Discrete Comput Geom 38:527–571 (2007)
DOI: 10.1007/s00454-007-1307-x

Discrete & Computational
Geometry

Enumeration of Non-Orientable 3-Manifolds Using Face-Pairing Graphs and Union-Find*

Benjamin A. Burton

Department of Mathematics, SMGS, RMIT University,
GPO Box 2476V, Melbourne, VIC 3001, Australia
bab@debian.org

Abstract. Drawing together techniques from combinatorics and computer science, we improve the census algorithm for enumerating closed minimal $\mathbb{P}^2$-irreducible 3-manifold triangulations. In particular, new constraints are proven for face-pairing graphs, and pruning techniques are improved using a modification of the union-find algorithm. Using these results we catalogue all 136 closed non-orientable $\mathbb{P}^2$-irreducible 3-manifolds that can be formed from at most 10 tetrahedra.

1. Introduction

With recent advances in computing power, topologists have been able to construct exhaustive tables of small 3-manifold triangulations, much like knot theorists have constructed exhaustive tables of simple knot projections. Such tables are valuable sources of data, but they suffer from the fact that enormous amounts of computer time are required to generate them.

Where knot tables are often limited by bounding the number of crossings in a knot projection, tables of 3-manifolds generally limit the number of tetrahedra used in a 3-manifold triangulation. A typical table (or *census*) of 3-manifolds lists all 3-manifolds of a particular type that can be formed from n tetrahedra or fewer.

Beyond their generic role as a rich source of examples, tables of this form have a number of specific uses. They still offer the only general means for proving that a triangulation is minimal (i.e., uses as few tetrahedra as possible), much in the same way as knot tables are used to calculate crossing number. Moreover, a detailed analysis of

* This project was supported by the Victorian Partnership for Advanced Computing e-Research Program Grant Scheme under Research Grant EPANRM155.2005.

these tables can offer insight into the combinatorial structures of minimal triangulations, as seen for example by the structural observations of Matveev [20], Martelli and Petronio [17] and Burton [5].

Unfortunately the scope of such tables is limited by the difficulty of generating them. In general, a census of triangulations formed from $\leq n$ tetrahedra requires computing time at least exponential in n. In the case of closed 3-manifold triangulations, results are only known for ≤ 11 tetrahedra in the orientable case and ≤ 8 tetrahedra in the non-orientable case. These results are particularly sparse in the non-orientable case—only 18 distinct manifolds are found, all of which are graph manifolds [8].

Clearly there is more to be learned by extending the existing censuses to higher numbers of tetrahedra. Due to the heavy computational requirements however, this requires significant improvements in the algorithms used to generate the census data. Such improvements form the main subject of this paper.

We restrict our attention here to closed 3-manifold triangulations. In the orientable case, successive tables have been generated by Matveev [20], Ovchinnikov, Martelli and Petronio [15], Martelli [14] and then Matveev again with 11 tetrahedra [21]. For non-orientable manifolds, tabulation begins with Amendola and Martelli [2], [3] and is continued by Burton [5], [8] up to eight tetrahedra.

The contributions of this paper are the following:

- Several improvements to the algorithm for generating census data, some of which increase the speed by orders of magnitude.
- An extension of the closed non-orientable census from 8 tetrahedra to 10 tetrahedra.
- A verification of previous closed orientable census results for up to 10 tetrahedra, and an extension of these results from a census of manifolds to a census of all minimal triangulations.

The algorithmic improvements are divided into two broad categories. The first set of results relate to *face-pairing graphs*, which are 4-valent graphs describing which tetrahedron faces are identified to which within a triangulation. The second set is based upon the *union-find* algorithm, which is a well-known method for finding connected components in a graph. Here the union-find algorithm is modified to support backtracking, and to monitor efficiently properties of vertex and edge links within a triangulation.

All computational work was performed using the topological software package *Regina* [4], [7]. Census generation forms only a small part of *Regina*, which is a larger software package for performing a variety of tasks in 3-manifold topology. The software is released under the GNU General Public License, and may be freely downloaded from `http://regina.sourceforge.net/`.

The bulk of this paper is devoted to improving the census algorithm. Section 2 begins with the precise census constraints, and follows with an overview of how a census algorithm is structured. In Section 3 we present a series of preliminary results, describing properties of minimal triangulations that will be required in later sections. Section 4 offers the first round of algorithmic improvements, based upon the analysis of face-pairing graphs. A more striking set of improvements is made in Section 5, in which a modified union-find algorithm is used to reduce the search space greatly. Both Sections 4 and 5 also include empirical results in which the effectiveness of these improvements is measured.

The improvements of Sections 4 and 5 have led to new closed census results, as outlined above. Section 6 summarises these new results, with a focus on the extension of the closed non-orientable census from 8 to 10 tetrahedra. A full list of non-orientable census manifolds is included in the Appendix.

2. Overview of the Census Algorithm

As is usual for a census of closed 3-manifolds, we restrict our attention to manifolds with the following properties:

- *Closed*: The 3-manifold is compact, with no boundary and no cusps.
- $\mathbb{P}^2$*-irreducible*: The 3-manifold contains no embedded two-sided projective planes, and every embedded 2-sphere bounds a ball.

The additional constraint of $\mathbb{P}^2$-irreducibility allows us to focus on the most "fundamental" manifolds—the properties of larger manifolds are often well understood in terms of their $\mathbb{P}^2$-irreducible constituents.

Recall that we are not just enumerating 3-manifolds, but also their triangulations. Throughout this paper we consider a *triangulation* to be a finite collection of n tetrahedra, where some or all of the $4n$ tetrahedron faces are affinely identified in pairs. For the census we focus only on triangulations with the following additional property:

- *Minimal*: The triangulation uses as few tetrahedra as possible. That is, the underlying 3-manifold cannot be triangulated using a smaller number of tetrahedra.

This minimality constraint is natural for a census, and is used throughout the literature. Note that a 3-manifold may have many different minimal triangulations, though of course all of these triangulations must use the same number of tetrahedra.

Minimal triangulations are tightly related to the Matveev complexity of a manifold [19]. Matveev defines complexity in terms of special spines, and it has been proven by Matveev in the orientable case and Martelli and Petronio in the non-orientable case [16] that, with the exceptions of S^3, $\mathbb{R}P^3$ and $L_{3,1}$, the Matveev complexity of a closed $\mathbb{P}^2$-irreducible 3-manifold is precisely the number of tetrahedra in its minimal triangulation(s).

2.1. *Stages of the Algorithm*

There are two stages involved in constructing a census of 3-manifold triangulations: the generation of triangulations, and then the analysis of these triangulations.

1. *Generation*: The generation stage typically involves a long computer search, in which tetrahedra are pieced together in all possible ways to form 3-manifold triangulations that might satisfy our census constraints.

   The result of this search is a large set of triangulations, guaranteed to include all of the triangulations that should be in the census. There are often unwanted triangulations also (for instance, triangulations that are non-minimal, or that represent

reducible manifolds). This is not a problem; these unwanted triangulations will be discarded in the analysis stage.

The generation of triangulations is entirely automated, but it is also extremely time-consuming—it may take seconds or centuries, depending upon the size of the census.

2. *Analysis*: Once the generation stage has produced a raw set of triangulations, these must be refined into a final census. This includes verifying that each triangulation is minimal and $\mathbb{P}^2$-irreducible (and throwing away those triangulations that are not). It also involves grouping triangulations into classes that represent the same 3-manifold, and identifying these 3-manifolds.

   Analysis is much faster than generation, but it typically requires a mixture of automation and human involvement. Techniques include the analysis of invariants and normal surfaces, combinatorial analysis of the triangulation structures, and applying elementary moves that change triangulations without altering their underlying 3-manifolds.

The generation stage is the critical bottleneck, due to the vast number of potential triangulations that can be formed from a small number of tetrahedra. Suppose we are searching for triangulations that can be formed using n tetrahedra. Even assuming that we know which tetrahedron faces are to be joined with which, each pair of faces can be identified according to one of six possible rotations or reflections, giving rise to 6^{2n} possible triangulations in total. For 10 tetrahedra, this figure is larger than 10^{15}. It is clear then why existing census data is limited to the small bounds that have been reached to date.

It should be noted that for an orientable census, the figure 6^{2n} becomes closer to $3^n 6^n$. This is because for a little over half the faces only three of the six rotations or reflections will preserve orientation. It is partly for this reason that the orientable census has consistently been further advanced in the literature than the non-orientable census. Nevertheless, for 10 tetrahedra this figure is still larger than 10^{12}, a hefty workload indeed.

Fortunately there are ways in which these figures can be reduced. The usual technique is to use topological arguments to derive constraints that must be satisfied by minimal $\mathbb{P}^2$-irreducible triangulations, and then to incorporate these constraints into the generation algorithm so that the search space can be reduced. Such techniques have been used throughout the history of census generation; examples include the edge degree results from the early cusped hyperbolic census of Hildebrand and Weeks [10], and the properties of minimal spines from Matveev's 6-tetrahedron closed orientable census [20].

2.2. *Generation of Triangulations*

Since generation is the most time-intensive stage of the census algorithm, it is there that we focus our attention. Before describing the generation process in more detail, it is useful to make the following definition.

Definition 2.1 (Face-Pairing Graph). Let T be a closed 3-manifold triangulation formed from n tetrahedra. The *face-pairing graph* of T is a 4-valent multigraph on n vertices describing which faces of which tetrahedra in T are identified.

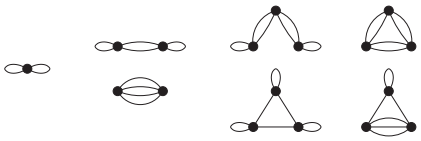

Fig. 1. All possible face-pairing graphs for ≤ 3 tetrahedra.

More specifically, each vertex of the face-pairing graph represents a tetrahedron of T, and each edge represents a pair of tetrahedron faces that are identified. Note that loops and multiple edges may (and frequently do) occur.

It is clear that every face-pairing graph must be 4-valent, since each tetrahedron has four faces. Figure 1 illustrates all possible face-pairing graphs for connected triangulations formed from ≤ 3 tetrahedra.

We return now to the generation process. Algorithm 2.2 outlines this process in more detail. This general scheme has been used by authors throughout the literature; a similar overview can be found in the early cusped hyperbolic census of Hildebrand and Weeks [10].

Algorithm 2.2 (Generation of Triangulations). Suppose that we wish to generate a set S that includes all closed minimal $\mathbb{P}^2$-irreducible triangulations formed from precisely n tetrahedra (note that S might also include some additional unwanted triangulations). The procedure is as follows:

1. *Enumerate face-pairing graphs*
   We enumerate (up to isomorphism) all connected 4-valent multigraphs on n vertices. Each of these becomes a candidate face-pairing graph for an eventual triangulation.
2. *Process face-pairing graphs*
   Each candidate face-pairing graph generated in step 1 is processed as follows. For each edge of the graph, we consider all six rotations and reflections by which the two corresponding tetrahedron faces might be identified. In this way we obtain 6^{2n} possible triangulations.

   In practice, constraints that must be satisfied by minimal $\mathbb{P}^2$-irreducible triangulations are used to prune this search. As a result, far fewer than 6^{2n} possible triangulations need to be processed.

   Some of these possible triangulations might not be triangulations of closed 3-manifolds (e.g., their vertex links might not be spheres, or they might have edges that are identified with themselves in reverse). Each triangulation that is indeed a closed 3-manifold triangulation is added to the final set S.

The enumeration of face-pairing graphs (step 1) is extremely fast—for 10 tetrahedra it takes a little over 5 minutes on a 2.40 GHz Pentium 4. As such, there is no urgency to improve the efficiency of this enumeration at the present time; a full list of candidate face-pairing graphs has been obtained for up to 13 tetrahedra, well beyond the current census limits. Table 1 presents the number of connected 4-valent multigraphs in these lists.

Table 1. The number of connected 4-valent multigraphs for ≤ 13 tetrahedra.

# Tet.	Graphs	# Tet.	Graphs	# Tet.	Graphs
1	1	6	97	11	316,520
2	2	7	359	12	2,305,104
3	4	8	1,635	13	18,428,254
4	10	9	8,296		
5	28	10	48,432		

On the other hand, the processing of face-pairing graphs (step 2) is extremely slow, consuming virtually the entire running time of the generation algorithm. For the 10-tetrahedron non-orientable census this processing required approximately $3\frac{2}{3}$ years of CPU time, though this was distributed amongst many machines in parallel with a final cost of only $1\frac{1}{2}$ months by the wall clock.

The division of tasks in Algorithm 2.2 lends itself to two avenues for improvement, both of which are explored in this paper.

- We can prove constraints that must be satisfied by the face-pairing graph of a closed minimal $\mathbb{P}^2$-irreducible 3-manifold triangulation. Graphs that do not satisfy these constraints can be discarded after step 1, and do not need to be processed.

  Several results of this type are proven in [6], and we prove more in Section 4. Since the enumeration of graphs takes negligible time, if we can avoid processing $k\%$ of potential graphs in this way then we can expect to save roughly $k\%$ of the overall running time.
- We can improve the processing itself, finding ways to prune the search for possible triangulations in step 2 of the algorithm. In Section 5 we accomplish this by using a modified union-find algorithm that tracks vertex and edge links. As a consequence we are able to decrease running times by orders of magnitude, as seen in the experimental results of Section 5.4.

3. Preliminary Results

Before moving on to specific algorithmic improvements, we present a number of useful properties of minimal $\mathbb{P}^2$-irreducible triangulations. These are simple structural properties that are easily tested by computer, and many will be called upon in later sections of this paper. We begin with a number of results from the literature, and then present some new variants on existing results.

Our first property constrains the number of vertices in a minimal $\mathbb{P}^2$-irreducible triangulation. The following lemma was originally proven in the equivalent setting of special spines, in the orientable case by Matveev [19] and in the non-orientable case by Martelli and Petronio [16].

Lemma 3.1. *Let M be a closed $\mathbb{P}^2$-irreducible 3-manifold that is not S^3, $\mathbb{R}P^3$ or $L_{3,1}$. Then every minimal triangulation of M contains precisely one vertex.*

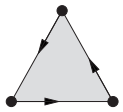

Fig. 2. All three edges of a face identified with each other.

It will prove useful in later sections to count edges as well as vertices. A quick Euler characteristic calculation yields the following immediate consequence.

Corollary 3.2. *Let T be a minimal triangulation of a closed $\mathbb{P}^2$-irreducible 3-manifold that is not S^3, $\mathbb{R}P^3$ or $L_{3,1}$, and suppose that T is formed from n tetrahedra. Then T contains precisely $n+1$ edges.*

The following structural results are proven for both orientable and non-orientable triangulations by this author in [6]. When restricted to the orientable case, a number of similar results have been proven by other authors; see [6] again for full details.

Lemma 3.3. *Let T be a closed minimal $\mathbb{P}^2$-irreducible triangulation containing ≥ 3 tetrahedra. Then no edge of T has degree one or two. Moreover, if an edge of T has degree three then that edge does not lie on three distinct tetrahedra (instead, it meets some tetrahedron more than once).*

Lemma 3.4. *Let T be a closed minimal $\mathbb{P}^2$-irreducible triangulation containing ≥ 3 tetrahedra. Then no face of T has all three of its edges identified with each other in the same direction around the face, as illustrated in Fig. 2.*

Lemma 3.5. *Let T be a closed minimal $\mathbb{P}^2$-irreducible triangulation containing ≥ 3 tetrahedra. Then no face of T has two of its edges identified to form a cone, as illustrated in Fig. 3.*

The next result regarding two-face cones is proven in a more general form by the author in [6]. Here we only require a simpler variant in which the interior edges of the cone are distinct.

Lemma 3.6. *Let T be a triangulation of a closed 3-manifold containing ≥ 3 tetrahedra. Suppose that two distinct faces of T are joined together along their edges to form a cone, as illustrated in Fig. 4. Moreover, suppose that the two internal edges of this cone are distinct (i.e., they are not identified in the overall triangulation). Then either T is non-minimal, or the 3-manifold that T represents is not $\mathbb{P}^2$-irreducible.*

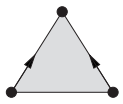

Fig. 3. One face with its edges identified to form a cone.

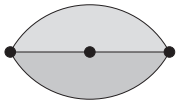

Fig. 4. Two faces joined together to form a cone.

Having cited results regarding one-face cones and two-face cones, we proceed to prove a similar result regarding three-face cones. A restriction of this result to orientable triangulations is proven by Martelli and Petronio in [15].

Lemma 3.7. *Let T be a closed minimal $\mathbb{P}^2$-irreducible triangulation containing ≥ 3 tetrahedra. Suppose that three faces of T are joined together along their edges to form a cone, as illustrated in Fig.* 5. *Moreover, suppose that the three internal edges of this cone are distinct* (*i.e., they are not identified in the overall triangulation*). *Then the three faces of this cone are distinct faces of the triangulation, and all three faces belong to a single tetrahedron. More specifically, this cone is the cone surrounding some vertex of the tetrahedron, as illustrated in Fig.* 6.

Proof. It is simple to see that the three faces of the cone are distinct. A quick run through the six possible ways in which two faces could be identified shows that, in each case, either two internal edges are also identified (contradicting the conditions of the lemma) or some internal edge is identified with itself in reverse (giving a structure that is not a 3-manifold triangulation).

Suppose then that the three-face cone does not surround some vertex of a single tetrahedron. Let n be the number of tetrahedra in triangulation T. Our strategy is to expand the triangulation in the region of the cone, and then to compress it along an edge to obtain a new triangulation of the same 3-manifold that uses fewer than n tetrahedra. This mirrors the procedure used by Martelli and Petronio in the orientable case. We take each step carefully in order to avoid corner cases that might behave unexpectedly.

Since the three interior edges of the cone are distinct, the interior of the cone is embedded within the triangulation. This allows us to thicken the interior of the cone as follows. We split the centre of the cone into two separate vertices and pull these vertices apart. Likewise, the original cone is pulled apart into two cones (an upper and a lower cone), joined along their three boundary edges. In the empty space that is created, we insert two tetrahedra.

This expansion is illustrated in Fig. 7, where the original vertex U is split into two vertices V and W. The two new tetrahedra are separated by the internal face $\Delta X_1X_2X_3$, with

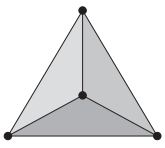

Fig. 5. Three faces joined together to form a cone.

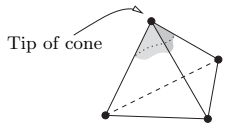

Fig. 6. A three-face cone surrounding a single tetrahedron vertex.

the upper tetrahedron containing vertices V, X_1, X_2 and X_3, and the lower tetrahedron containing vertices W, X_1, X_2 and X_3.

The result is a new triangulation T' that represents the same 3-manifold as T using precisely $n + 2$ tetrahedra. More importantly, the two new vertices V and W are distinct. Our aim now is to find an edge joining two distinct vertices that is suitable for compression.

Since the original cone does not surround the vertex of a single tetrahedron either above or below, it follows that the new upper faces ΔX_1X_2V, ΔX_2X_3V and ΔX_3X_1V collectively meet at least three distinct tetrahedra, including the new tetrahedron $X_1X_2X_3V$ (the only remaining possibilities for ≤ 2 distinct tetrahedra give a vertex link for V that is not a 2-sphere). Likewise, the lower faces ΔX_1X_2W, ΔX_2X_3W and ΔX_3X_1W collectively meet at least three distinct tetrahedra. Therefore at least two of the edges $\overline{X_1V}$, $\overline{X_2V}$ and $\overline{X_3V}$ meet ≥ 3 distinct tetrahedra each, and at least two of the edges $\overline{X_1W}$, $\overline{X_2W}$ and $\overline{X_3W}$ meet ≥ 3 distinct tetrahedra each.

It follows that for some $i \in \{1, 2, 3\}$, each of the edges $\overline{X_iV}$ and $\overline{X_iW}$ meets ≥ 3 distinct tetrahedra. Since vertices V and W are distinct, one of these two edges must join two distinct vertices in the triangulation. Call this edge e.

Our new aim is to compress the expanded triangulation T' along edge e, producing a new triangulation T'' with $< n$ tetrahedra. The compression is performed as follows:

1. We shrink edge e to a single point. Since e joins two distinct vertices, this does not alter the underlying 3-manifold. However, every tetrahedron that contains e changes its shape according to Fig. 8.

   The possible ways in which a tetrahedron may contain edge e one or more times are as follows:
   - The tetrahedron contains edge e only once. In this case the result is a triangular pillow with two bigon sides, as seen in the upper diagram of Fig. 8.
   - The tetrahedron contains edge e twice, where e appears as two opposite edges of the tetrahedron. The result here is a football with four bigon faces, as seen in the lower diagram of Fig. 8.

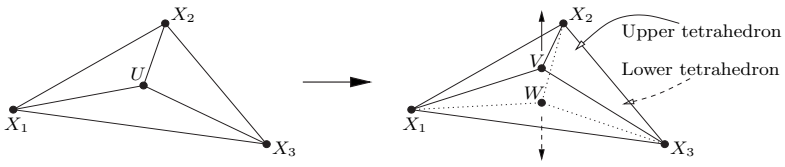

Fig. 7. Expanding a three-face cone into a region bounding two tetrahedra.

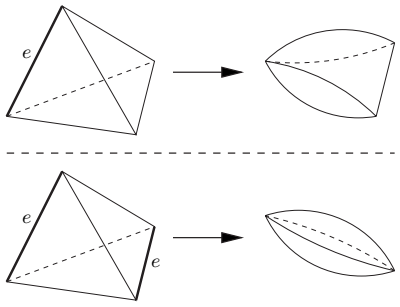

Fig. 8. Consequences of shrinking edge e to a single point.

- The tetrahedron contains edge e more than once, and e appears twice on a single face. Since the endpoints of e are distinct vertices, the face must have its edges e identified to form a cone as illustrated earlier in Fig. 3. This means that, unless the face is $\Delta X_1X_2X_3$, there is a corresponding face in T that also forms a cone, in contradiction to Lemma 3.5.

  Thus edge e belongs to face $\Delta X_1X_2X_3$ more than once, forming a one-face cone in T' but not in T. Without loss of generality, let directed edge $\vec{e} = \overrightarrow{X_1X_2} = \overrightarrow{X_3X_2}$ in the expanded triangulation T'. This implies that $\overrightarrow{X_1X_2} = \overrightarrow{X_3X_2}$ in the original triangulation T.

  To avoid a two-face cone that contradicts Lemma 3.6, these edges must also be identified with $\overline{UX_2}$ in some direction. To avoid a one-face cone that contradicts Lemma 3.5, this direction must be $\overrightarrow{X_1X_2} = \overrightarrow{X_3X_2} = \overrightarrow{X_2U}$. Moving back to the expanded triangulation T', the old edge $\overrightarrow{X_2U}$ is pulled apart into two distinct edges $\overrightarrow{X_2V}$ and $\overrightarrow{X_2W}$. However, one of these identifications with $\vec{e}$ will be preserved; without loss of generality, suppose that $\vec{e}$ is identified with the new directed edge $\overrightarrow{X_2V}$. This gives $\vec{e} = \overrightarrow{X_1X_2} = \overrightarrow{X_2V}$ in T', contradicting the fact that the endpoints of e are distinct.

2. We eliminate each football with four bigon faces, using a combination of the following two operations. The order in which these operations are performed is important, and will be discussed shortly. In the meantime however, we describe each operation in detail.

   (i) *Flattening bigon faces to edges*. This procedure is illustrated in Fig. 9.

   Flattening a bigon face has no effect upon the underlying 3-manifold unless the two edges of the bigon are identified. In this case the bigon forms either a 2-sphere or a projective plane.

   - If the bigon forms a 2-sphere, flattening the bigon has the effect of cutting the manifold along this sphere and filling the resulting boundaries with solid

Fig. 9. Flattening a bigon face to a single edge.

Fig. 10. Flattening a bigon pillow to a single bigon face.

balls. Since the original manifold is $\mathbb{P}^2$-irreducible, the resulting manifold is a disconnected union of the original manifold and a 3-sphere. In this case we simply toss away the extra 3-sphere (possibly reducing the overall number of tetrahedra) and keep the structure that corresponds to the original manifold.

- If the bigon forms a projective plane then this plane has just one vertex, one edge and one face, and is therefore embedded in the underlying 3-manifold. The plane cannot be two-sided, since the underlying manifold is $\mathbb{P}^2$-irreducible. Therefore the manifold contains an embedded one-sided projective plane, and is thus of the form $M\#\mathbb{R}P^3$ for some manifold M. From $\mathbb{P}^2$-irreducibility again it follows that M is trivial and the original manifold is simply $\mathbb{R}P^3$. This however can be triangulated with two tetrahedra [6], contradicting the requirement that the original triangulation T be minimal.

(ii) *Flattening bigon pillows to bigon faces*. This procedure is illustrated in Fig. 10.

Flattening a bigon pillow cannot change the underlying 3-manifold unless the two faces of the pillow are identified. If this is the case then the bigon pillow represents the entire 3-manifold, since it forms a complete closed structure. Running through the four possible ways in which the upper bigon face can be identified with the lower, we see that the 3-manifold must be either S^3 or $\mathbb{R}P^3$. Both manifolds can be triangulated using ≤ 2 tetrahedra [6], in contradiction to the requirement that triangulation T be minimal.

Having established that neither operation changes the underlying 3-manifold, we can eliminate all footballs using the following algorithm:

- The overall aim of the algorithm is to eliminate each football with four bigon faces by (i) flattening two of the four bigon faces to edges, thereby reducing the football to a bigon pillow, and then (ii) flattening this bigon pillow to a bigon face, removing the football completely.
- The algorithm consists of a single operation that is repeated over and over. Each time we begin the operation, all remaining footballs have either three or four bigon faces—this is certainly true at the very beginning of the algorithm, and it will be seen that this remains true each time the operation is applied.
- The operation is as follows. Locate a bigon face B that belongs to one of the remaining footballs, where B is not identified with some other face of the same football.

  We flatten this bigon B to an edge. This will reduce the number of faces of the chosen football by one. If B was identified with a bigon face of some different football, this other football will also have its face count reduced by one. Otherwise B was identified with a face of a triangular pillow, which will thereby lose one of its bigon sides.

  At this point we potentially have one or two footballs with only two bigon faces, i.e., we potentially have one or two bigon pillows. If this occurs, the bigon

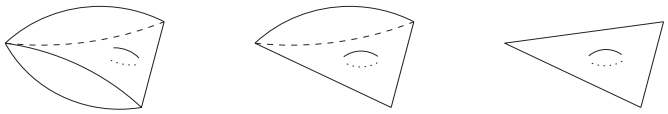

Fig. 11. Remaining triangular pillows with or without bigon sides.

pillows are immediately removed by flattening them to faces. All remaining footballs will therefore have either three or four bigon faces as required, and the operation can begin again.

- We continue until the operation can no longer be performed. If any footballs still remain, they must be four-face footballs whose four bigon faces are identified in two pairs. However, such a football is impossible—we could flatten the first bigon to an edge, reducing the football to a bigon pillow whose two remaining faces are identified, and this was seen to be impossible in step (ii) above.

Once this algorithm has finished, the footballs will have disappeared completely. In addition, some of the triangular pillows with bigon sides might have had their sides flattened to edges. The possible new shapes that these triangular pillows might take can be seen in Fig. 11.

3. We flatten all remaining bigon faces to edges. This simplifies the triangular pillows so that each pillow has just three edges and two triangular faces, as seen in the rightmost diagram of Fig. 11.
4. We flatten all triangular pillows to faces. This procedure is illustrated in Fig. 12.

   The only way in which flattening a triangular pillow can change the underlying 3-manifold is if the upper and lower faces of the pillow are identified. In such a case, the pillow must represent the entire 3-manifold (since it forms a closed structure), and a quick run through the six possible rotations and reflections of the triangle shows that this 3-manifold is either S^3 or $L_{3,1}$. Both manifolds can be triangulated using ≤ 2 tetrahedra as seen in [6], contradicting the claim that the original triangulation T is minimal.

At this point all of our unusual shapes have been flattened away, and we are left with only tetrahedra. This gives us a new triangulation T'' of the original 3-manifold, where each tetrahedron of T' that contained edge e has been removed. Since triangulation T' contained precisely $n + 2$ tetrahedra and edge e was chosen to belong to ≥ 3 distinct tetrahedra, it follows that our final triangulation T'' contains at most $n - 1$ tetrahedra. Therefore the original triangulation T could not have been minimal, and the proof is complete. □

To close this section, we examine some consequences of the previous results when applied to tori within a triangulation.

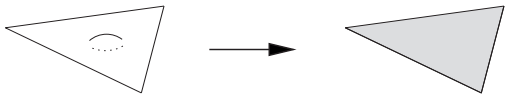

Fig. 12. Flattening a triangular pillow to a single face.

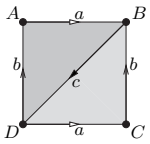

Fig. 13. Two faces joined together to form a torus.

Corollary 3.8. *Let T be a closed minimal $\mathbb{P}^2$-irreducible triangulation containing ≥ 3 tetrahedra. Suppose that two distinct faces of T are joined along their edges to form a torus, as illustrated in Fig.* 13. *Then all three edges of this torus are distinct, i.e., no two of these edges are identified in the overall triangulation.*

Proof. It is clear that all three edges of the torus cannot be identified together, since this would break either Lemma 3.4 or 3.5. Suppose then, without loss of generality, that edges a and b are identified.

If directed edges a and $-b$ are identified then each face has two edges identified to form a cone, in contradiction to Lemma 3.5. It follows that directed edges a and b are identified. This however creates a two-face cone centred about vertex D, with internal edges $a = b$ and c (where directed edge $a = b$ points away from the centre of the cone, and directed edge c points towards the centre of the cone).

Since all three edges of the torus are not identified, it follows that the internal edges of this cone are distinct. This contradicts Lemma 3.6 and concludes the proof. □

Corollary 3.9. *Let T be a closed minimal $\mathbb{P}^2$-irreducible triangulation containing ≥ 3 tetrahedra. Suppose that two distinct faces of T are joined along their edges to form a torus, as illustrated previously in Fig.* 13. *Moreover, suppose that some other face of T contains at least two of the edges a, b, c. Then all three faces belong to a single tetrahedron.*

Proof. Denote the third face by F. Without loss of generality, suppose that F contains directed edges a and b. Noting from Corollary 3.8 that a and b are distinct, Fig. 14 shows the possible ways in which these edges may be oriented within face F.

In case (i), face F forms a two-face cone with face ΔDCB from the torus (with vertex C at the centre of the cone). Similarly, in case (ii), face F forms a two-face cone with face ΔBAD. Lemma 3.6 provides a contradiction in both cases.

For case (iii), we obtain a cone from all three faces. Vertex D sits at the centre of this cone, with directed edges a and b running away from the centre and c running towards the centre. Since a, b and c are all distinct from Corollary 3.8, we see from Lemma 3.7 that all three faces belong to a single tetrahedron. Case (iv) is handled likewise. □

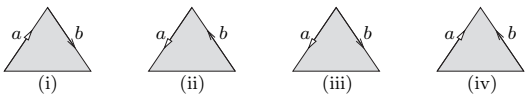

Fig. 14. Possible cases for directed edges a and b within the third face F.

4. Face-Pairing Graphs

Recall from Definition 2.1 that the *face-pairing graph* of a triangulation is a 4-valent graph describing which tetrahedron faces are identified with which. In this section we use an analysis of face-pairing graphs to obtain our first round of improvements to the census algorithm.

The earlier paper [6] likewise uses face-pairing graphs to improve the census algorithm; many of the results proven here are generalisations of these earlier theorems. Some core results of [6] are reviewed in Section 4.1, where we also introduce the recurring concepts of chains and layered solid tori.

Following this, Section 4.2 builds on this work to obtain new results and algorithmic improvements. In this section we obtain a series of properties that must be satisfied by the face-pairing graph of a closed minimal $\mathbb{P}^2$-irreducible triangulation. Using these results, we can improve Algorithm 2.2 by identifying graphs that do not satisfy these properties and eliminating them after step 1 of the algorithm (enumeration of graphs), without carrying them through to step 2 (processing of graphs).

Section 4.3 gives an indication of the usefulness of these results. In particular, it examines all possible face-pairing graphs on ≤ 10 vertices and identifies how many of these graphs can be eliminated using the results of Section 4.2, thereby offering a rough estimate of the processing time that can be saved.

4.1. *Previous Results*

Before summarising the key results of [6], some additional terminology is required. Most of the major results of [6] (and all of the graph-related results of this paper) involve subgraphs called *chains*.

Definition 4.1 (Chain). A *chain* of length k is a sequence of k double edges joined end-to-end in a linear fashion, as illustrated in Fig. 15. A chain of length k must link together $k + 1$ distinct vertices (i.e., no vertex may be repeated).

A *one-ended chain* of length k is a chain of length k with a loop added to one end, as illustrated in Fig. 16. Likewise, a *double-ended chain* of length k is a chain of length k with loops added to both ends.

Wherever this paper refers to a one-ended or double-ended chain, chains of length zero are also included. In particular, any statement involving one-ended chains is assumed to include the zero-length case of a single loop.

A key property of chains is that every vertex has degree four except for the two vertices at each end. This means that, within a 4-valent face-pairing graph, a chain is almost entirely self-contained—it may only join to the remainder of the graph at these two end vertices.

Fig. 15. A chain of length five.

Fig. 16. A one-ended chain of length four.

The value of this is seen when we translate to the language of triangulations. Here a chain of length k becomes a sequence of $k + 1$ tetrahedra joined together, with almost all of these tetrahedra meeting other tetrahedra from the chain along all four faces. The only tetrahedra with any spare faces for connecting this structure to the rest of the triangulation are the two tetrahedra at either end of the sequence. This constrains the possibilities enough for us to draw interesting conclusions about the rotations and reflections used when identifying tetrahedron faces together.

One-ended chains are even more constrained, since every vertex but one has degree four. This allows the corresponding tetrahedron-based structures to be identified even more precisely, as seen in [6] and reproduced in Theorem 4.4 below. To understand these structures, we must introduce the concept of layering.

Definition 4.2 (Layering). Consider a triangulation with boundary B, and let e be an edge of this boundary that lies between two distinct boundary faces. To *layer on edge e* is to perform the following operation.

We obtain a new tetrahedron Δ, and lay two adjacent faces of Δ directly onto the two boundary faces surrounding e. These two original faces surrounding e become internal faces of the triangulation, and the remaining two faces of Δ become new boundary faces. Likewise, the original boundary edge e becomes internal, making way for a new boundary edge f on the far side of tetrahedron Δ. This procedure is illustrated in Fig. 17.

It should be observed that performing a layering does not alter the underlying 3-manifold of a triangulation. What it does change is the curves made by the boundary edges of the triangulation, since in general the new boundary edge f does not represent the same curve as the original boundary edge e.

One of the more frequent examples of layering that can be found in a census of triangulations is in the construction of *layered solid tori*. These well-structured triangulations of solid tori have been discussed in depth by Jaco and Rubinstein [12], [13], and similar structures involving special spines have been described by Matveev [20]. In the context of a census of 3-manifold triangulations, they are parameterised in detail in [5].

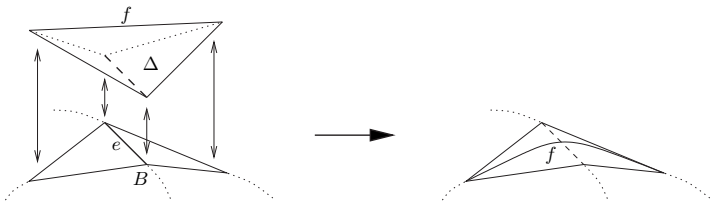

Fig. 17. Performing a layering.

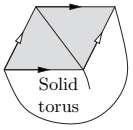

Fig. 18. The boundary of a layered solid torus.

We do not require such detail here, and so a full definition of a layered solid torus is omitted—the reader is referred to the aforementioned references for details. For this paper the following summary is sufficient.

Definition 4.3 (Layered Solid Torus). A *standard layered solid torus* is a triangulation of the solid torus obtained through successive layerings upon the boundary of T_1, where T_1 is the one-tetrahedron triangulation of the solid torus.

There is no limit upon the number of layerings—there may be a large number of layerings or there may be none at all. There is also no constraint upon which particular boundary edges are used for performing the layerings.

As a result, an infinite variety of different standard layered solid tori can be constructed. Each of these solid tori has two boundary faces, arranged to form a torus as illustrated in Fig. 18.

Equipped with this terminology, we are now in a position to recount the core results proven by the author in [6]. The first of these results describes the tetrahedron-based structures corresponding to one-ended chains. This theorem forms a starting point for many later results, including all of the results of Section 4.2 in this paper.

Theorem 4.4. *Let G be the face-pairing graph of a closed minimal $\mathbb{P}^2$-irreducible triangulation containing ≥ 3 tetrahedra. Suppose that G contains a one-ended chain. Then the tetrahedra corresponding to the vertices of this one-ended chain form a standard layered solid torus within the triangulation.*

Recall that every vertex of a one-ended chain has degree four except for a single end vertex, which has degree two. This allows a one-ended chain to be attached to the remainder of a face-pairing graph by two edges; in the language of triangulations, this corresponds to attaching a layered solid torus to the remainder of the triangulation along its two boundary faces.

In Section 3.2 of [6], the author proves a series of constraints on the face-pairing graphs of minimal triangulations. These constraints are bundled together into the single theorem below.

Theorem 4.5. *Let G be the face-pairing graph of a closed minimal $\mathbb{P}^2$-irreducible triangulation containing ≥ 3 tetrahedra. Then G does not contain any of the following subgraphs*:

(i) *A triple edge.*
(ii) *A triangle with a double edge and a one-ended chain connected to the opposite vertex.*

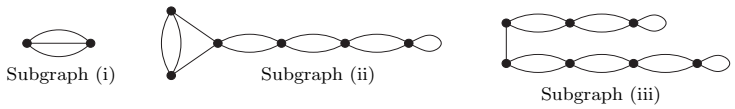

Fig. 19. Examples of prohibited subgraphs from earlier results.

(iii) *A single edge with a one-ended chain connected to each vertex, where this single edge does not form part of a double edge.*

Examples of these three prohibited types of subgraph are illustrated in Fig. 19. It will be seen in Section 4.2 that several of the new results proven in this paper can be viewed as generalisations of the various components of the theorem above.

4.2. *Elimination of Graphs*

As indicated at the end of Section 2, one method of improving the census algorithm involves proving constraints that must be satisfied by the face-pairing graphs of closed minimal $\mathbb{P}^2$-irreducible triangulations. Results of this type allow us to identify graphs that can be discarded immediately without any processing.

Theorem 4.5 illustrates earlier results of this type from [6]. In this section we generalise these earlier results, proving additional constraints in Theorems 4.6–4.8. In the following section we present empirical data that measures how useful these results might be in a real census.

It should be noted that in the orientable case, Martelli was previously aware of simpler variants of some of these constraints.[1] The results presented here are more general, and apply to both orientable and non-orientable triangulations.

Theorem 4.6. *Let G be the face-pairing graph of a closed minimal $\mathbb{P}^2$-irreducible triangulation containing ≥ 3 tetrahedra. Suppose that G contains a one-ended chain, and suppose that some edge of G joins this chain to another double edge as illustrated in Fig.* 20. *Then one of the following statements must be true*:

(i) *G contains a longer one-ended chain, incorporating the original one-ended chain, edge and double edge previously discussed.*
(ii) *The original one-ended chain is joined to both ends of a new chain of length two, where the original double edge belongs to this new chain.*
(iii) *There is some vertex V not belonging to either the original chain or the double edge, where V is joined by edges to the end of the original chain and to both ends of the double edge.*

These three possibilities are illustrated in Fig. 21, *where the original one-ended chain, edge and double edge are drawn in solid lines, and any new vertices or edges are drawn in dashes.*

[1] This was noted in a private communication with the author in November 2003.

Fig. 20. Initial conditions for Theorem 4.6.

Before embarking on a proof, it is worth noting the following points about Theorem 4.6.

- Whilst a little unwieldy, all three options presented in the theorem statement are necessary. Graphs corresponding to options (i) and (iii) appear in very early levels of the orientable census; illustrated examples can be found in the six-tetrahedron census results of Matveev [20]. Option (ii) is only required in the non-orientable case, where an 11-tetrahedron example has been found that necessitates it (the manifold is $\mathrm{SFS}(\dot{D} : (3, 1)) \cup \textit{Gieseking}$; see Section 6.1 for notation).
- This theorem generalises some of the earlier results of [6]. In particular, this new theorem covers both cases (ii) and (iii) of the earlier Theorem 4.5.

Proof of Theorem 4.6. Suppose the face-pairing graph G contains a one-ended chain attached to some other double edge, as described in the theorem statement. Assign the label V_1 to the vertex at the non-loop end of the chain, and assign the labels V_2 and V_3 to the endpoints of the other double edge, where vertices V_1 and V_2 are joined. This is illustrated in the leftmost diagram of Fig. 22.

Since vertex V_1 only has degree three in this diagram, there is an edge meeting V_1 that is not yet accounted for. This edge cannot join any other vertex of the chain, since these other vertices already have full degree four. Likewise, it cannot be a loop joining V_1 to itself since this would raise the degree of V_1 to five. If the edge runs from V_1 to V_2 then we have option (i) of the theorem statement, and if it runs from V_1 to V_3 then we have a situation that Theorem 4.5 proves impossible.

The only case remaining is where V_1 is joined to some new vertex not previously considered. Label this new vertex V_4; this is illustrated in the rightmost diagram of Fig. 22.

We translate now to the language of triangulations. Suppose that vertices V_1, V_2, V_3 and V_4 of the graph correspond to tetrahedra Δ_1, Δ_2, Δ_3 and Δ_4 of the triangulation respectively. From the graph we can draw the following conclusions:

- Two faces of tetrahedron Δ_1 form the boundary torus of a standard layered solid torus (this follows from applying Theorem 4.4 to the one-ended chain).
- One of these faces is identified with some face of Δ_2; the other is identified with some face of Δ_4.
- Two other faces of Δ_2 are each identified with faces of Δ_3.

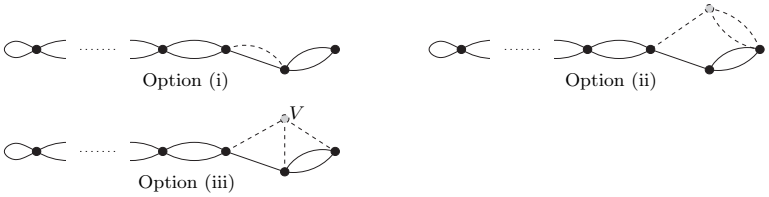

Fig. 21. Possible resolutions for Theorem 4.6.

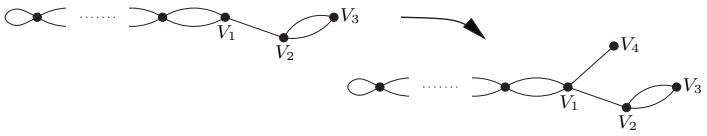

Fig. 22. Identifying graph vertices in Theorem 4.6.

Part of this configuration is illustrated in Fig. 23. The layered solid torus sits beneath the figure, with its two boundary faces ΔABD and ΔBDC drawn in bold. Note that edges $\overline{AB}$ and $\overline{DC}$ are identified, and likewise edges $\overline{AD}$ and $\overline{BC}$ are identified. Tetrahedron Δ_4 is then attached to face ΔABD, and uses vertices A, B, D and E. Likewise, Δ_2 is attached to face ΔBDC and uses vertices B, C, D and F. Through the symmetry of the torus we may assume that faces ΔBDF and ΔBCF are joined to tetrahedron Δ_3; in the rightmost diagram Δ_3 is placed upon face ΔBDF, using vertices B, D, F and G.

It remains then to identify ΔBCF (shaded in the diagram) with one of the three remaining faces of Δ_3. Although there are 18 possibilities (three faces, with six rotations and reflections each), we can quickly cut this number down.

Consider $\overline{BC}$; the final mapping must identify this edge with one of the edges of Δ_3. Noting that the distinct faces ΔABD and ΔBCD form a torus, Corollary 3.8 shows that $\overline{BC}$ cannot be identified with $\overline{BD}$. Corollary 3.9 then shows that $\overline{BC}$ cannot be identified with $\overline{BF}$ or $\overline{FD}$, since this would require ΔABD, ΔBDC and ΔBDF to belong to a single tetrahedron. Likewise, Corollary 3.9 also shows that $\overline{BC}$ cannot be identified with $\overline{BG}$ or $\overline{GD}$, since otherwise ΔABD, ΔBDC and ΔBDG would belong to a single tetrahedron.

The only remaining possibility is that $\overline{BC}$ is identified with $\overline{FG}$. Taking into account the two possible orientations of the edge and the two possible target faces ΔBFG and ΔDFG, this leaves us with the following four cases. In each case, the order in which we write the vertices of the two identified faces indicates which particular rotation or reflection is used (i.e., if we claim that faces ΔPQR and ΔXYZ are identified, this implies that vertices P, Q and R are identified with X, Y and Z respectively).

- ΔBCF is identified with ΔFGB (with directed edge $\overrightarrow{BC}$ identified with $\overrightarrow{FG}$).
  In this case we have directed edges $\overrightarrow{BF}$ and $\overrightarrow{FB}$ identified, which is impossible.
- ΔBCF is identified with ΔGFB (with directed edge $\overrightarrow{BC}$ identified with $\overrightarrow{GF}$).
  This forces the identification of directed edges $\overrightarrow{GB} = \overrightarrow{BF} = \overrightarrow{FC}$, as illustrated in the leftmost diagram of Fig. 24. Consider the three faces ΔABD, ΔDBG and

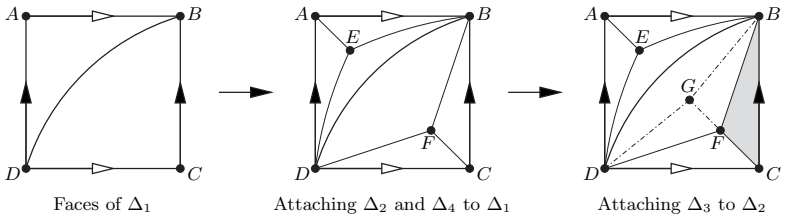

Fig. 23. Translating graph vertices to tetrahedra.

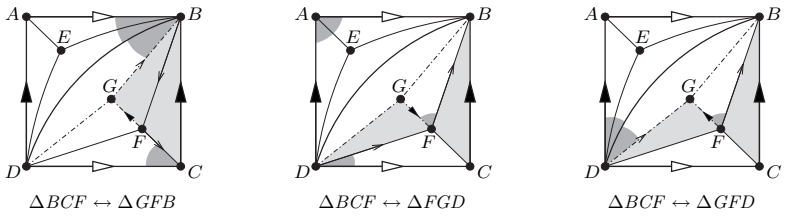

Fig. 24. Possibilities for the remaining join between Δ_2 and Δ_3.

ΔFCD. Because of the aforementioned edge identifications, these three faces come together to form a three-face cone—the three shaded arcs in the diagram mark the neighbourhood of the vertex at the centre of this cone.

Our aim then is to use Lemma 3.7. First we observe that all three faces are distinct, since ΔABD is already an interior face, and identifying ΔDBG with ΔFCD would create a triple edge in the face-pairing graph (which Theorem 4.5 states is impossible).

The three interior edges of the cone are $\overline{AB}$, $\overline{DB}$ and $\overline{GB} = \overline{BF} = \overline{FC}$. We already know from Corollary 3.8 that $\overline{AB}$ and $\overline{DB}$ are distinct; furthermore, since face ΔBCF contains both $\overline{GB}$ and the third edge of the original torus, Corollary 3.9 shows that $\overline{GB}$ is distinct from both $\overline{AB}$ and $\overline{DB}$ (otherwise ΔBCF, ΔABD and ΔBDC would belong to a common tetrahedron, an impossible situation).

Therefore we can apply Lemma 3.7 to the three-face cone described above. It follows that ΔABD, ΔDBG and ΔFCD belong to a single tetrahedron, which must be the upper-left tetrahedron Δ_4. Therefore vertices V_2 and V_3 each join to V_4 in the face-pairing graph, and we have case (iii) from the theorem statement.

- ΔBCF is identified with ΔFGD (with directed edge $\overrightarrow{BC}$ identified with $\overrightarrow{FG}$).

  As in the previous case, we seek to find and exploit a three-face cone. This time we have directed edge identifications $\overrightarrow{FB} = \overrightarrow{DF}$ and $\overrightarrow{GF} = \overrightarrow{CB} = \overrightarrow{DA}$, giving us a cone from faces ΔDAB, ΔCDF and ΔBFG. This is illustrated in the middle diagram of Fig. 24, with the shaded arcs once more representing a neighbourhood of the vertex at the centre of the cone.

  The three faces are distinct for the same reason as before (avoiding a triple edge in the face-pairing graph), and again Corollaries 3.8 and 3.9 show that the three internal edges of the cone are distinct. Thus Lemma 3.7 shows that ΔDAB, ΔCDF and ΔBFG belong to a common tetrahedron which must again be Δ_4, giving us case (iii) from the theorem statement as before.

- ΔBCF is identified with ΔGFD (with directed edge $\overrightarrow{BC}$ identified with $\overrightarrow{GF}$).

  In this final case the relevant directed edge identifications are $\overrightarrow{FB} = \overrightarrow{DG}$ and $\overrightarrow{FG} = \overrightarrow{CB} = \overrightarrow{DA}$. Again we find a three-face cone, this time using faces ΔADB, ΔBDG and ΔBFG.

  From here we diverge a little from previous cases. This time the three faces are distinct because the alternative would be to identify two faces of Δ_3, thereby adding a loop to V_3 in the face-pairing graph and contradicting case (iii) of Theorem 4.5. The three internal edges of the cone are distinct for the usual reasons, and so

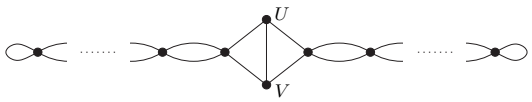

Fig. 25. Two adjacent vertices joined to two one-ended chains.

Lemma 3.7 shows that faces ΔADB, ΔBDG and ΔBFG all belong to a common tetrahedron.

As before, the only possible common tetrahedron is the upper-left tetrahedron Δ_4. Since both ΔBDG and ΔBFG belong to Δ_3, this adds a double edge from V_3 to V_4 in the face-pairing graph, giving us case (ii) from the theorem statement.

This concludes the list of possibilities, and the proof of Theorem 4.6 is complete. □

It should be noted that the face identifications in the final case above give a non-orientable structure, justifying our earlier claim that case (ii) is only necessary in the non-orientable case. A little further investigation shows that the manifold created by joining Δ_4, Δ_3, Δ_2 and the layered solid torus in this way is a bounded Seifert fibred space of the form $\mathrm{SFS}(\dot{D} : (p, q))$, where the orbifold $\dot{D}$ is the disc with half-reflector, half-regular boundary. See Section 6.1 for further discussion of manifolds of this type.

Our next two results extend cases (i) and (ii) of the old Theorem 4.5. Recall that case (i) of Theorem 4.5 outlaws a triple edge, and case (ii) can be viewed as outlawing a variant where one of these three edges is replaced with a one-ended chain. In Theorems 4.7 and 4.8, we likewise outlaw variants in which two and then all three of the edges in a triple edge are replaced with one-ended chains.

Since Theorems 4.7 and 4.8 are similar in structure, we begin by stating the two theorems without proof. Following this we present and prove Lemma 4.9, which contains logic common to both theorems. Finally we return to Theorems 4.7 and 4.8, using this new lemma to prove each in turn.

Theorem 4.7. *Let G be the face-pairing graph of a closed minimal $\mathbb{P}^2$-irreducible triangulation containing ≥ 3 tetrahedra. Then G cannot contain two distinct vertices U and V, where U and V are joined by an edge, and where U and V are each joined to the endpoints of two distinct one-ended chains. This prohibited situation is illustrated in Fig.* 25.

Theorem 4.8. *Let G be the face-pairing graph of a closed minimal $\mathbb{P}^2$-irreducible triangulation containing ≥ 3 tetrahedra. Then G cannot contain two distinct vertices U and V, where U and V are each joined to the endpoints of three distinct one-ended chains. This prohibited situation is illustrated in Fig.* 26.

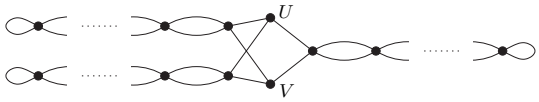

Fig. 26. Two vertices joined to three one-ended chains.

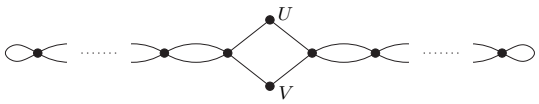

Fig. 27. Two vertices joined to two one-ended chains.

As indicated above, a certain amount of logic can be applied to both of these theorems simultaneously. In particular, we can obtain partial results by analysing the graph obtained by removing the central edge in Theorem 4.7, or by removing the third one-ended chain in Theorem 4.8.

Lemma 4.9. *Let G be the face-pairing graph of a closed minimal $\mathbb{P}^2$-irreducible triangulation containing ≥ 3 tetrahedra. Suppose that G contains two distinct vertices U and V, where U and V are each joined to the endpoints of two distinct one-ended chains as illustrated in Fig.* 27. *Then the tetrahedra corresponding to the vertices of this subgraph form the structure illustrated in Fig.* 28. *Layered solid tori corresponding to the two chains sit beneath the diagram*; *the boundary of the first solid torus is formed from faces ΔABE and ΔBEF, and the boundary of the second solid torus is formed from faces ΔBCF and ΔADE* (*note the directed edge identifications $\overrightarrow{AD} = \overrightarrow{CF}$, $\overrightarrow{AB} = \overrightarrow{EF}$, $\overrightarrow{BC} = \overrightarrow{DE}$ and $\overrightarrow{AE} = \overrightarrow{BF}$*). *The tetrahedron corresponding to graph vertex U is $ABDE$, and the tetrahedron corresponding to graph vertex V is $BCEF$* (*note that each of these tetrahedra shares a face with each of the aforementioned boundary tori*).

Proof. From Theorem 4.4 we know that each one-ended chain corresponds to a layered solid torus in the triangulation. Let Δ_V be the tetrahedron corresponding to vertex V in the face-pairing graph; since V is joined to each one-ended chain by an edge, it follows that Δ_V is joined to each layered solid torus along a face. By symmetry we may assume the leftmost diagram of Fig. 29, where the first solid torus has boundary faces ΔABE and ΔBEF, the second solid torus has boundary faces ΔBCF and ΔCFG, and tetrahedron Δ_V has vertices B, C, E and F. As in the theorem statement, both solid tori sit beneath the diagram.

Similarly, let Δ_U be the tetrahedron corresponding to the graph vertex U. Since U is joined to each one-ended chain, Δ_U must be attached to both faces ΔABE and ΔCFG. In the rightmost diagram of Fig. 29 we see it attached to face ΔCFG, where Δ_U uses vertices C, F, G and H.

It remains to identify ΔABE with one of the three remaining faces of Δ_U (specifically ΔCHF, ΔFHG or ΔGHC). Most possible identifications can be eliminated by examining

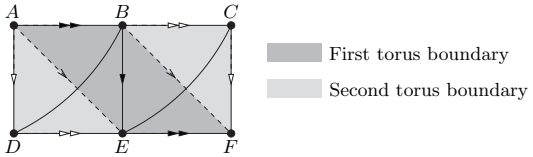

Fig. 28. The configuration of tetrahedra obtained in Lemma 4.9.

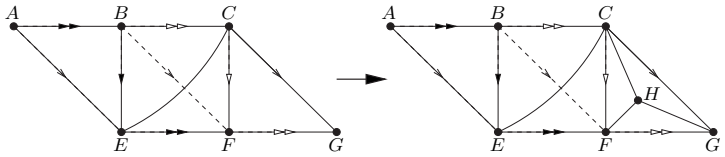

Fig. 29. Joining two tetrahedra to two layered solid tori.

edge $\overline{AE}$, which (through its existing identification with $\overline{CG}$) forms part of the torus $BCGF$. Corollary 3.8 shows that $\overline{AE}$ cannot be identified with $\overline{CF}$ or $\overline{FG}$, and Corollary 3.9 shows that $\overline{AE}$ cannot be identified with $\overline{CH}$, $\overline{FH}$ or $\overline{GH}$. Finally, directed edge $\overrightarrow{AE}$ cannot be identified with $\overrightarrow{GC}$ since it is already identified with the reverse edge $\overrightarrow{CG}$.

The only possible identification of faces that does not involve one of these impossible edge matchings is for ΔABE to be identified with ΔCHG (in particular, with directed edge $\overrightarrow{AE}$ mapping to $\overrightarrow{CG}$). This gives us the configuration described in the lemma statement, and the proof is complete. □

Having achieved the partial results described in Lemma 4.9, we can apply these results in different ways to obtain individual proofs of Theorems 4.7 and 4.8.

Proof of Theorem 4.7. Suppose the face-pairing graph G contains a subgraph as illustrated in Fig. 25. From Lemma 4.9 it is clear that the triangulation contains the structure illustrated in Fig. 28, where one additional face of tetrahedron $\Delta_U = ABDE$ is identified with one additional face of tetrahedron $\Delta_V = BCEF$.

By symmetry we may assume the face of Δ_U is ΔBDE. This identification of faces may cause edge $\overline{BE}$ to be identified with some other edge in the diagram. However, since $\overline{BE}$ belongs to the torus $ABEF$, a little investigation shows that Corollaries 3.8 and 3.9 do not allow $\overline{BE}$ to be identified with any other edge. It follows that the two identified faces must be ΔBDE and ΔBCE, with directed edge $\overrightarrow{BE}$ mapping to itself.

This however implies that edges $\overline{DE}$ and $\overline{CE}$ become identified, so that face ΔCEF contains two distinct edges of the second torus (the torus with faces ΔBCF and ΔADE). Applying Corollary 3.9 once more gives us a contradiction and the proof is complete. □

Proof of Theorem 4.8. Suppose the face-pairing graph G contains a subgraph as illustrated in Fig. 26. This time Lemma 4.9 shows that the tetrahedra form the structure illustrated in Fig. 28, where an additional layered solid torus T is attached to one face of tetrahedron $\Delta_U = ABDE$ (call this face F') and one face of tetrahedron $\Delta_V = BCEF$ (call this face F'').

Attaching a layered solid torus to faces F' and F'' will cause the three edges of F' to be identified with the three edges of F''. By symmetry we may assume $F' = \Delta BDE$. As in the previous proof, edge $\overline{BE}$ may not be identified with any other edge in the diagram. It therefore follows that $F'' = \Delta BCE$, where the new solid torus gives directed edge identifications $\overrightarrow{DE} = \overrightarrow{BC}$ and $\overrightarrow{BD} = \overrightarrow{CE}$. This is illustrated in Fig. 30, where the first

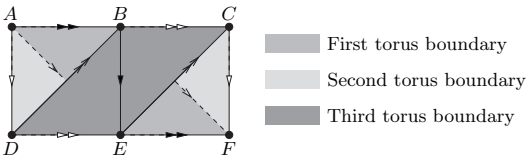

Fig. 30. The configuration of tetrahedra obtained in the proof of Theorem 4.8.

two solid tori sit beneath the diagram as before, and the new solid torus sits above the diagram (but is not drawn).

We conclude the proof with an examination of vertex links. Note that each layered solid torus is a one-vertex structure, where the link of this single vertex is a disc [5], [12]. Moreover, our entire construction as seen in Fig. 30 has only one vertex, since each layered solid torus has only one vertex, and since vertices A, B, C, D, E and F are all identified.

An examination of the link of this single vertex shows that it is not a disc—instead the vertex link in our construction is a punctured torus. There is no way that we can complete this triangulation to give a 2-sphere vertex link (as required for a closed 3-manifold triangulation), and so the scenario described is impossible. □

4.3. *Experimental Results*

As outlined in Section 2.2, we can use Theorems 4.6–4.8 to improve the generation of triangulations in the census algorithm. Specifically, we refuse to process any face-pairing graphs that do not satisfy the constraints of these theorems.

Since the generation of face-pairing graphs takes negligible time, a very rough estimate suggests that if we can eliminate $k\%$ of graphs in this way, we should save approximately $k\%$ of the overall running time. We therefore focus our efforts on measuring how many graphs are eliminated in comparison with the total number of graphs available.

An initial set of figures is provided in Table 2 for a sense of scale. The "Total" column measures the total number of connected 4-valent graphs on the given number of vertices. The "Census" column indicates how many of these in fact appear as face-pairing graphs of

Table 2. Total numbers of face-pairing graphs on ≤ 10 vertices.

Vertices	Total	Census	Not used
3	4	2	2
4	10	4	6
5	28	8	20
6	97	14	83
7	359	29	330
8	1,635	66	1,569
9	8,296	143	8,153
10	48,432	404	48,028

Table 3. Elimination of face-pairing graphs on ≤ 10 vertices.

	Graphs eliminated							
Vertices	Old results		New results		All results		Left over	
3	2	(100%)	1	(50%)	2	(100%)	0	(0%)
4	6	(100%)	4	(67%)	6	(100%)	0	(0%)
5	16	(80%)	14	(70%)	19	(95%)	1	(5%)
6	58	(70%)	60	(72%)	74	(89%)	9	(11%)
7	221	(67%)	238	(72%)	290	(88%)	40	(12%)
8	997	(64%)	1,116	(71%)	1,343	(86%)	226	(14%)
9	4,930	(60%)	5,834	(72%)	6,904	(85%)	1,249	(15%)
10	27,681	(58%)	34,452	(72%)	40,353	(84%)	7,675	(16%)

closed minimal $\mathbb{P}^2$-irreducible triangulations in the orientable or non-orientable census, and the "Not used" column counts the graphs that do not appear (so the second column is the sum of the final two).

It should be noted that, as the number of vertices grows large, the number of graphs that actually appear in the census becomes negligible—by the time we reach 10 vertices, the number of census graphs is under 1%. It is clear that a great deal of running time could be saved if we were able to eliminate all of these unused graphs through theorems such as those proven here.

Alas we do not achieve our dream of eliminating every unnecessary graph, but we do manage to eliminate well over 80% of them. Table 3 shows how well our new theorems perform. The "Old results" column counts the number of graphs eliminated using the earlier Theorem 4.5 from [6], and the "New results" column counts the number of graphs eliminated using the newly proven Theorems 4.6–4.8. The column labelled "All results" combines these figures to count the total number of graphs that can be removed using any of the old or new results, and the "Left over" column counts the number of graphs that do not appear in the census but were not eliminated at any stage.

Although the new results appear weak for very small numbers of vertices, they quickly outperform the earlier results of [6]. By the time we reach 10 tetrahedra they have climbed to a steady level of eliminating 71–72% of all unused graphs, and when combined with the earlier results they eliminate well over 80%. It should be noted that the percentages in this table are with respect to the total number of unused graphs (e.g., 48,028 graphs for 10 vertices). However, since the number of census graphs is negligible for larger numbers of vertices, we should still expect to save ourselves well over 80% of the total running time.

Looking forward, the data to focus on is the final "Left over" column. This lists the number of graphs that *could* have been eliminated (since they do not appear in the census), but were not eliminated by any of our old or new theorems. Although small with respect to the total number of graphs, these figures are not negligible—it is clear that with further work we could still make significant improvements to the running time. For reference, Fig. 31 lists the 10 unused graphs on ≤ 6 vertices that are not eliminated by any of theorems discussed here.

As a final note, Table 4 shows how each of the three new theorems performs individually. It is clear that Theorem 4.6 is by far the most useful of the three, though in a census

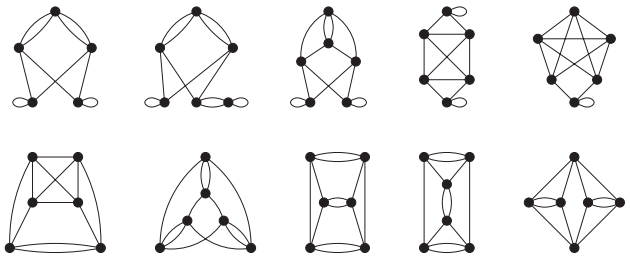

Fig. 31. Unused graphs on ≤ 6 vertices that are not eliminated.

that may require months or years of processing time, the contributions of Theorems 4.7 and 4.8 should not be ignored.

5. Tracking Vertex and Edge Links

We now put face-pairing graphs behind us and move on to our second round of algorithmic improvements. Here we aim to prune the search tree by tracking partially constructed vertex and edge links, with the help of a modified union-find algorithm.

Section 5.1 gives an overview of how the census algorithm is modified, including which branches of the search tree we aim to prune, and outlines some of the implementation difficulties that arise. In Section 5.2 we introduce the classical union-find algorithm, and in Section 5.3 we describe how it can be modified to address the difficulties previously raised. Finally, experimental data is presented in Section 5.4 to show just how well these improvements perform in practice.

5.1. *Pruning the Search Tree*

Recall Algorithm 2.2, in which we split the generation of triangulations into the enumeration and then processing of face-pairing graphs. The processing stage consumes

Table 4. Frequency of prohibited subgraphs on ≤ 10 vertices.

	Graphs eliminated			
Vertices	Theorem 4.6	Theorem 4.7	Theorem 4.8	Combined
3	1	0	0	1
4	4	1	0	4
5	13	2	1	14
6	56	5	2	60
7	227	13	5	238
8	1,083	46	14	1,116
9	5,730	170	47	5,834
10	34,059	746	176	34,452

virtually all of the running time, and involves a lengthy search through all possible rotations and reflections by which tetrahedron faces might be identified according to a particular face-pairing graph.

It is natural to implement this search as a recursive (depth-first) search with backtracking. If we have n tetrahedra then we have $2n$ pairs of faces to join; we first choose one of the six possible rotations and reflections for the first pair, then we choose one for the second pair and so on. If at any stage we find ourselves in an impossible situation, or if the six options for a particular pair are exhausted, we backtrack to the previous pair of faces, move on to the next rotation or reflection for that earlier pair, and then push forwards again. In this way all 6^{2n} possible rotations and reflections are covered.

A side-effect of such a backtracking algorithm is that at each stage we have a partially constructed triangulation—some tetrahedron faces are identified in pairs, and some faces are still waiting for their gluing instructions. If we can identify from a partially constructed triangulation that the finished triangulation cannot belong to our census, we can avoid any deeper searching and backtrack immediately. This is what is meant by "pruning a branch of the search tree". Note that the earlier we prune, the better—if we can identify an unwanted triangulation after joining together k pairs of faces, we remove 6^{2n-k} potential triangulations from the search.

This explains the mechanics of pruning; the question remains however of how to detect an unwanted triangulation when only some of the tetrahedron faces have been joined together. Fortunately several of the results of Section 3 are useful in this regard, since they refer only to small regions within a triangulation (e.g., single faces or single edges). The specific results we use include:

- If an edge link has been completely constructed (i.e., the edge does not belong to any faces that are still not joined to their partners), then that edge must have degree ≥ 3, and if it lies on three distinct tetrahedra then it must have degree > 3 (Lemma 3.3).
- No vertex link may be completely constructed unless the entire triangulation is finished, since otherwise we will obtain a ≥ 2-vertex triangulation (Lemma 3.1).
- No face may have two of its edges identified to form a cone, and no face may have all three edges identified (Lemmas 3.5 and 3.4).
- No edge may be joined to itself in reverse, and no vertex may have a non-orientable link (required for a 3-manifold triangulation).

Whilst the following two results are global rather than local properties, we add them to the list since they will prove easy to test within the infrastructure that we develop. Both results use the fact that each pairing of tetrahedron faces introduces at most three new edge identifications and three new vertex identifications.

- If we have joined k pairs of tetrahedron faces, the number of distinct vertex classes must be $\leq 1 + 3(2n - k)$ (Lemma 3.1).
- If we have joined k pairs of tetrahedron faces, the number of distinct edge classes must be $\geq n + 1$ and $\leq n + 1 + 3(2n - k)$ (Corollary 3.2).

All of the properties described above can be tested by examining edge and vertex links within the partially constructed triangulation, i.e., by examining which individual tetrahedron edges or vertices come together to form a single edge or vertex of the overall

triangulation. The simplest way of doing this is to group the individual tetrahedron edges and vertices into equivalence classes, where an equivalence class represents all edges or vertices that are identified together.

We could potentially recreate these equivalence classes each time we test our pruning constraints. However, this would be far too slow—the creation of equivalence classes could take up to linear time in n, and this would need to be done every time we joined a pair of tetrahedron faces together or pulled them apart again. A preferable solution is to modify the equivalence classes dynamically as we modify the triangulation, in the hope that only a small amount of work needs to be done at each step to keep the two in sync.

The question still remains of how to implement our equivalence classes. It is typically seen in computationally intensive problems such as this that a careful choice of underlying data structures and algorithms can have a great effect upon the overall running time. This choice is the primary focus of Sections 5.2 and 5.3. In the meantime, we examine the specific operations that our implementation of equivalence classes must support:

(i) *Merging together two equivalence classes*: This occurs when a new pair of tetrahedron faces are joined (specifically, each join results in ≤ 3 merges of edge classes and ≤ 3 merges of vertex classes).
(ii) *Splitting equivalence classes*: This occurs when we backtrack and pull a pair of tetrahedron faces apart. As a result we must undo whatever merges were performed in the previous step.
(iii) *Testing for equivalence*: We must be able to identify whether two tetrahedron edges or vertices belong to the same equivalence class (this is required when testing for conditions such as faces with multiple edges identified).
(iv) *Testing for completeness*: We need to identify whether an edge or vertex link has been completely constructed, i.e., does not involve any faces not yet joined to their partners (this is required for the one-vertex test and the edge degree tests).
(v) *Measuring class sizes*: We must be able to count the total number of edges in an equivalence class (this is required for the edge degree tests).
(vi) *Tracking orientation*: We must keep track of the orientation of tetrahedron edges within edge classes, as well as the orientation of the triangular pieces of vertex link that each tetrahedron supplies for each vertex class (this is required for the 3-manifold tests).
(vii) *Counting classes*: We must be able to count the total number of edge and vertex classes (this is required for the edge and vertex counting tests).

It is simple enough to implement all of the above operations in a naïve way. However, all of these operations are performed *extremely* frequently—we merge and split classes every time we step forwards and backwards in the search tree, and we run the remaining tests every time we have a new partially constructed triangulation. Because of this, every one of these operations must be very fast; otherwise the time we save in pruning the search tree may be lost in testing whether we are able to prune.

Naïve array-based and list-based implementations of equivalence classes typically require linear time (or worse) for one or more of the operations in this list. In the following section we introduce the union-find algorithm, and show how it can be modified to perform all of the above operations in logarithmic time at worst.

5.2. *Introducing the Union-Find Algorithm*

The *union-find* algorithm is a classical algorithm for building a set of equivalence classes from a collection of objects and relationships. It also appears frequently in the equivalent setting of building connected graph components from vertices and edges. Union-find is designed to perform the following operations extremely quickly:

- Merging two equivalence classes, i.e., operation (i) in the list above.
- Testing whether two objects are equivalent, i.e., operation (iii) in the list above.

A key property of union-find is that operations of these two types may be interspersed. For instance, we may merge some classes, test some pairs of objects, merge some more classes, test additional pairs of objects, and so on. This is particularly important for our application, since the backtracking nature of the census algorithm means that updates (merging and splitting classes) and tests will be interleaved in this way.

Because of the highly optimised nature of the union-find algorithm, it is *not* well suited for other operations such as splitting classes or listing all objects within a class—as in many classical algorithms, the benefit of improved speed comes at the cost of reduced functionality. In Section 5.3 we modify the algorithm, trading a small (but acceptable) loss of speed for the increased functionality required by operations (i)–(vii) in the list above.

In the meantime, however, we give a brief overview of a typical union-find algorithm. This algorithm is described more thoroughly by Sedgewick [23], and Cormen et al. offer a detailed discussion of its time complexity [9].

Algorithm 5.1 (Union-Find). Suppose we have a set of objects $\{x_1, \ldots, x_n\}$ and a set of relationships of the form $x_i \equiv x_j$. These relationships group the objects into disjoint equivalence classes. For each x_i, denote the class containing x_i by $[x_i]$.

We store each equivalence class as a tree structure, where each object x_i is a node in the relevant tree. Unlike ordinary trees in which parents keep lists of their children, we only require that each child keeps track of its parent (so it is easy to move up the tree, but difficult to move down). Such a structure is illustrated in the leftmost diagram of Fig. 32.

For each equivalence class, the node at the top of the tree is considered the *representative node* of that class. For any object x_i, it is easy to locate the representative of class $[x_i]$—simply continue moving to parent nodes until we reach a node that has no parent. We denote this representative node by $c(x_i)$.

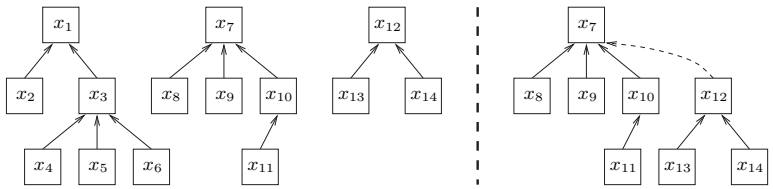

Fig. 32. Tree structures in a union-find algorithm.

To test whether two objects x_i and x_j are equivalent, we simply calculate the representatives $c(x_i)$ and $c(x_j)$ as described above and see whether they are the same.

To add a relationship $x_i \equiv x_j$, we again calculate the representatives $c(x_i)$ and $c(x_j)$. If these representatives are identical then the relationship is redundant, and we discard it. Otherwise we merge the trees together by inserting $c(x_i)$ as a child of $c(x_j)$, or vice versa. This procedure is illustrated in the rightmost diagram of Fig. 32.

It is clear that the time complexity of both adding a relationship and testing for equivalence is bounded by the maximum tree depth. As it stands, the simple union-find described above can produce worst-case trees whose depth is linear in n, resulting in linear-time merging and equivalence testing. We now describe two classical optimisations that reduce this time to "almost constant", in a sense that is described more precisely below.

The first optimisation is a form of balancing, and the second is a form of tree compression. It should be noted that there are several variants of each type of optimisation; see the aforementioned references [9] and [23] for other examples.

Algorithm 5.2 (Height Balancing). This is a simple modification to the merging operation in a union-find algorithm. For each node x_i, we store a number $d(x_i)$ which is an upper bound on the depth of the subtree rooted at x_i. At the beginning of the algorithm (before any relationships are introduced), each $d(x_i)$ is set to 1.

Suppose we are adding the relationship $x_i \equiv x_j$, which requires us to merge trees $c(x_i)$ and $c(x_j)$ as described in Algorithm 5.1. If $d(c(x_i)) < d(c(x_j))$ then we insert $c(x_i)$ as a child of $c(x_j)$. Otherwise we insert $c(x_j)$ as a child of $c(x_i)$, and if the depths are equal then we increment $d(c(x_i))$ by one.

Put simply, height balancing ensures that whenever a merge occurs, the shorter tree is inserted as a child within the taller tree and not the other way around. This reduces the maximum depth to $O(\log n)$, and therefore ensures logarithmic time for both merging and equivalence testing.

Although $d(x_i)$ is defined as an upper bound on the subtree depth, it is clear from Algorithm 5.2 that it is in fact the precise subtree depth. The reason for defining it more loosely as an upper bound is so that height balancing can be used in conjunction with path compression, as described below.

Algorithm 5.3 (Path Compression). This is a modification to the union-find tree traversal operation, where we begin with an object x_i and move upwards to locate the representative $c(x_i)$. Each time we traverse the tree in this way, we return to x_i and make a second run up the tree, this time changing the parent of every intermediate node to point directly to the root $c(x_i)$.

This has the effect of flattening sections of trees at every convenient opportunity, again with the aim of keeping the depths of nodes as small as possible. When path compression is combined with height balancing, the running time for both merging and equivalence testing becomes "almost constant". More specifically, if we perform k operations on n objects then the total running time is guaranteed to be $O(k\alpha(n))$, where α is a function

that grows so slowly that it can be considered constant for all practical purposes. This result is due to Tarjan; see [9] for a summary of the proof.

It should be noted that, when we compress the tree, we do not alter any of the depth estimates $d(x_j)$. To do so would (in the worst case) require linear time, losing the efficiency that we have gained. However, since compression can never increase the depth of a subtree, it is clear that each $d(x_j)$ remains an upper bound as required by Algorithm 5.2.

5.3. *Modifying the Union-Find Algorithm*

Whilst the optimised union-find algorithm is well known for its speed in building equivalence classes, it suffers from the fact that it is *only* designed for building classes—it is not well suited to pulling them apart again. This makes it difficult to apply union-find to the census algorithm, which includes backtracking as a core requirement.

Fortunately we can work around this problem with only a small reduction in speed. Here we describe a series of modifications that can be applied to union-find, resulting in an algorithm that supports not only the merging and splitting of classes, but all of the required operations (i)–(vii) from Section 5.1.

Modification A (No path compression)

We optimise union-find using height balancing as before, but path compression is abandoned. The reason is that the compression of trees is a very difficult operation to undo when backtracking.

By abandoning path compression, our trees become a little deeper on average. However, the height balancing still ensures that the maximum depth is $O(\log n)$, and so merging and equivalence testing both remain logarithmic-time operations.

Modification B (Support splitting)

For each relationship that we add to the system, we store (i) whether that relationship caused two classes to be merged, and (ii) if so, which node formed the root of the smaller tree that was merged into the larger tree. For the merge depicted in Fig. 32, the node stored would be x_{12}.

This makes splitting classes a constant-time operation. As we backtrack and remove a relationship from the system, we simply look up the corresponding node in our table and remove its parent link. This will revert the entire set of trees to its previous state. To undo the merge depicted in Fig. 32, we would look up x_{12} in our table and remove its parent link, which is represented by the dotted arrow.

Note that this procedure requires us to remove relationships in the reverse order to which we added them. However, this requirement is trivially satisfied in the backtracking algorithm of Section 5.1.

Modification C (Add data to nodes)

For each node x_i, we store some additional pieces of information to assist with our various pruning tests. If x_i is the representative of its equivalence class (i.e., the root of its tree), then these pieces of information are as follows:

- We store the total number of objects in the equivalence class, denoted by $n(x_i)$.
- If we are examining vertex links, we also store the number of boundary edges in the partially constructed link of vertex x_i, denoted by $b(x_i)$.

Storing $n(x_i)$ allows us to measure class sizes efficiently (this becomes a constant time lookup). Likewise, storing $b(x_i)$ allows us to test for complete vertex links efficiently, since the link of a representative vertex x_i is complete if and only if $b(x_i) = 0$. We do not bother storing boundary information in the case of edge links, since testing for completeness is easy—an edge link becomes complete as soon as we find a new relationship between two edges already in the same equivalence class.

Recall that we only defined $n(x_i)$ and $b(x_i)$ for the root node of each equivalence class. We also store values of $n(x_i)$ and $b(x_i)$ for other child nodes, though these values differ from the roots and do not describe any current equivalence classes. Instead, they contain historical data, which we will shortly see can be used to support backtracking.

When two classes are merged, only the information at the new root node is updated. Specifically, suppose that trees rooted at x_p and x_c are merged, with x_c inserted as a child of x_p. Then $n(x_p)$ is replaced with $n(x_p) + n(x_c)$, since the size of the new tree is the sum of the two original tree sizes. Likewise, in the case of vertex links, $b(x_p)$ is replaced with $b(x_p) + b(x_c) - 2$. Note that we subtract two because, when vertices x_p and x_c become identified, one boundary edge from the link of x_p becomes joined to one boundary edge from the link of x_c.

It should be observed that these updates involve only constant-time calculations based on the information stored in each original root node. An example of this merging procedure can be seen in Fig. 33, where $x_p = x_1$ and $x_c = x_3$.

Consider now when a class is split. Suppose we are backtracking from the earlier merge operation, and are therefore cutting the subtree rooted at x_c away from its root node x_p. Since we are restoring the system to the state just before the merge, we see that the data stored in node x_c is already accurate—it was certainly accurate before the merge (back when it was a root node), and it has not been changed since. The old data stored in node x_p is restored by replacing $n(x_p)$ with $n(x_p) - n(x_c)$ and replacing $b(x_p)$ with $b(x_p) - b(x_c) + 2$. Again these are constant-time calculations.

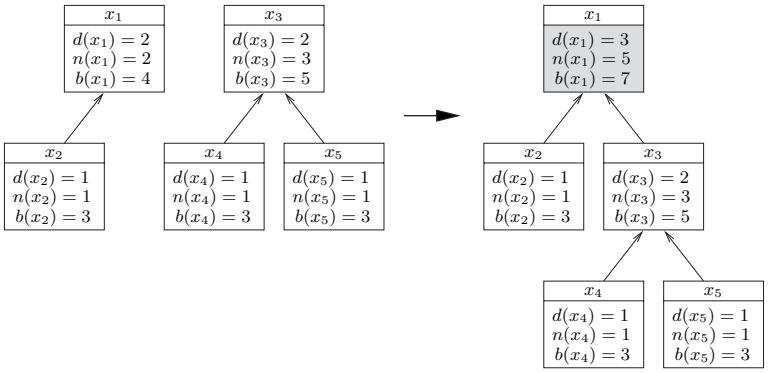

Fig. 33. Updating node data during a merge.

Modification D (Track orientations)

For every edge of every tetrahedron in the triangulation, we define a canonical orientation. Likewise, for every vertex of every tetrahedron, we define a canonical orientation of the corresponding triangular piece of vertex link.

We use these orientations as follows. Recall that each equivalence class represents a set of edges or vertices that are identified together. For each child–parent relationship in a union-find tree, we store a boolean value (true or false) indicating whether the child and parent objects (edges or vertices) are identified in an orientation-preserving or orientation-reversing manner.

Note that this allows us to determine in logarithmic time whether *any* two objects x_i, x_j in the same tree are identified in an orientation-preserving or orientation-reversing manner—we simply follow separate paths from x_i and x_j to the common root node, and keep track of these boolean values as we go.

Consider now the situation in which two classes are merged. This occurs when two objects x_i, x_j from different equivalence classes become identified. The merge requires us to add a new child–parent relationship between the root nodes $c(x_i)$ and $c(x_j)$; we must therefore establish whether or not this new relationship preserves orientation. This is simple to determine—we already know how the orientations of x_i and x_j relate (since this is the identification that is causing the merge), and we can see how x_i and $c(x_i)$ relate and how x_j and $c(x_j)$ relate by following paths to the root of each tree as described above.

The value of storing these booleans is that we can now efficiently test for non-orientable vertex links, as well as for tetrahedron edges that are identified with themselves in reverse (recall that either condition allows us to prune our search tree). Such conditions arise when we attempt to identify two tetrahedron edges or two tetrahedron vertices that already belong to the same equivalence class. By following paths in the corresponding union-find tree from the identified objects x_i, x_j to the common root node $c(x_i) = c(x_j)$ and tracking these boolean values, we can establish whether x_i and x_j are already identified in an orientation-preserving or orientation-reversing manner, and whether the new identification contradicts this.

Modification E (Count classes)

Finally, we keep a separate variable that counts the total number of equivalence classes. This variable is decremented each time we merge two classes, and incremented each time we split a class. This allows us to query the total number of equivalence classes in constant time.

We can now summarise the efficiency of each of the required operations (i)–(vii) from Section 5.1. Merging classes and locating root nodes are logarithmic-time operations, since height balancing ensures that trees have a logarithmic worst-case depth. Likewise, testing for equivalence is logarithmic time. Both splitting classes and counting classes are constant-time operations, and once the class representative (i.e., the root node) has been located, testing for completeness and measuring class sizes become constant time also. Finally, tracking orientations requires logarithmic time since the relevant procedures each involve two child-to-root traversals through a tree.

Table 5. Running times for the modified union-find algorithm (h:m:s).

		Modified algorithm		
Tetrahedra	Old algorithm	Vertex links	Edge links	Both
5	8:38	1:24	0:06	0:02
6	13:49:27	1:23:15	2:40	0:46
7	$\sim$ 65 days	$\sim$ 4 days	2:06:05	21:38

We see then that the time complexity of every required operation is logarithmic at worst. This is a small price to pay for such enhanced functionality, and in the following section we see that it is definitely worth the cost.

5.4. *Experimental Results*

We conclude this section with experimental data that measures the running time of the census algorithm with and without the help of the modified union-find described in Section 5.3.

Table 5 lists precise running times for the generation of non-orientable census triangulations for five to seven tetrahedra. The range five to seven was chosen because smaller censuses all give running times of ≤ 5 seconds, and because for larger censuses it is infeasible to measure the running time of the original algorithm.

The "Old algorithm" column gives the running time of the original census algorithm, as used in [5] and [8]. Note that some pruning is still performed in this case; in particular, the original algorithm performs tests that do not require the construction of large edge links or vertex links, such as tests for edges of degree ≤ 3. The remaining three columns list the running times for the new census algorithm, according to whether we use a modified union-find structure to track vertex links, edge links or both. All times are measured on a single 1.7 GHz IBM Power5 processor.

The results are spectacular. The five-tetrahedron census is reduced from $8\frac{1}{2}$ minutes to 2 seconds, and the seven-tetrahedron census is reduced from 65 days to a mere 22 minutes. This shows improvements of several orders of magnitude, increasing the speed by over 4000 times in the seven-tetrahedron case.

It is only through the use of this modified algorithm that a 10-tetrahedron non-orientable census was even feasible. The results of this census (as well as the sister 10-tetrahedron orientable census) are discussed in the following section.

6. Census Results

Using the algorithmic improvements of Sections 4 and 5, it has been possible to generate new census results in both the non-orientable and orientable cases for up to 10 tetrahedra. In particular:

- The closed non-orientable census has been extended well beyond its previous limit of eight tetrahedra, with a much richer variety of structures appearing as a result.

- The closed orientable census has been extended from just a list of manifolds to a full list of all minimal triangulations of these manifolds.

The non-orientable and orientable cases are treated separately in Sections 6.1 and 6.2 below.

The full census data, including all 16,109 closed minimal $\mathbb{P}^2$-irreducible triangulations formed from ≤ 10 tetrahedra (both orientable and non-orientable), is far too large to include here. This data may be downloaded from the *Regina* website [4] in the form of `.rga` data files, which may be loaded into *Regina* and examined, analysed and/or exported to some other format (the scripting facility built into *Regina* may assist with this). As of version 4.3 these data files are shipped as part of *Regina*, and may be accessed through the *File* $\rightarrow$ *Open Example* menu entry. Each data file begins with a text description of the census parameters and notation used.

The physical hardware requirements for these census runs were filled by the excellent resources of the Victorian Partnership for Advanced Computing (VPAC). *Regina*'s census generation routines provide native support for parallel processing, allowing for an effective use of VPAC's cluster environments.

6.1. *Non-Orientable Census*

The seven-tetrahedron non-orientable censuses of Amendola and Martelli [3] and Burton [5] show that there are eight distinct closed non-orientable $\mathbb{P}^2$-irreducible 3-manifolds that can be formed from ≤ 7 tetrahedra. All of these 3-manifolds fall into one of the following categories:

- Torus bundles over the circle.
- Seifert fibred spaces over either $\mathbb{R}P^2$ or $\bar{D}$ with two exceptional fibres, where $\bar{D}$ is the disc with reflector boundary.

Subsequent results of Burton [8] show that when the limit is raised to ≤ 8 tetrahedra, a total of 18 distinct 3-manifolds are found, all of which belong to the same two categories listed above. We see then that the eight-tetrahedron non-orientable census remains extremely limited in scope.

Happily a much richer variety of 3-manifolds appears in the 9-tetrahedron and 10-tetrahedron non-orientable census tables. We begin with a formal presentation of the new non-orientable census results, and follow this with a summary of the new types of 3-manifolds that are found.

Theorem 6.1. *A total of* 136 *distinct closed non-orientable* $\mathbb{P}^2$*-irreducible* 3*-manifolds can be constructed using* ≤ 10 *tetrahedra. These manifolds are listed individually in Table* 9 *in the Appendix, and are described by a total of* 1390 *distinct minimal triangulations.*

A breakdown of these figures can be found in Table 6*, which lists the number of distinct* 3*-manifolds and distinct minimal triangulations for six to ten tetrahedra. None of these* 3*-manifolds can be constructed from five tetrahedra or fewer.*

Table 6. Closed non-orientable census results.

Tetrahedra	3-Manifolds	Triangulations
≤ 5	0	0
6	5	24
7	3	17
8	10	59
9	33	307
10	85	983
Total	136	1,390

It is immediate from Table 9 that the 9-tetrahedron and 10-tetrahedron manifolds are greater not just in number but also in variety. In addition to the two original categories described above, the following new types of 3-manifolds are also seen:

- Seifert fibred spaces over $\mathbb{R}P^2$ and $\bar{D}$ with three exceptional fibres.
- Seifert fibred spaces over the torus, Klein bottle, $\bar{M}$ and $\bar{\bar{A}}$ with one exceptional fibre, where $\bar{M}$ is the Möbius band with reflector boundary and $\bar{\bar{A}}$ is the annulus with two reflector boundaries.
- Non-geometric graph manifolds of the following types:
  - Seifert fibred spaces over the disc with two exceptional fibres joined to Seifert fibred spaces over the Möbius band or $\bar{A}$ with one exceptional fibre, where $\bar{A}$ is the annulus with one reflector boundary.
  - Seifert fibred spaces over the annulus whose two boundary tori are joined together.
  - The special manifold $\mathrm{SFS}(\dot{A}/o_2)/\left[\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right]$, described below.
- The special non-geometric manifold $\mathrm{SFS}(\dot{D} : (2,1)) \bigcup$ *Gieseking*, also described below.

Two "special manifolds" are referred to in the list above. These have noteworthy properties that set them apart from the remaining manifolds in the census. Each special manifold is described in turn.

- *The nine-tetrahedron manifold* $\mathrm{SFS}(\dot{A}/o_2)/\left[\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right]$.
  This is a non-geometric graph manifold, formed from a single Seifert fibred space by identifying its two boundaries. What sets this apart from the other graph manifolds is that the boundaries are Klein bottles, not tori.

  The base orbifold of the Seifert fibred space is $\dot{A}$, the annulus with one regular boundary curve and one half-reflector, half-regular boundary curve. In addition, the path around the annulus is a fibre-reversing curve in the Seifert fibred space (this is the meaning of the symbol o_2). This base orbifold is illustrated in the leftmost diagram of Fig. 34.

  The resulting Seifert fibred space has two Klein bottle boundaries. The lower central diagram of Fig. 34 shows the Klein bottle corresponding to the half-regular boundary on the outside of the annulus, and the upper central diagram shows the Klein bottle corresponding to the regular boundary on the inside of the annulus. The dashed lines on each Klein bottle depict the fibres of the Seifert fibred space.

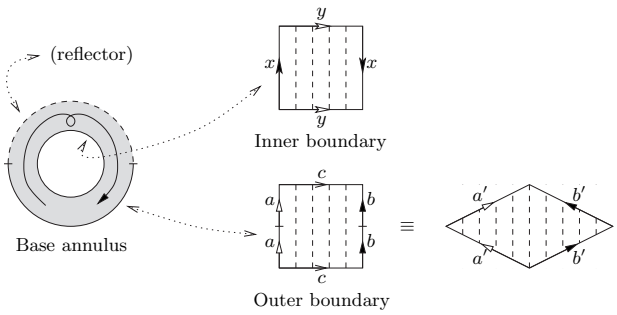

Fig. 34. The base orbifold and boundary Klein bottles for SFS($\dot{A}/o_2$).

It is clear that the two Klein bottles cannot be identified so that the two sets of fibres match. Instead, consider the diamond representation of the outer boundary Klein bottle as depicted in the rightmost diagram of Fig. 34. The two Klein bottles are identified so that the fibre x maps to the transverse curve $a'b'^{-1}$, and the closed curve a' (which is isotopic to the closed half-fibre a) maps to the transverse curve y.

- *The* 10*-tetrahedron manifold* SFS($\dot{D}$: (2, 1)) $\bigcup$ *Gieseking*
  This is also non-geometric, but it is not a graph manifold. It consists of a Seifert fibred block and a hyperbolic block joined along a common Klein bottle boundary.

  The base orbifold of the Seifert fibred block is $\dot{D}$, the disc with half-reflector, half-regular boundary. A single exceptional fibre of index 2 is also present within this block.

  The hyperbolic block is a truncated Gieseking manifold. The *Gieseking manifold* is the smallest volume cusped hyperbolic manifold [1]. It is non-orientable with a single Klein bottle cusp, and has hyperbolic volume 1.01494161. In the census shipped with *SnapPea* [24] it is listed as the manifold `m000`.

  Here we truncate the Gieseking manifold at its cusp, giving rise to a real Klein bottle boundary. This boundary is then identified with the boundary of the Seifert fibred block, producing the closed manifold identified here.

Note that, with the single exception of the special 10-tetrahedron manifold described above, every manifold in this census is a graph manifold. Theorem 6.1 therefore yields the following corollary:

Corollary 6.2. *All closed non-orientable* $\mathbb{P}^2$*-irreducible* 3*-manifolds formed from* ≤ 9 *tetrahedra are graph manifolds. Moreover, this bound is sharp—there exists a closed non-orientable* $\mathbb{P}^2$*-irreducible* 3*-manifold formed from* 10 *tetrahedra whose JSJ decomposition contains a hyperbolic block.*

It can also be noted that none of the manifolds in this census is a closed hyperbolic manifold. This contrasts with the orientable census, in which closed hyperbolic manifolds first appear at the nine-tetrahedron level [19].

We can nevertheless deduce the point at which closed non-orientable hyperbolic manifolds will appear. Hodgson and Weeks present a list of candidate smallest volume

non-orientable hyperbolic manifolds [11]. The smallest volume manifold from this list has volume 2.02988321, and can be triangulated using 11 tetrahedra. We therefore obtain the following sharp bound.

Corollary 6.3. *For any closed non-orientable $\mathbb{P}^2$-irreducible hyperbolic 3-manifold M, a minimal triangulation of M must contain ≥ 11 tetrahedra. Moreover, this bound is sharp—the non-orientable manifold of volume* 2.02988321 *from the Hodgson–Weeks census* [11] *can be triangulated using precisely* 11 *tetrahedra.*

Since both the orientable and non-orientable census tables contain a great many graph manifolds, it is interesting to compare these tables to see at what stage non-graph manifolds first appear. Combining the new non-orientable results with the 11-tetrahedron orientable census of Matveev [21], we find the following:

- The first geometric non-graph manifolds are orientable, with 9 tetrahedra (these are the four closed hyperbolic manifolds of smallest known volume). Such manifolds do not appear in the non-orientable tables until 11 tetrahedra.
- The first non-geometric non-graph manifold is non-orientable, with 10 tetrahedra (this is $\mathrm{SFS}(\dot{D} : (2,1)) \bigcup$ *Gieseking* as described above). Such manifolds do not appear in the orientable tables until 11 tetrahedra.

We close these non-orientable census results with a discussion of the combinatorial structures of minimal triangulations. In the 8-tetrahedron non-orientable census paper [8], the following conjectures are made:

- Every minimal triangulation of a non-flat torus bundle over the circle is a *layered torus bundle* (Conjecture 3.1).
- Every minimal triangulation of a non-flat Seifert fibred space over $\mathbb{R}P^2$ or $\bar{D}$ with two exceptional fibres is either a *plugged thin I-bundle* or a *plugged thick I-bundle* (Conjecture 3.2).

Here layered torus bundles and plugged thin/thick I-bundles are specific families of triangulations; see either [8] for an overview or [5] for full details.

The new census results confirm these conjectures at the 9-tetrahedron and 10-tetrahedron levels. Specifically, for 9 and 10 tetrahedra, every minimal triangulation of a torus bundle over the circle is a layered torus bundle, and every minimal triangulation of a Seifert fibred space over $\mathbb{R}P^2$ or $\bar{D}$ with two exceptional fibres is a plugged thin or thick I-bundle.

6.2. *Orientable Census*

As discussed in the Introduction, closed orientable census results are presently known for ≤ 11 tetrahedra (though Matveev has recently announced partial results for 12 tetrahedra also [22]). However, for 7 tetrahedra and above these results are presented by their respective authors as lists of manifolds. Full lists of all minimal triangulations (or even the numbers of such triangulations) are not included.

Table 7. Closed orientable census results.

Tetrahedra	3-Manifolds	Triangulations
1	3	4
2	6	9
3	7	7
4	14	15
5	31	40
6	74	115
7	175	309
8	436	945
9	1,154	3,031
10	3,078	10,244
Total	4,978	14,719

The most comprehensive lists of orientable minimal triangulations to date are those of Matveev [20], enumerated for ≤ 6 tetrahedra and presented in the equivalent language of special spines. Here we extend this list of minimal triangulations to ≤ 10 tetrahedra.

Theorem 6.4. *There are precisely* 14,719 *minimal triangulations of closed orientable irreducible* 3*-manifolds that are constructed using* ≤ 10 *tetrahedra. A breakdown of this count into* $1, 2, \ldots, 10$ *tetrahedra is provided in Table* 7.

Unlike the non-orientable census, both the triangulations and the 3-manifolds are too numerous to list here. As described in the beginning of Section 6, the full data can be downloaded from the *Regina* website [4].

Note that Table 7 also lists the number of distinct 3-manifolds obtained at each level of the census. These manifold counts agree with the previous results of Martelli [14] and Matveev [21].

Moreover, the full list of 4,978 manifolds obtained here with *Regina* has been compared with the lists of Martelli and Petronio [18]. A detailed matching confirms that the 4,978 distinct manifolds obtained in this orientable census are identical to the 4,978 distinct manifolds obtained by Martelli and Petronio. Since both sets of census results rely heavily on computer searches, an independent verification such as this provides an extra level of confidence in both sets of software.

Acknowledgements

The author is grateful to the Victorian Partnership for Advanced Computing for supporting this work and for the use of their exceptional computational resources. Thanks are also offered to the anonymous referees for their suggestions.

Appendix. Tables of Non-Orientable Manifolds

In this appendix we present the full list of all closed non-orientable $\mathbb{P}^2$-irreducible 3-manifolds that can be constructed using 10 tetrahedra or fewer, as discussed in Section 6.1. This list, given as Table 9 on the following pages, is divided into sections according to the number of tetrahedra in a minimal triangulation. Note that none of these 3-manifolds can be formed from fewer than six tetrahedra.

For each manifold M, this list also presents the first homology group $H_1(M)$ and the number of distinct minimal triangulations of M.

The notation used for the base orbifolds of Seifert fibred spaces is explained in Table 8. In addition, non-geometric graph manifolds appear in the list as follows:

- $\mathrm{SFS}(\cdots) \bigcup_N \mathrm{SFS}(\cdots)$
  This represents two Seifert fibred spaces, each with a single torus boundary, where these spaces are joined together along their torus boundaries according to the 2×2 matching matrix N.
- $\mathrm{SFS}(\cdots)/N$
  This represents a single Seifert fibred space with two torus boundaries, where these boundaries are identified according to the 2×2 matching matrix N.

The special non-geometric manifolds $\mathrm{SFS}(\dot{A}/o_2)/\left[\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right]$ and $\mathrm{SFS}(\dot{D} : (2,1)) \bigcup$ *Gieseking* do not fit neatly into this notation scheme; see Section 6.1 for a detailed description of each.

Table 8. Notation for base orbifolds of Seifert fibred spaces.

Symbol	Base orbifold
D	Disc
$\bar{D}$	Disc with reflector boundary
$\dot{D}$	Disc with half-reflector, half-regular boundary
M	Möbius band
$\bar{M}$	Möbius band with reflector boundary
$\bar{M}/n_2$	Möbius band with reflector boundary and fibre-reversing curves
A	Annulus
$\bar{A}$	Annulus with one reflector boundary
$\bar{\bar{A}}$	Annulus with two reflector boundaries
$\bar{\bar{A}}/o_2$	Annulus with two reflector boundaries and fibre-reversing curves
$\dot{A}/o_2$	Annulus with one half-reflector, half-regular boundary and fibre-reversing curves
$\mathbb{R}P^2$	Projective plane
T^2/o_2	Torus containing fibre-reversing curves
K^2	Klein bottle
K^2/n_3	Klein bottle containing fibre-reversing curves, but with total space remaining non-orientable

Table 9. Details for each closed non-orientable $\mathbb{P}^2$-irreducible 3-manifold.

# Tet.	3-Manifold	# Tri.	Homology
6	$T^2 \times I/\left[\begin{smallmatrix}1&1\\1&0\end{smallmatrix}\right]$	1	$\mathbb{Z}$
	$T^2 \times I/\left[\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right]$	6	$\mathbb{Z} \oplus \mathbb{Z}$
	$T^2 \times I/\left[\begin{smallmatrix}1&0\\0&-1\end{smallmatrix}\right]$	3	$\mathbb{Z} \oplus \mathbb{Z} \oplus \mathbb{Z}_2$
	SFS($\mathbb{R}P^2$: (2, 1) (2, 1))	9	$\mathbb{Z} \oplus \mathbb{Z}_4$
	SFS($\bar{D}$: (2, 1) (2, 1))	5	$\mathbb{Z} \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_2$
7	$T^2 \times I/\left[\begin{smallmatrix}2&1\\1&0\end{smallmatrix}\right]$	4	$\mathbb{Z} \oplus \mathbb{Z}_2$
	SFS($\mathbb{R}P^2$: (2, 1) (3, 1))	10	$\mathbb{Z}$
	SFS($\bar{D}$: (2, 1) (3, 1))	3	$\mathbb{Z} \oplus \mathbb{Z}_2$
8	$T^2 \times I/\left[\begin{smallmatrix}3&1\\1&0\end{smallmatrix}\right]$	10	$\mathbb{Z} \oplus \mathbb{Z}_3$
	$T^2 \times I/\left[\begin{smallmatrix}3&2\\2&1\end{smallmatrix}\right]$	2	$\mathbb{Z} \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_2$
	SFS($\mathbb{R}P^2$: (2, 1) (4, 1))	10	$\mathbb{Z} \oplus \mathbb{Z}_2$
	SFS($\mathbb{R}P^2$: (2, 1) (5, 2))	10	$\mathbb{Z}$
	SFS($\mathbb{R}P^2$: (3, 1) (3, 1))	7	$\mathbb{Z} \oplus \mathbb{Z}_6$
	SFS($\mathbb{R}P^2$: (3, 1) (3, 2))	9	$\mathbb{Z} \oplus \mathbb{Z}_3$
	SFS($\bar{D}$: (2, 1) (4, 1))	3	$\mathbb{Z} \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_2$
	SFS($\bar{D}$: (2, 1) (5, 2))	3	$\mathbb{Z} \oplus \mathbb{Z}_2$
	SFS($\bar{D}$: (3, 1) (3, 1))	3	$\mathbb{Z} \oplus \mathbb{Z}_3$
	SFS($\bar{D}$: (3, 1) (3, 2))	2	$\mathbb{Z} \oplus \mathbb{Z}_3$
9	$T^2 \times I/\left[\begin{smallmatrix}4&1\\1&0\end{smallmatrix}\right]$	18	$\mathbb{Z} \oplus \mathbb{Z}_4$
	$T^2 \times I/\left[\begin{smallmatrix}4&3\\3&2\end{smallmatrix}\right]$	10	$\mathbb{Z} \oplus \mathbb{Z}_6$
	SFS($\mathbb{R}P^2$: (2, 1) (5, 1))	10	$\mathbb{Z}$
	SFS($\mathbb{R}P^2$: (2, 1) (7, 2))	10	$\mathbb{Z}$
	SFS($\mathbb{R}P^2$: (2, 1) (7, 3))	10	$\mathbb{Z}$
	SFS($\mathbb{R}P^2$: (2, 1) (8, 3))	10	$\mathbb{Z} \oplus \mathbb{Z}_2$
	SFS($\mathbb{R}P^2$: (3, 1) (4, 1))	10	$\mathbb{Z}$
	SFS($\mathbb{R}P^2$: (3, 1) (4, 3))	10	$\mathbb{Z}$
	SFS($\mathbb{R}P^2$: (3, 1) (5, 2))	10	$\mathbb{Z}$
	SFS($\mathbb{R}P^2$: (3, 1) (5, 3))	10	$\mathbb{Z} \oplus \mathbb{Z}_2$
	SFS($\mathbb{R}P^2$: (2, 1) (2, 1) (2, 1))	18	$\mathbb{Z} \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_2$
	SFS($\bar{D}$: (2, 1) (5, 1))	3	$\mathbb{Z} \oplus \mathbb{Z}_2$
	SFS($\bar{D}$: (2, 1) (7, 2))	3	$\mathbb{Z} \oplus \mathbb{Z}_2$
	SFS($\bar{D}$: (2, 1) (7, 3))	3	$\mathbb{Z} \oplus \mathbb{Z}_2$
	SFS($\bar{D}$: (2, 1) (8, 3))	3	$\mathbb{Z} \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_2$
	SFS($\bar{D}$: (3, 1) (4, 1))	3	$\mathbb{Z} \oplus \mathbb{Z}_2$
	SFS($\bar{D}$: (3, 1) (4, 3))	3	$\mathbb{Z} \oplus \mathbb{Z}_2$
	SFS($\bar{D}$: (3, 1) (5, 2))	3	$\mathbb{Z}$
	SFS($\bar{D}$: (3, 1) (5, 3))	3	$\mathbb{Z}$
	SFS($\bar{D}$: (2, 1) (2, 1) (2, 1))	5	$\mathbb{Z} \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_2$
	SFS(T^2/o_2 : (2, 1))	12	$\mathbb{Z} \oplus \mathbb{Z}$
	SFS(K^2 : (2, 1))	21	$\mathbb{Z} \oplus \mathbb{Z}$
	SFS(K^2/n_3 : (2, 1))	39	$\mathbb{Z} \oplus \mathbb{Z}_8$
	SFS($\bar{M}$: (2, 1))	9	$\mathbb{Z} \oplus \mathbb{Z} \oplus \mathbb{Z}_2$
	SFS($\bar{M}/n_2$: (2, 1))	9	$\mathbb{Z} \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_4$
	SFS($\bar{\bar{A}}$: (2, 1))	2	$\mathbb{Z} \oplus \mathbb{Z} \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_2$

Continued

Table 9 (*Continued*)

# Tet.	3-Manifold	# Tri.	Homology
9 (ctd.)	$\mathrm{SFS}(\bar{\bar{A}}/o_2 : (2,1))$	2	$\mathbb{Z} \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_2$
	$\mathrm{SFS}(D : (2,1)(2,1)) \bigcup_{\left[\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right]} \mathrm{SFS}(M : (2,1))$	10	$\mathbb{Z} \oplus \mathbb{Z}_4$
	$\mathrm{SFS}(D : (2,1)(2,1)) \bigcup_{\left[\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right]} \mathrm{SFS}(\bar{A} : (2,1))$	3	$\mathbb{Z} \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_4$
	$\mathrm{SFS}(A : (2,1))/\left[\begin{smallmatrix}0&-1\\1&0\end{smallmatrix}\right]$	35	$\mathbb{Z}$
	$\mathrm{SFS}(A : (2,1))/\left[\begin{smallmatrix}0&-1\\1&1\end{smallmatrix}\right]$	7	$\mathbb{Z} \oplus \mathbb{Z}_3$
	$\mathrm{SFS}(A : (2,1))/\left[\begin{smallmatrix}1&2\\0&1\end{smallmatrix}\right]$	1	$\mathbb{Z} \oplus \mathbb{Z}_2$
	$\mathrm{SFS}(\dot{A}/o_2)/\left[\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right]$	2	$\mathbb{Z} \oplus \mathbb{Z}_4$
10	$T^2 \times I/\left[\begin{smallmatrix}5&1\\1&0\end{smallmatrix}\right]$	30	$\mathbb{Z} \oplus \mathbb{Z}_5$
	$T^2 \times I/\left[\begin{smallmatrix}5&4\\4&3\end{smallmatrix}\right]$	16	$\mathbb{Z} \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_4$
	$T^2 \times I/\left[\begin{smallmatrix}8&3\\3&1\end{smallmatrix}\right]$	14	$\mathbb{Z} \oplus \mathbb{Z}_9$
	$T^2 \times I/\left[\begin{smallmatrix}8&5\\5&3\end{smallmatrix}\right]$	2	$\mathbb{Z} \oplus \mathbb{Z}_{11}$
	$\mathrm{SFS}(\mathbb{R}P^2 : (2,1)\ (6,1))$	10	$\mathbb{Z} \oplus \mathbb{Z}_4$
	$\mathrm{SFS}(\mathbb{R}P^2 : (2,1)\ (9,2))$	10	$\mathbb{Z}$
	$\mathrm{SFS}(\mathbb{R}P^2 : (2,1)\ (9,4))$	10	$\mathbb{Z}$
	$\mathrm{SFS}(\mathbb{R}P^2 : (2,1)\ (10,3))$	10	$\mathbb{Z} \oplus \mathbb{Z}_4$
	$\mathrm{SFS}(\mathbb{R}P^2 : (2,1)\ (11,3))$	10	$\mathbb{Z}$
	$\mathrm{SFS}(\mathbb{R}P^2 : (2,1)\ (11,4))$	10	$\mathbb{Z}$
	$\mathrm{SFS}(\mathbb{R}P^2 : (2,1)\ (12,5))$	10	$\mathbb{Z} \oplus \mathbb{Z}_2$
	$\mathrm{SFS}(\mathbb{R}P^2 : (2,1)\ (13,5))$	10	$\mathbb{Z}$
	$\mathrm{SFS}(\mathbb{R}P^2 : (3,1)\ (5,1))$	10	$\mathbb{Z} \oplus \mathbb{Z}_2$
	$\mathrm{SFS}(\mathbb{R}P^2 : (3,1)\ (5,4))$	10	$\mathbb{Z}$
	$\mathrm{SFS}(\mathbb{R}P^2 : (3,1)\ (7,2))$	10	$\mathbb{Z}$
	$\mathrm{SFS}(\mathbb{R}P^2 : (3,1)\ (7,3))$	10	$\mathbb{Z} \oplus \mathbb{Z}_2$
	$\mathrm{SFS}(\mathbb{R}P^2 : (3,1)\ (7,4))$	10	$\mathbb{Z}$
	$\mathrm{SFS}(\mathbb{R}P^2 : (3,1)\ (7,5))$	10	$\mathbb{Z} \oplus \mathbb{Z}_2$
	$\mathrm{SFS}(\mathbb{R}P^2 : (3,1)\ (8,3))$	10	$\mathbb{Z}$
	$\mathrm{SFS}(\mathbb{R}P^2 : (3,1)\ (8,5))$	10	$\mathbb{Z}$
	$\mathrm{SFS}(\mathbb{R}P^2 : (4,1)\ (4,1))$	7	$\mathbb{Z} \oplus \mathbb{Z}_8$
	$\mathrm{SFS}(\mathbb{R}P^2 : (4,1)\ (4,3))$	9	$\mathbb{Z} \oplus \mathbb{Z}_8$
	$\mathrm{SFS}(\mathbb{R}P^2 : (4,1)\ (5,2))$	10	$\mathbb{Z}$
	$\mathrm{SFS}(\mathbb{R}P^2 : (4,1)\ (5,3))$	10	$\mathbb{Z}$
	$\mathrm{SFS}(\mathbb{R}P^2 : (5,2)\ (5,2))$	7	$\mathbb{Z} \oplus \mathbb{Z}_{10}$
	$\mathrm{SFS}(\mathbb{R}P^2 : (5,2)\ (5,3))$	9	$\mathbb{Z} \oplus \mathbb{Z}_5$
	$\mathrm{SFS}(\mathbb{R}P^2 : (2,1)\ (2,1)\ (3,1))$	92	$\mathbb{Z} \oplus \mathbb{Z}_4$
	$\mathrm{SFS}(\bar{D} : (2,1)\ (6,1))$	3	$\mathbb{Z} \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_2$
	$\mathrm{SFS}(\bar{D} : (2,1)\ (9,2))$	3	$\mathbb{Z} \oplus \mathbb{Z}_2$
	$\mathrm{SFS}(\bar{D} : (2,1)\ (9,4))$	3	$\mathbb{Z} \oplus \mathbb{Z}_2$
	$\mathrm{SFS}(\bar{D} : (2,1)\ (10,3))$	3	$\mathbb{Z} \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_2$
	$\mathrm{SFS}(\bar{D} : (2,1)\ (11,3))$	3	$\mathbb{Z} \oplus \mathbb{Z}_2$
	$\mathrm{SFS}(\bar{D} : (2,1)\ (11,4))$	3	$\mathbb{Z} \oplus \mathbb{Z}_2$
	$\mathrm{SFS}(\bar{D} : (2,1)\ (12,5))$	3	$\mathbb{Z} \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_2$
	$\mathrm{SFS}(\bar{D} : (2,1)\ (13,5))$	3	$\mathbb{Z} \oplus \mathbb{Z}_2$
	$\mathrm{SFS}(\bar{D} : (3,1)\ (5,1))$	3	$\mathbb{Z}$
	$\mathrm{SFS}(\bar{D} : (3,1)\ (5,4))$	3	$\mathbb{Z}$
	$\mathrm{SFS}(\bar{D} : (3,1)\ (7,2))$	3	$\mathbb{Z}$

Continued

Table 9 (*Continued*)

# Tet.	3-Manifold	# Tri.	Homology
10 (ctd.)	SFS($\bar{D}$: (3, 1) (7, 3))	3	$\mathbb{Z}$
	SFS($\bar{D}$: (3, 1) (7, 4))	3	$\mathbb{Z}$
	SFS($\bar{D}$: (3, 1) (7, 5))	3	$\mathbb{Z}$
	SFS($\bar{D}$: (3, 1) (8, 3))	3	$\mathbb{Z} \oplus \mathbb{Z}_2$
	SFS($\bar{D}$: (3, 1) (8, 5))	3	$\mathbb{Z} \oplus \mathbb{Z}_2$
	SFS($\bar{D}$: (4, 1) (4, 1))	3	$\mathbb{Z} \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_4$
	SFS($\bar{D}$: (4, 1) (4, 3))	2	$\mathbb{Z} \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_4$
	SFS($\bar{D}$: (4, 1) (5, 2))	3	$\mathbb{Z} \oplus \mathbb{Z}_2$
	SFS($\bar{D}$: (4, 1) (5, 3))	3	$\mathbb{Z} \oplus \mathbb{Z}_2$
	SFS($\bar{D}$: (5, 2) (5, 2))	3	$\mathbb{Z} \oplus \mathbb{Z}_5$
	SFS($\bar{D}$: (5, 2) (5, 3))	2	$\mathbb{Z} \oplus \mathbb{Z}_5$
	SFS($\bar{D}$: (2, 1) (2, 1) (3, 1))	19	$\mathbb{Z} \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_2$
	SFS(T^2/o_2 : (3, 1))	12	$\mathbb{Z} \oplus \mathbb{Z}$
	SFS(T^2/o_2 : (3, 2))	12	$\mathbb{Z} \oplus \mathbb{Z} \oplus \mathbb{Z}_2$
	SFS(K^2 : (3, 1))	21	$\mathbb{Z} \oplus \mathbb{Z}$
	SFS(K^2 : (3, 2))	21	$\mathbb{Z} \oplus \mathbb{Z} \oplus \mathbb{Z}_2$
	SFS(K^2/n_3 : (3, 1))	39	$\mathbb{Z} \oplus \mathbb{Z}_{12}$
	SFS(K^2/n_3 : (3, 2))	39	$\mathbb{Z} \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_6$
	SFS($\bar{M}$: (3, 1))	15	$\mathbb{Z} \oplus \mathbb{Z}$
	SFS($\bar{M}/n_2$: (3, 1))	15	$\mathbb{Z} \oplus \mathbb{Z}_{12}$
	SFS($\bar{\bar{A}}$: (3, 1))	3	$\mathbb{Z} \oplus \mathbb{Z} \oplus \mathbb{Z}_2$
	SFS($\bar{\bar{A}}/o_2$: (3, 1))	3	$\mathbb{Z} \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_6$
	SFS(D : (2, 1)(2, 1)) $\bigcup_{\left[\begin{smallmatrix}1&1\\0&1\end{smallmatrix}\right]}$ SFS(M : (2, 1))	29	$\mathbb{Z} \oplus \mathbb{Z}_8$
	SFS(D : (2, 1)(2, 1)) $\bigcup_{\left[\begin{smallmatrix}-1&2\\0&1\end{smallmatrix}\right]}$ SFS(M : (2, 1))	20	$\mathbb{Z} \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_2$
	SFS(D : (2, 1)(2, 1)) $\bigcup_{\left[\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right]}$ SFS(M : (3, 1))	10	$\mathbb{Z} \oplus \mathbb{Z}_4$
	SFS(D : (2, 1)(2, 1)) $\bigcup_{\left[\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right]}$ SFS(M : (3, 2))	10	$\mathbb{Z} \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_2$
	SFS(D : (2, 1)(2, 1)) $\bigcup_{\left[\begin{smallmatrix}1&1\\0&1\end{smallmatrix}\right]}$ SFS($\bar{A}$: (2, 1))	8	$\mathbb{Z} \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_4$
	SFS(D : (2, 1)(2, 1)) $\bigcup_{\left[\begin{smallmatrix}-1&2\\0&1\end{smallmatrix}\right]}$ SFS($\bar{A}$: (2, 1))	6	$\mathbb{Z} \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_2$
	SFS(D : (2, 1)(2, 1)) $\bigcup_{\left[\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right]}$ SFS($\bar{A}$: (3, 1))	3	$\mathbb{Z} \oplus \mathbb{Z}_4$
	SFS(D : (2, 1)(2, 1)) $\bigcup_{\left[\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right]}$ SFS($\bar{A}$: (3, 2))	3	$\mathbb{Z} \oplus \mathbb{Z}_4$
	SFS(D : (2, 1)(3, 1)) $\bigcup_{\left[\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right]}$ SFS(M : (2, 1))	19	$\mathbb{Z}$
	SFS(D : (2, 1)(3, 1)) $\bigcup_{\left[\begin{smallmatrix}-1&1\\0&1\end{smallmatrix}\right]}$ SFS(M : (2, 1))	46	$\mathbb{Z}$
	SFS(D : (2, 1)(3, 1)) $\bigcup_{\left[\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right]}$ SFS($\bar{A}$: (2, 1))	5	$\mathbb{Z} \oplus \mathbb{Z}_2$
	SFS(D : (2, 1)(3, 1)) $\bigcup_{\left[\begin{smallmatrix}-1&1\\0&1\end{smallmatrix}\right]}$ SFS($\bar{A}$: (2, 1))	12	$\mathbb{Z} \oplus \mathbb{Z}_2$
	SFS(D : (2, 1)(3, 2)) $\bigcup_{\left[\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right]}$ SFS(M : (2, 1))	19	$\mathbb{Z}$
	SFS(D : (2, 1)(3, 2)) $\bigcup_{\left[\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right]}$ SFS($\bar{A}$: (2, 1))	5	$\mathbb{Z} \oplus \mathbb{Z}_2$
	SFS(A : (2, 1))/$\left[\begin{smallmatrix}0&-1\\1&2\end{smallmatrix}\right]$	41	$\mathbb{Z} \oplus \mathbb{Z}_5$

Continued

Table 9 (*Continued*)

# Tet.	3-Manifold	# Tri.	Homology
10 (ctd.)	$\mathrm{SFS}(A : (2,1))/\left[\begin{smallmatrix}-1 & -2\\ 1 & 1\end{smallmatrix}\right]$	11	$\mathbb{Z} \oplus \mathbb{Z}_6$
	$\mathrm{SFS}(A : (2,1))/\left[\begin{smallmatrix}1 & 3\\ 0 & 1\end{smallmatrix}\right]$	4	$\mathbb{Z} \oplus \mathbb{Z}_3$
	$\mathrm{SFS}(A : (2,1))/\left[\begin{smallmatrix}-1 & 3\\ 0 & -1\end{smallmatrix}\right]$	4	$\mathbb{Z} \oplus \mathbb{Z}_3$
	$\mathrm{SFS}(A : (3,1))/\left[\begin{smallmatrix}0 & -1\\ 1 & 0\end{smallmatrix}\right]$	35	$\mathbb{Z}$
	$\mathrm{SFS}(A : (3,1))/\left[\begin{smallmatrix}0 & -1\\ 1 & 1\end{smallmatrix}\right]$	7	$\mathbb{Z} \oplus \mathbb{Z}_4$
	$\mathrm{SFS}(A : (3,1))/\left[\begin{smallmatrix}1 & 2\\ 0 & 1\end{smallmatrix}\right]$	1	$\mathbb{Z} \oplus \mathbb{Z}_2$
	$\mathrm{SFS}(A : (3,2))/\left[\begin{smallmatrix}0 & -1\\ 1 & 0\end{smallmatrix}\right]$	35	$\mathbb{Z} \oplus \mathbb{Z}_2$
	$\mathrm{SFS}(A : (3,2))/\left[\begin{smallmatrix}0 & -1\\ 1 & 1\end{smallmatrix}\right]$	7	$\mathbb{Z} \oplus \mathbb{Z}_5$
	$\mathrm{SFS}(A : (3,2))/\left[\begin{smallmatrix}1 & 2\\ 0 & 1\end{smallmatrix}\right]$	1	$\mathbb{Z} \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_2$
	$\mathrm{SFS}(\dot{D} : (2,1)) \bigcup \mathit{Gieseking}$	13	$\mathbb{Z} \oplus \mathbb{Z}_2$

References

1. C. C. Adams, The noncompact hyperbolic 3-manifold of minimal volume, *Proc. Amer. Math. Soc.* **100**(4) (1987), 601–606.
2. G. Amendola and B. Martelli, Non-orientable 3-manifolds of small complexity, *Topology Appl.* **133**(2) (2003), 157–178.
3. G. Amendola and B. Martelli, Non-orientable 3-manifolds of complexity up to 7, *Topology Appl.* **150**(1–3) (2005), 179–195.
4. B. A. Burton, *Regina*: Normal surface and 3-manifold topology software, `http://regina.sourceforge.net/`, 1999–2006.
5. B. A. Burton, Structures of small closed non-orientable 3-manifold triangulations, *J. Knot Theory Ramifications* **16**(5) (2007), 545–574.
6. B. A. Burton, Face-pairing graphs and 3-manifold enumeration, *J. Knot Theory Ramifications* **13**(8) (2004), 1057–1101.
7. B. A. Burton, Introducing Regina, the 3-manifold topology software, *Experiment. Math.* **13**(3) (2004), 267–272.
8. B. A. Burton, Observations from the 8-tetrahedron non-orientable census, *Experiment. Math.*, to appear, `math.GT/0509345`, 2005.
9. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd edn., MIT Press, Cambridge, MA, 2001.
10. M. V. Hildebrand and J. R. Weeks, A computer generated census of cusped hyperbolic 3-manifolds, in *Computers and Mathematics* (Proc. Conf. Cambridge, MA, 1989), pp. 53–59, Springer, New York, 1989.
11. C. D. Hodgson and J. R. Weeks, Symmetries, isometries and length spectra of closed hyperbolic three-manifolds, *Experiment. Math.* **3**(4) (1994), 261–274.
12. W. Jaco and J. H. Rubinstein, 0-efficient triangulations of 3-manifolds, *J. Differential Geom.* **65**(1) (2003), 61–168.
13. W. Jaco and J. H. Rubinstein, Layered-triangulations of 3-manifolds, Preprint, `math.GT/0603601`, 2006.
14. B. Martelli, Complexity of 3-manifolds, Preprint, `math.GT/0405250`, 2005.
15. B. Martelli and C. Petronio, Three-manifolds having complexity at most 9, *Experiment. Math.* **10**(2) (2001), 207–236.
16. B. Martelli and C. Petronio, A new decomposition theorem for 3-manifolds, *Illinois J. Math.* **46** (2002), 755–780.
17. B. Martelli and C. Petronio, Complexity of geometric three-manifolds, *Geom. Dedicata* **108**(1) (2004), 15–69.

18. B. Martelli and C. Petronio, Computer data on 3-manifolds, http://www.dm.unipi.it/pages/petronio/public_html/data.html, 2004.
19. S. V. Matveev, Complexity theory of three-dimensional manifolds, *Acta Appl. Math.* **19**(2) (1990), 101–130.
20. S. V. Matveev, Tables of 3-manifolds up to complexity 6, Max-Planck-Institut für Mathematik Preprint Series (1998), no. 67, http://www.mpim-bonn.mpg.de/html/preprints/preprints.html.
21. S. V. Matveev, Recognition and tabulation of three-dimensional manifolds, *Dokl. Akad. Nauk* **400**(1) (2005), 26–28.
22. S. V. Matveev, Tabulation of three-dimensional manifolds, *Russian Math. Surveys* **60**(4) (2005), 673–698.
23. R. Sedgewick, *Algorithms in C++*, Addison-Wesley, Reading, MA, 1992.
24. J. R. Weeks, *SnapPea*: Hyperbolic 3-manifold software, http://www.northnet.org/weeks/index/SnapPea.html, 1991–2000.

Received May 8, 2006, *and in revised form October* 7, 2006. *Online publication August* 22, 2007.