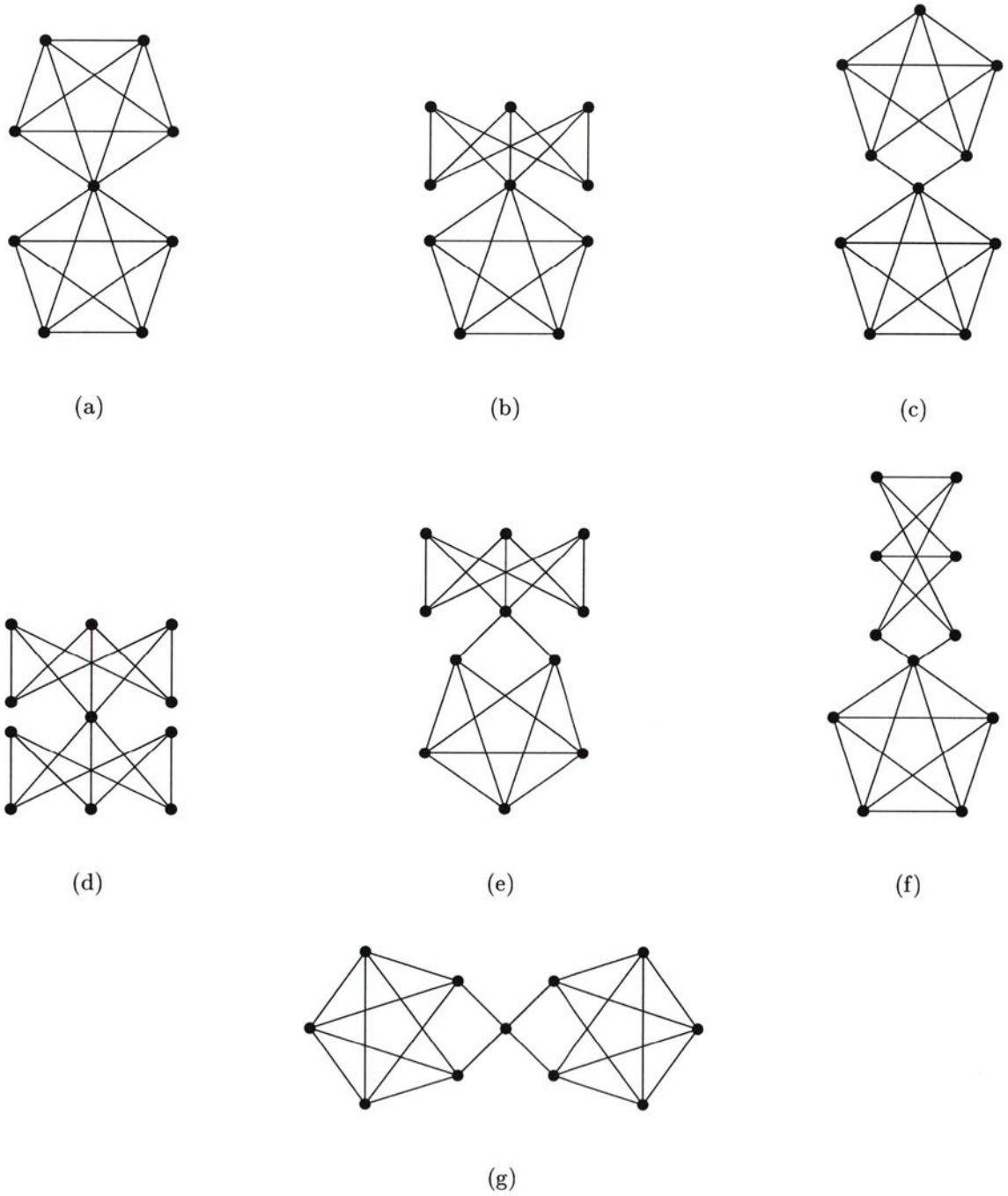
Figure 3.1: The Naive algorithm.

consisting of two K_5's and the other of a K_5 and a $K_{3,3}$. They were added independently. The one-connected obstructions of order less than twelve are shown in Figure 3.2. They consist of a nine-vertex minor order obstruction (Figure 3.2(a)), a ten-vertex minor order obstruction (Figure 3.2(b)), a ten-vertex topological obstruction (Figure 3.2(c)), an eleven-vertex minor order obstruction (Figure 3.2(d)), and three eleven-vertex topological obstructions (Figures 3.2(e), 3.2(f) and 3.2(g)). The one-connected obstructions greater than eleven, shown in Figure 3.3, consist of two order twelve topological obstructions (Figures 3.3(a) and 3.3(b)) and one order thirteen topological obstruction (Figure 3.3(c)). That the ten graphs shown in Figures 3.2 and 3.3 constitute the complete set of the one-connected torus obstructions is shown by Theorem 3.4.5 in Section 3.4 below.

We found that our number of biconnected obstructions agrees with the number established by Neufeld and Myrvold [22], except in the case of order ten, where they missed a 26-edge obstruction. This graph was first found by Skala in his work on torus embedding [25], and subsequently by Naive. Table 3.1 displays size and order statistics for the complete set of torus obstructions, both connected and disconnected, of order eleven and less.

	$n = 8$	9	10	11
$m = 18$				5
19		2	15	2
20		5	9	49
21		2	35	87
22	1	9	40	270
23		17	190	892
24	1	6	170	1878
25	1	2	102	1092
26		5	76	501
27			21	120
28			1	22
29				4
30			1	1
31				
32				
total	3	48	660	4923

Table 3.1: Size and order data for Naive results.

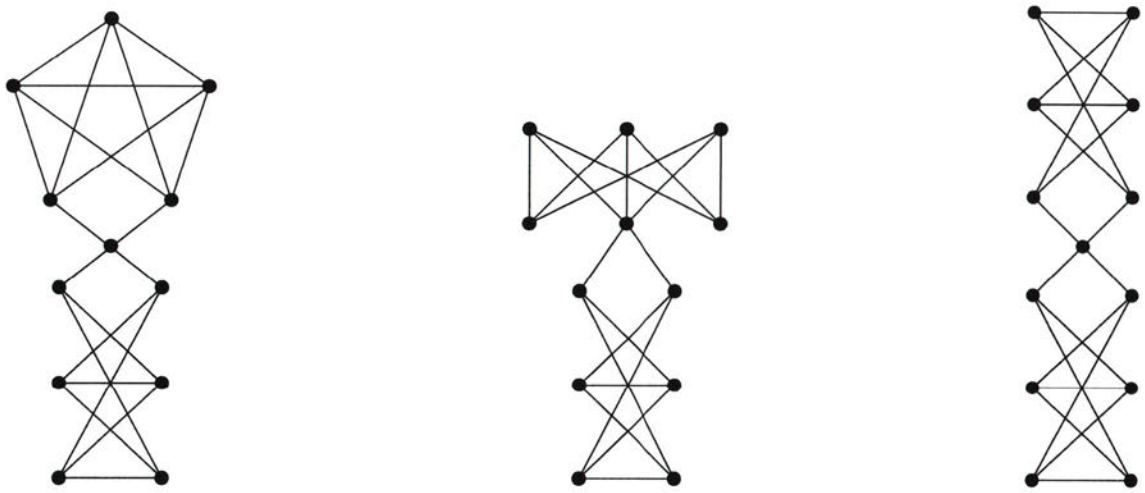


(a) (b) (c)

(d) (e) (f)

(g)

Figure 3.2: One-connected obstructions of order less than twelve found by Naive.



(a) (b) (c)

Figure 3.3: One-connected obstructions of order greater than eleven.

3.2 The Split-Delete algorithm

Although Naive is guaranteed to find all the obstructions of a particular size and order, it is impractical to use extensively because of its time cost. In this section, we turn to a more efficient approach, namely the Split-Delete algorithm. This approach to finding obstructions was suggested by Glover, Huneke, and Wang [12], who used it to derive all the projective plane obstructions from a set of just five projective plane obstructions.

Vertices u and v of graph G are *adjacent* if there is an edge (u, v) in G. Let v be a vertex of graph G and let $N(v)$ denote the set of vertices adjacent to v. Also, let (A, B) be a *bipartition* of $N(v)$, where a bipartition of $N(v)$ is an ordered pair of disjoint sets whose union is $N(v)$. The function Splitter, shown in Figure 3.4, copies graph G to graph H, then *splits* vertex v of H according to a bipartition (A, B).

To ensure that the splitting operation does not result in a graph with a vertex of degree less than three, only *admissible* bipartitions of $N(v)$ are used, where an

```
function Splitter( G, H : graph; v : integer; A, B : array )
begin
    Copy G to H;
    Delete v and its incident edges from H;
    Add vertices v1 and v2 to H;
    Add edge (v1, v2) to H;
    for all u in A do
        add edge (u, v1) to H;
    for all u in B do
        add edge (u, v2) to H;
end {Splitter}
```

Figure 3.4: The algorithm to split a vertex using a bipartition.

admissible bipartition (A, B) of $N(v)$ is such that $|A|$ and $|B|$ are each greater than one. The function Split, shown in Figure 3.5, splits the vertices of graph G in all admissible ways and collects the split graphs in array *Result*. Brendan McKay's canonical labelling utility, nauty [17], allows us to put all split graphs in canonical form and, using an array, collect a set of graphs containing no isomorphs, where graphs G_1 and G_2 are *isomorphic* if there is a bijection ϕ from the vertices of G_1 to the vertices of G_2 such that (u, v) is an edge in G_1 if and only if $(\phi(u), \phi(v))$ is an edge in G_2.

The Delete algorithm, shown in Figure 3.6, recursively generates all the spanning subgraphs of the order n input graph G that are non-toroidal and collects those that are torus obstructions in array *Result*. During generation, isomorphic graphs are produced. To prevent redundantly processing these, each generated graph is put in canonical form using nauty and inserted into an array if it is not already there. If the graph is in the array, its processing is terminated.

Edge (a, b) is *lexicographically larger* than edge (c, d) if a is less than b, c is less

```
function Split( G : graph; Result : array )
local H, H_can : graph;
begin
    for every vertex v of G do begin
        for every admissible partition (A, B) of N(v) do begin
            Splitter( G, H, v, A, B );
            Put H in canonical form, H_can, using nauty;
            if H_can is not in Result then add to Result;
        end for
    end for
end {Split}
```

Figure 3.5: The algorithm to split all vertices of a graph.

than d, and either a is greater than c or both a is equal to c and b is greater than d. Parameters u and v of Delete represent the endpoints of an edge (u, v). Delete uses these parameters to constrain edge removal to a lexicographical order: only edges that are lexicographically larger than (u, v) are removed at deeper levels of the recursion. This prevents redundant computations.

The Split_Delete routine, which combines both Split and Delete, is shown in Figure 3.7. The input to Split_Delete is a graph G and its order n. The obstructions generated are collected in array *Result*.

3.3 Application of Split_Delete and Naive

A reason for the effectiveness of the Split_Delete algorithm as a tool for finding torus obstructions is provided by the following obvious result:

Theorem 3.3.1 *Let graph H be obtained from graph G by splitting a vertex v of G. If G does not embed in surface Σ then neither does H.*

```
function Delete( n, u, v : integers; G : graph; Result : array )
local G_can : graph; i, j, ob_code : integers; A : array;
begin
    Put G in canonical form, G_can;
    if G_can is in A then return;
    else add G_can to A;
    ob_code = Obstruction_Test( G );        /* Obstruction_Test( G ) is */
    if ob_code == 0 then return;            /* defined in Figure 2.1 */
    if ob_code == 1 or 2 then begin
        Put G_can in Result;
        return;
    end if
    for i = u to n - 1 (where n is the order of G) do begin
        for j = i + 1 to n - 1 do begin
            if i > u or j > v then begin
                if (i, j) ∈ E(G) then Delete( n, i, j, G - (i, j), Result );
            end if
        end for
    end for
end {Delete}
```

Figure 3.6: The algorithm to delete edges from a graph and test for torus obstructions.

```
function Split_Delete( n : integer; G : graph; Result : array )
local K, H : graph; Res1, Res2 : array;
begin
    Empty Res1;
    Split( G, Res1 );
    while Res1 is not empty do begin
        Empty Res2;
        Get next graph H from Res1;
        Delete( n + 1, 0, 0, H, Res2 );
        for every graph K in Res2 do
            if K is not in Result then add to Result;
    end while
end {Split_Delete}
```

Figure 3.7: The Split_Delete algorithm.

Proof: Let e be the edge that is added during the splitting of v. Suppose H embeds in Σ. Contracting e of H gives an embedding of G in Σ, establishing the contrapositive of the desired result. □

According to Theorem 3.3.1, applying the Split algorithm to a non-toroidal graph generates a set P of non-toroidal graphs. A graph G of P either is an obstruction or results in at least one after some combination of edges is removed. The Delete algorithm implements an edge removal process that finds minimum degree three spanning subgraphs of G that are also torus obstructions.

Theorem 3.3.1 suggests that the Split_Delete algorithm can be used for obstruction generation by applying it to obstructions of order n to generate obstructions of order $n+1$. An algorithm that implements this approach, Gen_SD_Set, is shown in Figure 3.8. Input to Gen_SD_Set is an array P of order n torus obstructions. The resulting order $n+1$ obstructions are put in array *Result*. As will be explained shortly, this algorithm is responsible for generating most of our torus obstructions.

```
function Gen_SD_Set( n : integer; P, Result : array )
local K, H : graph; Res : array;
begin
    while P is not empty do begin
        Empty Res;
        Get next graph H from P;
        Split_Delete( n, H, Res );
        for every graph K in Res do
            if K is not in Result then add to Result;
    end while
end {Gen_SD_Set}
```

Figure 3.8: An algorithm for generating a set of order $n+1$ torus obstructions from a set of order n torus obstructions.

The work of Glover, Huneke, and Wang [12] suggests a further application of the Split_Delete algorithm. They observe that there is an order on graphs which can be defined as follows: for graphs G and H, $G <_s H$ if H can be obtained from G by a non-zero sequence of vertex splittings followed by zero or more edge deletions. Such an ordering has a set of minimal elements which are defined as follows. For a set S of graphs, M is the *split-delete minimal* subset of S if

1. for all G in M, there is no H in S such that $H <_s G$, and
2. for all G in $S - M$, there is some H in M such that $H <_s G$

The split-delete minimal subset of the plane obstructions is $\{K_5\}$. Figure 3.9 shows how $K_{3,3}$, the only other plane obstruction, can be obtained from K_5 by splitting one vertex and then removing two appropriately chosen edges.

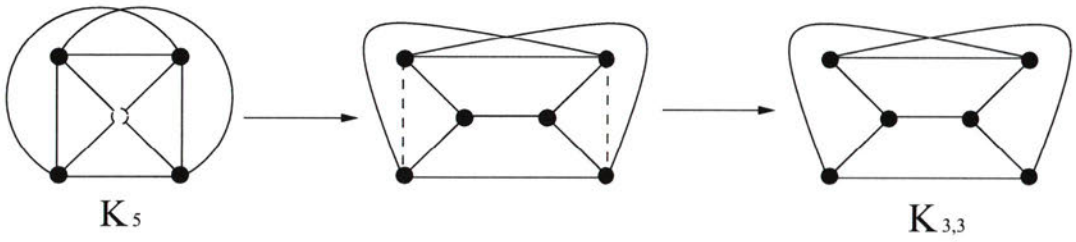


Figure 3.9: Showing how $K_{3,3}$ is derived from K_5 by splitting a vertex and deleting two edges.

Recall that the projective plane has 103 topological obstructions, 35 of which are minor order. Glover, Huneke, and Wang showed that all 103 projective plane obstructions can be derived by performing successive splittings and deletions on just five projective plane obstructions. Their conjecture that this set of five is the split-delete minimal subset of the projective plane obstructions was later proved by Archdeacon [2].

One of the initial goals of this research was to generate the split-delete minimal subset of the torus obstructions. Algorithm Gen_SD_Minset, shown in Figure 3.10, was used for this purpose. The input to Gen_SD_Minset is an array P of the order

$n-1$ torus obstructions; the output is an array M of the split-delete minimal torus obstructions of order less than or equal to max_order. Initially, n is set to eight and P and M are empty.

function Gen_SD_Minset(n, max_order : integer; P,M : array)
local $SDRes$, All : array;
begin
    Gen_SD_Set($n-1$, P, $SDRes$);
    Naive(n, All);
    $M = (All - SDRes) \cup M$;
    if $n == max\_order$ **then return**;
    Gen_SD_Minset($n+1$, max_order, All, M);
end {Gen_SD_Minset}

Figure 3.10: An algorithm for generating the split-delete minimal subset of the torus obstructions.

The upper bound on the order of the split-delete minimal torus obstructions is not known, hence max_order cannot be set with the certainty that all the split-delete minimal obstructions will be found. We ran Gen_SD_Minset using max_order equal to eleven because computing higher orders was impractical in the time available. Consequently, our set of split-delete minimal torus obstructions is potentially not complete. Order and size statistics about this set are shown in Table 3.2.

An interesting observation that emerged from our efforts to generate a split-delete minimal subset of the torus obstructions using Gen_SD_Minset was that for n, $9 \leq n \leq 11$, the ratio of torus obstructions of order n generated by Gen_SD_Set ($SDRes$) to all the torus obstructions of order n (All) was increasing. Table 3.3 shows this encouraging trend. Based on these figures, it seems likely that the application of Gen_SD_Set to the order eleven torus obstructions generated most of the order twelve obstructions.

	$m = 20$	21	22	23	24	25	26	27	28	29	30	Total
$n = 8$			1		1	1						3
9	2		2	3	3	1	4					15
10	1	1		1	16	40	54	17	1		1	132
11					9	76	50	19	7	3	1	165

Table 3.2: Size and order data for our set of split-delete minimal torus obstructions of order less than twelve.

n	$SDRes$	All	$(SDRes/All) \times 100$
9	33	48	68.75
10	528	660	80
11	4758	4923	96.65

Table 3.3: The ratio of split-delete graphs of order n to all the obstructions of order n is increasing.

3.4 Results

Although we began to generate torus obstructions with Gen_SD_Minset, it proved to be too time consuming to continue using beyond generating all the order eleven obstructions. At that point, beginning with all the order eleven obstructions, Gen_SD_Set was used successively to generate most of the obstructions in our set, up to and including the two order 24 topological obstructions, the largest order obstructions we have found. (The small number of obstructions in our set not generated by Gen_SD_Set were generated by a version of algorithm Gen_DYD that disallows the formation of graphs with multiple edges or vertices of degree less than three – Gen_DYD itself is described in Section 5.2). Tables 3.4 and 3.5 contain order and size statistics about our sets of minor order and topological torus obstructions, respectively.

Our set of torus obstructions includes all the one-connected and disconnected

obstructions. In addition to the two disconnected obstructions already mentioned, there is one more, consisting of two copies of $K_{3,3}$. The order eleven and twelve disconnected obstructions can be derived from the order ten disconnected obstruction by successive applications of Split_Delete. Similarly for the one-connected obstructions, which are all derivable from the order nine obstruction shown in Figure 3.2(a).

The proofs for the completeness of our sets of disconnected and one-connected torus obstructions are given in theorems 3.4.4 and 3.4.5, respectively. Theorems 3.4.1 and 3.4.2, by Battle, Harary, Kodama, and Young [5], as well as Lemma 3.4.3, provide auxiliary results. Graph G is a *block* if there does not exist vertex v of G such that $G - v$ is disconnected. A *component* of G is a subgraph of G that is connected and is not contained in any other connected subgraph of G.

Theorem 3.4.1 *The genus of any graph is the sum of the genera of its blocks.*

Theorem 3.4.2 *The genus of any graph is the sum of the genera of its components.*

Lemma 3.4.3 *The genus of a torus obstruction is two.*

Proof: Let G be a torus obstruction and let the genus of G be greater than two. If the genus of G is greater than two then G does not embed in a surface with less than three handles. Let e be an edge of G. Graph $G - e$ will not embed in a surface with less than two handles since the genus of $G - e$ is at least two. But then $G - e$ is not toroidal, which contradicts the assumption that G is a torus obstruction. □

Theorem 3.4.4 *The disconnected torus obstructions are* $K_5 \bigcup K_5$, $K_{3,3} \bigcup K_5$, *and* $K_{3,3} \bigcup K_{3,3}$.

Proof: Let G be a disconnected torus obstruction. Such a graph has at least two components, none of which can have a vertex of less than degree three (by our definition of a torus obstruction). Also, none of the components can have genus zero since removal or contraction of an edge of such a component does not result

in a graph of smaller genus. Further, none of the components can have genus two since if one did have genus two, then the other component(s) would have to have genus zero (by lemma 3.4.3), which, as just shown, is impossible.

So any disconnected torus obstruction has components of genus one and, therefore, has only two such components.

By Kuratowski's Theorem [cited in [3] p.17], there are only two genus one graphs such that neither has a vertex of less than degree three and such that removing any one of their edges results in a graph of genus zero: K_5 and $K_{3,3}$.

By Wagner's Theorem [cited in [3] p.17], there are only two genus one graphs such that neither has a vertex of less than degree three and such that contracting any one of their edges results in a graph of genus zero: K_5 and $K_{3,3}$. □

Theorem 3.4.5 *There are ten one-connected torus obstructions (shown in Figures 3.2 and 3.3 above).*

Proof: Let G be a one-connected torus obstruction. Such a graph has at least two blocks. None of the blocks are genus zero since removal or contraction of an edge of such a block does not result in a graph of smaller genus. Further, none of the blocks can have genus two since if one did, then the other(s) would have to have genus zero.

So any one-connected torus obstruction has blocks of genus one and, therefore, has only two such blocks. Each block has the property that contraction or removal of any one of its edges results in a graph of genus zero. By Kuratowski's and Wagner's Theorems [cited in [3] p.17], such blocks are homeomorphs of K_5 and $K_{3,3}$.

Neither of the blocks of G has more than one vertex of degree two, otherwise G would have a vertex of less than degree three. This implies that there are just four possibilities for the blocks constituting the one-connected obstructions: K_5, K_5 with a subdivided edge, $K_{3,3}$, and $K_{3,3}$ with a subdivided edge. There are ten possible pairings of these blocks, as shown in Figures 3.2 and 3.3. □

	$n=8$	9	10	11	12	13	14	15	16	17	18
$m=18$				5	1						
19		2	14	2							
20		5	3								
21		2	18	46	52	5					
22	1	9	31	131	238	98	9				
23		13	117	569	1217	835	68				
24	1	6	90	998	2512	1969	461	21			
25	1	2	92	745	1734	1734	894	118	4		
26		4	72	287	422	320	182	38	3		
27			17	43	59	34	23	10	5		
28			1	8	15	41	85	85	41	8	1
29				3							
30			1	1							
31											
32											
total	3	43	456	2838	6250	5036	1722	272	53	8	1

Table 3.4: Size and order data for our set of minor order torus obstructions.

	$n=8$	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
$m=18$				5	1												
19		2	15	2	12												
20		5	9	49	6	12											
21		2	35	87	201	19	9										
22	1	9	40	270	808	820	38										
23		17	190	892	2695	4963	2476	33									
24	1	6	170	1878	6669	12632	15215	3646	20								
25	1	2	102	1092	5951	14590	24153	22401	2689								
26		5	76	501	1681	4496	15486	20800	17469	837							
27			21	120	364	786	3208	8150	10515	8099	133						
28			1	22	68	138	586	1615	3032	4127	2332						
29				4	2	10	166	612	1207	1080	1458	393					
30			1	1			6	173	673	1043	506	435	39				
31								4	66	367	636	292	16				
32									1	11	102	255	162	4			
33											1	15	7	63	2		
34														1	22		
35																4	
36																	2
total	3	48	660	4923	18458	38466	61343	57434	35672	15564	5168	1390	224	68	24	4	2

Table 3.5: Size and order data for our set of topological torus obstructions.

Chapter 4

Finding cubic torus obstructions

Early in this project, we wanted an idea of how large torus obstructions could be. At the time, the largest was order 20 [20]. The cubic graphs seemed a favourable place to search for such obstructions for two reasons. First, there were a manageable number of cubic graphs in the range in which we were interested (22 and 24 vertices). There are about seven million cubics of order 22 and about 118 million of order 24, amounts that Obstruction_Test(G) can process in about two weeks, assuming 41 processors are used. Second, because the set of obstructions is finite, we expected that as the number of vertices increased, the maximum vertex degree would decrease, with three being the lower limit.

Using Naive, as described in Chapter 3, all connected cubic torus obstructions of orders 22 and 24 were determined. For completeness, all connected cubic obstructions between orders 8 and 20 inclusive were also determined. There is only one disconnected cubic obstruction, the graph consisting of two $K_{3,3}$'s. See Table 4.1 for size and order statistics about these results. Only about 20 million of the roughly two trillion cubics of order 26 have been tested, none of which are obstructions.

Order	# obstructions
12	1
14	9
16	20
18	133
20	39
22	2
24	2
Total	206

Table 4.1: Frequency and order data for the set of cubic torus obstructions of order less than 26.

Chapter 5

Generating minor order torus obstructions that are projective planar

In this chapter, Dan Archdeacon's proposal for generating the complete set of minor order torus obstructions is sketched and a first step in its implementation taken. In Section 5.1, the overall two-step proposal is summarized. The first step requires finding all the projective planar minor order torus obstructions. Results established by Randby [23] and Fiedler, Huneke, Richter, and Robertson [9] allow this to be easily accomplished. Section 5.2 explains the algorithm used to find them, and Section 5.3 discusses how the algorithm was applied and the results.

5.1 Archdeacon's proposal

Archdeacon [1] sketches the following approach to finding the minor order torus obstructions.

1. Begin by determining the minor order torus obstructions that are also projective planar. The remaining minor order torus obstructions will not be

projective planar and, therefore, will contain a subgraph homeomorphic to one of the 103 projective plane obstructions.

2. Discover a way to systematically extend the projective plane obstructions to generate the remaining minor order torus obstructions.

Step two is expected to be difficult and has not been attempted here. Step one, however, we have accomplished. Before presenting the process used to generate the projective planar minor order torus obstructions, the algorithm central to that process is described.

5.2 The Gen_DYD algorithm

In this section, the $Y - \triangle$ and $\triangle - Y$ operations are defined, then the Gen_DYD algorithm which incorporates them is presented, and finally some features of the algorithm will be noted.

If vertex v of graph G has degree three and $N(v) = \{a, b, c\}$ is the set of vertices adjacent to v, then a $Y - \triangle$ operation on v consists in deleting v and its incident edges from G and adding edges (a, b), (a, c), and (b, c) to G. This process will result in multiple edges if any of edges (a, b), (a, c), or (b, c) exist before the application of $Y - \triangle$.

A *triangle* of graph G is a subgraph of G which is isomorphic to the complete graph on three vertices. If T is a triangle of G with edges (u, v), (u, w), and (v, w), then a $\triangle - Y$ operation on T consists in deleting edges (u, v), (u, w), and (v, w) from G, then adding vertex p and edges (u, p), (v, p), and (w, p) to G. This operation will result in a vertex of degree less than three if any of vertices u, v, or w has degree two or three before the application of $\triangle - Y$.

Given a graph G, Gen_DYD, shown in Figure 5.1, creates all graphs obtainable from G by some sequence of the $Y - \triangle$ and $\triangle - Y$ operations. That Gen_DYD will generate a finite number of graphs from any input graph G is ensured by the fact

```
function Gen_DYD( G : graph; Result : array )
local front, end : integer; G_can, J_can, K_can : graph;
begin
    front = end = 0;
    Put G in canonical form, G_can, using nauty;
    Add G_can to Result[end];
    end++;
    while end − front > 0 do begin
        Get graph H at Result[front];
        front++;
        for each degree three vertex v of H do begin
            Apply Y − △ to v to generate graph J;
            Put J in canonical form, J_can;
            if J_can is not in Result then begin
                add J_can to Result[end];
                end++;
            end if
        end for
        for every triangle T of H do begin
            Apply △ − Y to T to generate graph K;
            Put K in canonical form, K_can;
            if K_can is not in Result then begin
                add K_can to Result[end];
                end++;
            end if
        end for
    end while
end {Gen_DYD}
```

Figure 5.1: The algorithm applying the $Y - \triangle$ and $\triangle - Y$ operations.

that all generated graphs have the same size as G. Also, Gen_DYD has the following property: let S be the set of graphs obtained by applying the algorithm to graph G and let H be any element of S, then S can be generated by applying the algorithm to H. Thus, the algorithm induces an equivalence relation on all graphs, and the graphs in S form an equivalence class.

5.3 Application of Gen_DYD and Results

The process used to generate the projective planar minor order torus obstructions derives from theorems published by Randby [23] and Fiedler, Huneke, Richter and Robertson [9]. In this section, the theorems will be discussed and the results of the generation process presented.

Let C be a continuous closed curve on surface Σ. Curve C is *noncontractible* in Σ if it cannot be continuously contracted to a point in Σ. Let Γ be an embedding of graph G in surface Σ. The *face-width* or *representativity* of Γ, denoted $\rho(\Gamma)$, is the smallest integer k such that there exists a noncontractible closed curve in Σ which meets Γ in k points.

The three operations of contracting an edge, deleting an edge, and deleting an isolated vertex are called *minor operations.* When these operations are applied to an embedding, the structure of the rest of the embedding is preserved. If embedding Γ_1 is obtained by the application of a finite sequence of minor operations to embedding Γ_2, then Γ_1 is called a *minor* of Γ_2, denoted $\Gamma_1 \leq_m \Gamma_2$. If embedding Γ_1 is a minor of embedding Γ_2 and Γ_1 is not equal to Γ_2, then Γ_1 is a *proper minor* of Γ_2. If $\rho(\Gamma) \geq n$ for some embedding Γ, and every proper minor of Γ has face-width less than n, then Γ is called a *minor minimal face-width n* embedding.

The following remarkable theorem allows us to determine the face-width of a projective plane embedding of a minor order torus obstruction:

Theorem 5.3.1 (Fiedler,Huneke,Richter,Robertson [9, p.298]) *Let G be a*

nonplanar graph and let Γ *be an embedding of* G *in the projective plane. Then the following are equivalent for integer* $n > 0$*:*

1. *the face-width of* Γ *is one of* $2n$ *or* $2n+1$*;*

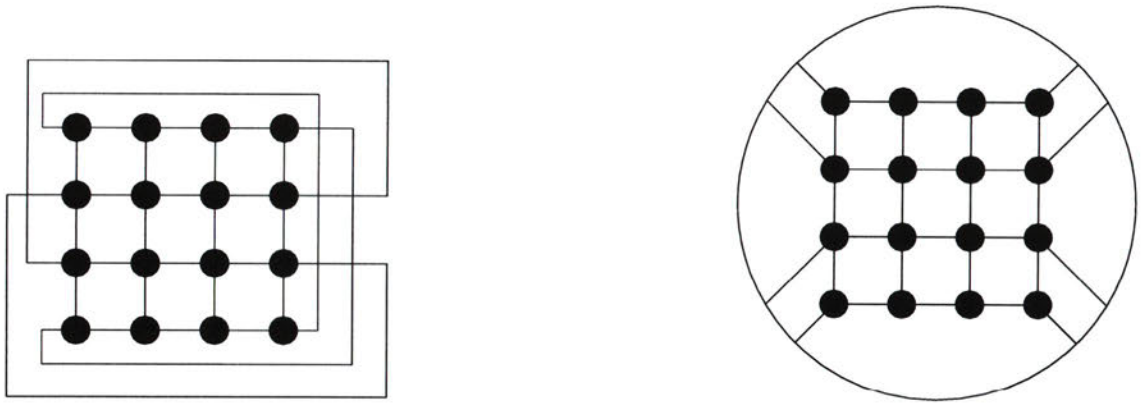
2. *the orientable genus of* G *is* n*.* □

Since the orientable genus of a torus obstruction is two, Theorem 5.3.1 implies that if a torus obstruction has a projective plane embedding then the face-width of that embedding is either four or five. A minor operation on a minor order torus obstruction results in a graph of genus one. By Theorem 5.3.1, a projective plane embedding of a graph of genus one has face-width two or three. Since a minor operation can reduce the face-width of a projective plane embedding by at most one, a projective plane embedding of a minor order torus obstruction must have face-width four. Furthermore, such embeddings must be minor minimal face-width four embeddings.

Randby's result, stated in the following theorem, implies that all graphs with a minor minimal face-width four projective plane embedding are in the same Gen_DYD equivalence class:

Theorem 5.3.2 (Randby [23, Theorem 2.1, p.156]) *Let* $n \geq 2$ *and let* Γ *be a minor minimal face-width* n *embedding in the projective plane. Then* Γ *is either an embedding of an* $n \times n \times n$ *projective grid or an embedding obtainable from it by a sequence of* $\triangle - Y$ *and/or* $Y - \triangle$ *operations.* □

Theorems 5.3.1 and 5.3.2 together indicate how to find the projective planar minor order torus obstructions. The result by Fiedler, Huneke, Richter, and Robertson establishes that any projective plane embedding of a projective planar minor order torus obstruction has face-width four and that such an embedding will be minor minimal with respect to face-width. Using Randby's result, the projective planar graphs with a minor minimal face-width four embedding can be generated by ap-



(a) (b)

Figure 5.2: A $4 \times 4 \times 4$ projective grid and its projective plane embedding.

plying Gen_DYD to a $4 \times 4 \times 4$ projective grid. Figure 5.2 shows (a) a $4 \times 4 \times 4$ projective grid along with (b) its projective plane embedding.

Applying Gen_DYD to the $4 \times 4 \times 4$ projective grid yields a set of 270 graphs, all of which are minor order torus obstructions of size 28. Order and frequency statistics about this set are shown in Table 5.1. Because the dual of a minor minimal face-width four projective plane embedding is a minor minimal face-width four projective plane embedding, the results in Table 5.1 display the expected symmetry.

Order	Frequency
11	1
12	8
13	41
14	85
15	85
16	41
17	8
18	1
Number of graphs	270

Table 5.1: Order and frequency data for the set of projective planar minor order torus obstructions.

Chapter 6

Future work

Although we have generated a large number of torus obstructions, generating a provably complete set remains to be done. One way of doing so would be to determine the upper bound u on the order of the split-delete minimal torus obstructions. The Naive algorithm – perhaps with a faster torus testing algorithm – could be used to generate all obstructions up to and including those of order u. Algorithm Gen_SD_Set could then be applied repeatedly, starting with the order u obstructions, until no further obstructions are found. The resulting set of obstructions would be complete.

Short of a conclusive effort, various projects could be undertaken that would enhance present results. We will list some possibilities below; however, before undertaking any of the them, it is strongly recommended that a different and faster torus tester be substituted for the one presently in use in the decision algorithm Obstruction_Test. Implementing Juvan and Mohar's algorithm [14] might be a worthwhile project, if not already done. What follows are suggestions for verifying and extending present results and assumes that a different and faster torus tester is used.

1. Verify that all the obstructions in our collection are torus obstructions and of the right type (topological or minor order).

2. Use Naive to collect complete sets of torus obstructions of orders higher than eleven.

3. Use Gen_SD_Minset to verify the partial split-delete minimal set we have collected and extend it, if possible.

4. Use Gen_SD_Set to obtain new torus obstructions from any new split-delete minimal obstructions.

5. Theorem 6.0.3 shows that any topological (and not minor order) obstruction can be reduced to a minor order obstruction. New topological torus obstructions may reduce to new minor order ones in this way.

   Theorem 6.0.3 *If G is a topological (and not a minor order) obstruction for surface Σ, then G is reducible to a minor order obstruction by a sequence of edge contractions.*

   Proof: Consider the following process:

   Step 1. Because G is a topological (and not a minor order) obstruction, there is an edge e of G such that G contract e is not embeddable on Σ. Let H be $G \circ e$.

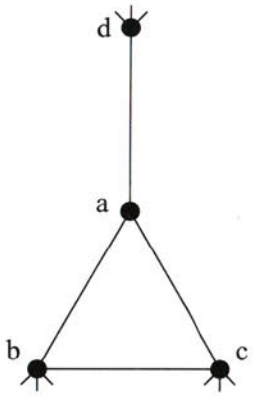
   Step 2. Graph H is either a topological (and not a minor order) obstruction or it is a minor order obstruction. If the latter, then we have the desired result. If the former, then let G equal H and go to step one.

   Since both the size and order of the successive graphs produced by this process are decreasing, the resulting sequence of graphs is finite and the last graph of the sequence is a minor order obstruction for Σ. □

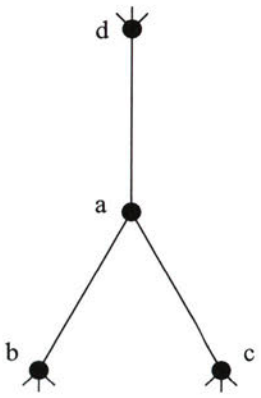
6. Use Naive to determine if there are any order 26 cubic obstructions. If any are found, successively contract edges to generate new smaller order obstructions. The following result implies that the search can be restricted to graphs of girth four or greater, where *girth* is the length of the shortest cycle in the graph.

Theorem 6.0.4 *If G is an obstruction for surface Σ then G has no triangles such that some vertex of the triangle has degree three.*

Proof: Suppose that G is an obstruction for surface Σ and it has a triangle with edges (a, b), (a, c), and (b, c) and vertex a has degree three:



Suppose G has genus g. Because G is an obstruction, $G - e$ has genus $g - 1$ for all edges e of G. Consider an embedding of $G - (b, c)$ in the surface of genus $g - 1$:



The order of the neighbours of a are either d, c, b or d, b, c. In case of an embedding with the latter order, a mirror reversal of the embedding yields one with the former order. Since b and c are consecutive neighbours of a, the edge (b, c) can be added to the embedding without increasing the genus. This contradicts the assumption that G has genus g □

7. Beginning with the eight-vertex torus obstructions, generate Gen_DYD equivalence classes for all known obstructions, taking care to avoid processing obstructions of classes that have already been checked. It is possible that some new obstructions can be generated this way.

8. During the Delete portion of the Split_Delete algorithm, if a deletion results in a non-toroidal graph with at least one vertex of degree two, then that graph is not checked by Obstruction_Test to determine if it is an obstruction. The Obstruction_Test algorithm could be modified so that if a non-toroidal graph G with at least one vertex of degree two is encountered, all degree two vertices of G are removed and replaced with an edge, and then G is checked to determine if it is an obstruction. More extensive reductions involving leaf and isolated point removal are possible but may incur an impractical time cost.

9. Following Archdeacon [3], all the projective plane obstructions of order less than or equal to n could be extended to order n by adding vertices and edges in such a way that the graphs produced each have a subgraph homeomorphic to one of the projective plane obstructions. The set would then be checked for torus obstructions.

Finding torus obstructions has been an outstanding problem for at least ten years, taking Bodendiek and Wagner's interest in the subject in 1989 as a benchmark. Prior to this research, the largest number of torus obstructions generated by a single effort was 3884 by Neufeld and Myrvold [22]. We have substantially added to that number, collecting a total of 239,451 topological obstructions, 16,682 of which are minor order. It is hoped that such a substantial set should prove valuable to future research in this area.

Bibliography

[1] D. Archdeacon. Problems in topological graph theory – questions I can't answer. *Yokohama Mathematical Journal*, 47(1999), Special Issue, 89-92.

[2] D. Archdeacon. A Kuratowski theorem for the projective plane. *Journal of Graph Theory*, 5:243–246, 1981.

[3] D. Archdeacon. Topological graph theory: A survey. *Congressus Numeratum*, 115:5–54, 1996.

[4] D. Archdeacon and J. P. Huneke. A Kuratowski theorem for nonorientable surfaces. *Journal of Combinatorial Theory, Series B*, 46:173–231, 1989.

[5] J. Battle, F. Harary, Y. Kodama, and J. W. T. Youngs. Additivity of the genus of a graph. *Bulletin of the American Mathematical Society*, 68:565-568, 1962.

[6] R. Bodendiek and K. Wagner. Solution to König's graph embedding problem. *Math. Nachr.*, 140:251–272, 1989.

[7] J. A. Bondy and U. S. R. Murty. *Graph theory with applications.* North-Holland, 1976.

[8] K. Cattell, M. J. Dinneen, R. G. Downey, M. R. Fellows and M. A. Langston. On computing graph minor obstruction sets. *Theoretical Computer Science*, 233 (2000), no.1-2, 107-127.

[9] J. R. Fiedler, J. P. Huneke, R. B. Richter, and N. Robertson. Computing the orientable genus of projective graphs. *Journal of Graph Theory*, 20(3):297–308, 1995.

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

[11] A. Gibbons. *Algorithmic graph theory.* Cambridge University Press, 1995.

[12] H. Glover, J. Huneke, and C. Wang. 103 graphs that are irreducible for the projective plane. *Journal of Combinatorial Theory, Series B*, 27:332–370, 1979.

[13] M. Henle. *A combinatorial introduction to topology.* Dover Publications, Inc., New York, 1994.

[14] M. Juvan and B. Mohar. A simplified algorithm for embedding graphs in the torus. Submitted, 10 pages, 2002.

[15] L. C. Kinsey. *Topology of Surfaces.* Undergraduate Texts in Mathematics. Springer-Verlag, New York, 1993.

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

[17] B. D. McKay. `nauty` user's guide (version 1.5). Technical Report TR-CS-90-02, Department of Computer Science, Australian National University, 1990.

[18] B. D. McKay. Isomorph-free exhaustive generation. *Journal of Algorithms*, 26(2):306–324, Feb. 1998.

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

[20] W. Myrvold. Personal communication.

[21] E. Neufeld. Practical toroidality testing. Master's thesis, Department of Computer Science, University of Victoria, 1993.

[22] E. Neufeld and W. Myrvold. Practical toroidality testing. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (New Orleans, Louisiana, January 5–7, 1997)*, pages 574–580, 1997.

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

[24] N. Robertson and P. Seymour. Graph minors VIII: A Kuratowski theorem for general surfaces. *Journal of Combinatorial Theory, Series B*, 48:255–288, 1990.

[25] M. Skala. Generation of graphs embedded on the torus. Master's thesis, Department of Computer Science, University of Victoria, 2001.

[26] D. B. West. *Introduction to graph theory.* Prentice Hall, 1996.

VITA

Surname: Chambers Given Names: John

Place of Birth: London, Ontario, Canada

Educational Institutions Attended:

University of Western Ontario	1986 to 1991
University of Victoria	1996 to 2002

Degrees Awarded:

B.A.	University of Western Ontario	1991

UNIVERSITY OF VICTORIA PARTIAL COPYRIGHT LICENSE

Title of Thesis:

Hunting for Torus Obstructions

Author [redacted]

John Chambers

April, 2002