Fixed parameter tractable algorithms in combinatorial topology

Benjamin A. Burton* and William Pettersson

The University of Queensland, Brisbane, Australia,
`bab@maths.uq.edu.au`, `william@ewpettersson.se`

Abstract. To enumerate 3-manifold triangulations with a given property, one typically begins with a set of potential face pairing graphs (also known as dual 1-skeletons), and then attempts to flesh each graph out into full triangulations using an exponential-time enumeration. However, asymptotically most graphs do not result in *any* 3-manifold triangulation, which leads to significant "wasted time" in topological enumeration algorithms. Here we give a new algorithm to determine whether a given face pairing graph supports any 3-manifold triangulation, and show this to be fixed parameter tractable in the treewidth of the graph.

We extend this result to a "meta-theorem" by defining a broad class of properties of triangulations, each with a corresponding fixed parameter tractable existence algorithm. We explicitly implement this algorithm in the most generic setting, and we identify heuristics that in practice are seen to mitigate the large constants that so often occur in parameterised complexity, highlighting the practicality of our techniques.

1 Introduction

In combinatorial topology, a triangulated 3-manifold involves abstract tetrahedra whose faces are identified or "glued" in pairs. Many research questions involve looking for a triangulated manifold which fits certain requirements, or is pathologically bad for certain algorithms, or breaks some conjecture. One invaluable tool for such tasks is an exhaustive *census* of triangulated 3-manifolds.

The first of these was the census of cusped hyperbolic 3-manifold triangulations on ≤ 5 tetrahedra by Hildebrand and Weeks [18] in 1989, later extended to ≤ 9 tetrahedra [8, 12, 27]. Another much-used example is the census of closed orientable prime minimal triangulations of ≤ 6 tetrahedra by Matveev [24], later extended to ≤ 12 tetrahedra [22, 23].

In all of these prior works, the authors enumerate all triangulated manifolds on n tetrahedra by first enumerating all 4-regular multigraphs on n nodes (very fast), and then for each graph G essentially modelling every possible triangulation with G as its dual graph (very slow). If any such triangulation built from G is the triangulation of a 3-manifold, we say that G is *admissible*. If G admits a 3-manifold triangulation with some particular property p, we say that G is *p-admissible*.

---

*Supported by the Australian Research Council (DP1094516, DP110101104).

Using state-of-the-art public software [9], generating such a census on 12 tetrahedra takes 1967 CPU-days, of which over 1588 CPU-days is spent analysing non-admissible graphs. Indeed, for a typical census on ≤ 10 tetrahedra, less than 1% of 4-regular graphs are admissible [7]. Moreover, Dunfield and Thurston [17] show that the probability of a random 4-regular graph being admissible tends toward zero as the size of the graph increases. Clearly an efficient method of determining whether a given graph is admissible could have significant effect on the (often enormous) running time required to generating such a census.

We use parameterized complexity [16] to address this issue. A problem is *fixed parameter tractable* if, when some parameter of the input is fixed, the problem can be solved in polynomial time in the input size. In Theorem 14 we show that to test whether a graph G is admissible is fixed parameter tractable, where the parameter is the treewidth of G. Specifically, if the treewidth is fixed at $\leq k$ and G has size n, we can determine whether G is admissible in $O(n \cdot f(k))$ time.

Courcelle showed [14, 13] that for graphs of bounded treewidth, an entire class of problems have fixed parameter tractable algorithms. However, employing this result for our problem of testing admissibility looks to be highly non-trivial. In particular, it is not clear how the topological constraints of our problem can be expressed in monadic second-order logic, as Courcelle's theorem requires. Even if Courcelle's theorem could be used, our results here provide significantly better constants than a direct application of Courcelle's theorem would.

Following the example of Courcelle's theorem, however, we generalise our result to a larger class of problems (Theorem 18). Specifically, we introduce the concept of a *simple property*, and give a fixed parameter tractable algorithm which, for any simple property p, determines whether a graph admits a triangulated 3-manifold with property p (again the parameter is treewidth).

We show that these results are practical through an explicit implementation, and identify some simple heuristics which improve the running time and memory requirements. To finish the paper, we identify a clear potential for how these ideas can be extended to the more difficult enumeration problem, in those cases where a graph *is* admissible and a complete list of triangulations is required.

Parameterised complexity is very new to the field of 3-manifold topology [10, 11], and this paper marks the first exploration of parameterised complexity in 3-manifold enumeration problems. Given that 3-manifold algorithms are often extremely slow and complex, our work here highlights a growing potential for parameterised complexity to offer practical alternative algorithms in this field.

2 Background

To avoid ambiguity with the words "vertex" and "edge", we use the terms *node* and *arc* instead for graphs, and *vertex* and *edge* in the context of triangulations.

Many NP-hard problems on graphs are fixed parameter tractable in the *treewidth* of the graph (e.g., [1, 2, 4, 5, 13]). Introduced by Robertson and Seymour [26], the treewidth measures precisely how "tree-like" a graph is:

Definition 1 (Tree decomposition and treewidth). *Given a graph G, a tree decomposition of G is a tree H with the following additional properties:*

- *Each node of H, also called a* bag*, is associated with a set of nodes of G;*
- *For every arc a of G, some bag of H contains both endpoints of a;*
- *For any node v of G, the subforest in H of bags containing v is connected.*

If the largest bag of H contains k nodes of G, we say that the tree decomposition has width $k + 1$*. The* treewidth *of G, denoted* $\mathrm{tw}(G)$*, is the minimum width of any tree decomposition of G.*

Bodlaender [4] gave a linear time algorithm for determining if a graph has treewidth $\leq k$ for fixed k, and for finding such a tree decomposition, and Kloks [21] demonstrated algorithms for finding "nice" tree decompositions.

A closed 3-manifold is essentially a topological space in which every point has some small neighbourhood homeomorphic to $\mathbb{R}^3$. We first define *general triangulations*, and then give conditions under which they represent 3-manifolds.

Definition 2 (General triangulation). *A general triangulation is a set of abstract tetrahedra $\{\Delta_1, \Delta_2, \ldots, \Delta_n\}$ and a set of face identifications or "gluings" $\{\pi_1, \pi_2, \ldots, \pi_m\}$, such that each π_i is an affine identification between two distinct faces of tetrahedra, and each face is a part of at most one such identification.*

Note that this is more general than a simplicial complex (e.g., we allow an identification between two distinct faces of the same tetrahedron), and it need not represent a 3-manifold. Any face which is not identified to another face is called a *boundary face* of the triangulation. If a triangulation has no such boundary faces, we say it is *closed*. We also note that there are six ways to identify two faces, given by the six symmetries of a regular triangle.

We can partially represent a triangulation by its face pairing graph, which describes *which* faces are identified together, but not *how* they are identified.

Definition 3 (Face pairing graph). *The face pairing graph of a triangulation $\mathcal{T}$ is the multigraph $\Gamma(\mathcal{T})$ constructed as follows. Start with an empty graph G, and insert one node for every tetrahedron in $\mathcal{T}$. For every face identification between two tetrahedra T_i and T_j, insert the arc $\{i, j\}$ into the graph G.*

Note that a face pairing graph will have parallel arcs if there are two distinct face identifications between T_i and T_j, and loops if two faces of the same tetrahedron are identified together. $\mathcal{T}$ is connected if and only if $\Gamma(\mathcal{T})$ is connected.

Some edges of tetrahedra will be identified together as a result of these face identifications (and likewise for vertices). Some edges may be identified directly via a single face identification, while others may be identified indirectly through a series of face identifications.

We assign an arbitrary orientation to each edge of each tetrahedron. Given two tetrahedron edges e and e' that are identified together via the face identifications, we write $e \simeq e'$ if the orientations agree, and $e \simeq \overline{e'}$ if the orientations are reversed. In settings where we are not interested in orientation, we write $e \sim e'$ if the two edges are identified (i.e. one of $e \simeq e'$ or $e \simeq \overline{e'}$ holds).

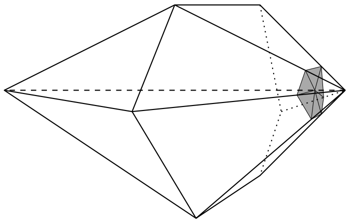


Fig. 1. A triangulation of a 3-ball with 6 tetrahedra meeting along an internal edge.

This leads to the natural notation $[e] = \{e' : e \sim e'\}$ as an equivalence class of identified edges (ignoring orientation). We refer to $[e]$ as an *edge of the triangulation*. Likewise, we use the notation $v \sim v'$ for vertices of tetrahedra that are identified together via the face identifications, and we call an equivalence class $[v]$ of identified vertices a *vertex of the triangulation*.

A *boundary edge / vertex* of a triangulation is an edge / vertex of the triangulation whose equivalence class contains some edge / vertex of a boundary face.

The *link* of a vertex $[v]$ is the (2-dimensional) frontier of a small regular neighbourhood of $[v]$. Figure 1 shows the link of the top vertex shaded in grey; in this figure, the link is homeomorphic (topologically equivalent) to a disc. The link is a 2-dimensional triangulation (in the example it has six triangles), and we use the term *arc* to denote an edge in this triangulation. In this paper, whether "arc" refers to a graph or a vertex link is always clear from context.

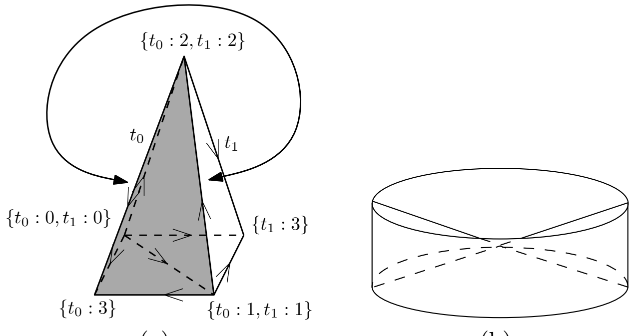
Definition 4 (Closed 3-manifold triangulation). *A closed 3-manifold triangulation* $\mathcal{T}$ *is a general triangulation for which (i)* $\mathcal{T}$ *is connected; (ii) for any vertex* v *in* $\mathcal{T}$*, the link of* v *is homeomorphic to a 2-sphere; and (ii) no edge* e *in* $\mathcal{T}$ *is identified with itself in reverse (i.e.* $e \not\sim \overline{e}$*).*

These properties are necessary and sufficient for the underlying topological space to be a 3-manifold. We say that a graph G is *admissible* if it is the face pairing graph for any closed 3-manifold triangulation $\mathcal{T}$.

Definition 5 (Partial-3-manifold triangulation). *A partial-3-manifold triangulation* $\mathcal{T}$ *is a general triangulation for which (i) for any vertex* v *in* $\mathcal{T}$*, the link of* v *is homeomorphic to a 2-sphere with zero or more punctures; and (ii) no edge* e *in* $\mathcal{T}$ *is identified with itself in reverse (i.e.* $e \not\sim \overline{e}$*).*

These are in essence "partially constructed" 3-manifold triangulations; the algorithms of Section 4.1 build these up into full 3-manifold triangulations. Note that the underlying space of $\mathcal{T}$ might not even be a 3-manifold with boundary: there may be "pinched vertices" whose links have many punctures.

We can make some simple observations: (i) the boundary vertices of a partial 3-manifold triangulation are precisely those whose links have at least one puncture; (ii) a connected partial-3-manifold triangulation with no boundary faces is a closed 3-manifold triangulation, and vice-versa; (iii) a partial-3-manifold triangulation with a face identification removed, or an entire tetrahedron removed, is still a partial-3-manifold triangulation.



(a) (b)

Fig. 2. The triangulation from Example 7. The grey shaded tetrahedron is t_0. Edges are marked with their orientations, and the double-ended arrow indicates the identification of two opposing faces of the pyramid. The resulting space resembles a hockey puck with the centre pinched into a point. This pinch is the vertex $\{t_0{:}2, t_1{:}2\}$.

3 Configurations

The algorithms in Section 4.1 build up 3-manifold triangulations one tetrahedron at a time. As we add tetrahedra, we must track what happens on the boundary of the triangulation, but we can forget about the parts of the triangulation not on the boundary—this is key to showing fixed parameter tractability. In this section we define and analyse edge and vertex configurations of general triangulations, which encode exactly those details on the boundary that we must retain.

Definition 6 (Edge configuration). *The edge configuration of a triangulation $\mathcal{T}$ is a set C_e of triples detailing how the edges of the boundary faces are identified together. Each triple is of the form $((f, e),\ (f', e'),\ o)$, where: f and f' are boundary faces; e and e' are tetrahedron edges that lie in f and f' respectively; e and e' are identified in $\mathcal{T}$; and o is a boolean "orientation indicator" that is true if $e \simeq e'$ and false if $e \simeq \overline{e'}$.*

This mostly encodes the 2-dimensional triangulation of the boundary, though additional information describing "pinched vertices" is still required.

Example 7 (2-tetrahedra pinched pyramid). In all examples, we use the notation $t_i : a$ to denote vertex a of tetrahedron t_i, and $t_i : abc$ to denote face abc of tetrahedron t_i. Face identifications are denoted as $t_i{:}abc \leftrightarrow t_j{:}def$, which means that face abc of t_i is mapped to face def of t_j such that $a \leftrightarrow d$, $b \leftrightarrow e$ and $c \leftrightarrow f$.

Take two tetrahedra t_0 and t_1, each with vertices labelled $0, 1, 2, 3$, and apply the face identifications $t_0{:}012 \leftrightarrow t_1{:}012$ and $t_0{:}023 \leftrightarrow t_1{:}321$.

The resulting triangulation is a square based pyramid with one pair of opposing faces identified (see Figure 2(a)). The final space resembles a hockey puck with a pinch in the centre, as seen in Figure 2(b). Note that the vertex at top of the pyramid, which becomes the pinched centre of the puck, has a link homeomorphic to a 2-sphere with two punctures. Therefore, although this is a partial 3-manifold triangulation, the underlying space is not a 3-manifold.

The edge configuration of this triangulation is:

$$\begin{aligned}
\{((t_0{:}013,03),(t_1{:}013,13),f), &\quad ((t_0{:}013,01),(t_1{:}013,01),t),\\
((t_0{:}013,13),(t_0{:}123,13),t), &\quad ((t_0{:}123,12),(t_0{:}123,23),f),\\
((t_1{:}013,03),(t_1{:}023,03),t), &\quad ((t_1{:}023,02),(t_1{:}023,23),f)\};
\end{aligned}$$

here t and f represent *true* and *false* respectively.

Definition 8 (Vertex configuration). *The vertex configuration C_v of a triangulation $\mathcal{T}$ is a partitioning of those tetrahedron vertices that belong to boundary faces, where vertices v and v' are in the same partition if and only if $v \sim v'$.*

In partial-3-manifold triangulations, vertex links may have multiple punctures; the vertex configuration then allows us to deduce which punctures belong to the same link. In essence, the vertex configuration describes how the triangulation is "pinched" inside the manifold at vertices whose links have too many punctures.

For instance, the vertex configuration of Example 7 is given by

$$\{\{t_0{:}0,t_1{:}0,t_1{:}3\},\quad \{t_0{:}1,t_0{:}3,t_1{:}1\},\quad \{t_0{:}2,t_1{:}2\}\}.$$

The partition $\{t_0 : 2, t_1 : 2\}$ represents the pinch at the center of the "hockey puck".

Definition 9 (Boundary configuration). *The boundary configuration C of a triangulation $\mathcal{T}$ is the pair (C_e, C_v) where C_e is the edge configuration and C_v is the vertex configuration.*

Lemma 10. *For b boundary faces, there are $\frac{(3b)!}{(3b/2)!}$ possible edge configurations.*

Proof Note that b must be even; let $b = 2m$. Each boundary face has three edges, so there are $6m$ possible pairs (f, e) where e is an edge on a boundary face f. Each such pair must be identified with exactly one other pair, with either $e \simeq e'$ or $e \simeq \overline{e'}$, and so the number of possible edge configurations is

$$2 \cdot (6m-1) \cdot 2 \cdot (6m-3) \cdot \ldots \cdot 2 \cdot 3 \cdot 2 \cdot 1 = \frac{(6m)!}{(3m)!} = \frac{(3b)!}{(3b/2)!}. \qquad \square$$

Lemma 11. *For b boundary faces, the number of possible boundary configurations is bounded from above by*

$$\frac{(3b)!}{(3b/2)!} \cdot \left(\frac{2.376b}{\ln(3b+1)}\right)^{3b}.$$

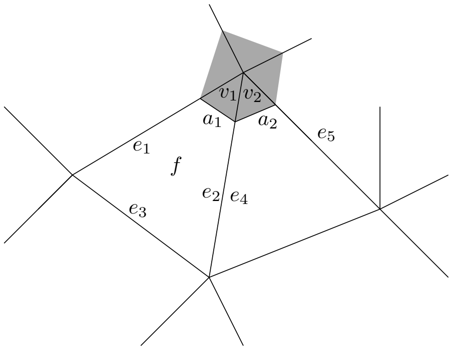


Fig. 3. Part of the boundary of a triangulation. The link of the top vertex is shaded grey; this link does not contain the vertex, but instead cradles the vertex from below.

Proof There are $3b$ tetrahedron vertices on boundary faces, and so the number of possible vertex configurations is the Bell number B_{3b}. The result now follows from Lemma 10 and the following inequality of Berend [3]:

$$B_{3b} = \frac{1}{e} \sum_{i=0}^{\infty} \frac{i^{3b}}{i!} < \left(\frac{2.376b}{\ln(3b+1)} \right)^{3b} . \qquad \square$$

Corollary 12. *The number of possible boundary configurations for a triangulation on n tetrahedra with b boundary faces depends on b, but not on n.*

The boundary configuration can be used to partially reconstruct the links of vertices on the boundary of the triangulation. In particular:

- The edge configuration allows us to follow the arcs around each puncture of a vertex link—in Figure 3 for instance, we can follow the sequence of arcs $a_1, a_2, \ldots$ that surround the puncture in the link of the top vertex.
- The vertex configuration tells us whether two sequences of arcs describe punctures in the *same* vertex link, versus *different* vertex links.

In this way, we can reconstruct all information about punctures in the vertex links, even though we cannot access the full (2-dimensional) triangulations of the links themselves. As the next result shows, this means that the boundary configuration retains all data required to build up a partial-3-manifold triangulation, without knowledge of the full triangulation of the underlying space.

Lemma 13. *Let $\mathcal{T}$ be a partial 3-manifold triangulation with b boundary faces, and let $\mathcal{T}'$ be formed by introducing a new identification between two boundary faces of $\mathcal{T}$. Given the boundary configuration of $\mathcal{T}$ and the new face identification, we can test whether $\mathcal{T}'$ is also a partial-3-manifold triangulation in $O(b)$ time.*

A full proof appears in the full version of this paper. The basic idea is to check whether the conditions in Definition 5 are preserved. The edge configuration allows us to easily test for edges identified together in reverse, and the partial reconstruction of the vertex links (as described above) allows us to test whether all vertex links are still 2-spheres with zero or more punctures.

4 Algorithms and simple properties

Recall that the motivating problem for our work was to quickly detect whether a given graph admits a closed 3-manifold triangulation. To this end we show:

Theorem 14. *Given a connected 4-regular multigraph G, the problem of determining whether there exists a closed 3-manifold triangulation $\mathcal{T}$ such that $\Gamma(\mathcal{T}) = G$ is fixed parameter tractable in the treewidth of G.*

This is a special case of our more general Theorem 18, and so we do not prove it in detail here. The basic idea is as follows.

We say that a boundary configuration C is *viable* for a graph G if there exists some partial-3-manifold triangulation $\mathcal{T}$ with $\Gamma(\mathcal{T}) = G$ and with C as its boundary configuration. Our algorithm starts with an empty triangulation, and then introduces tetrahedra and face identifications in a way that essentially works from the leaves up to the root of the tree decomposition of G. For each subtree in the tree decomposition we compute which configurations are viable for the corresponding subgraph of G, and then propagate these configurations further up the tree. The running time at each node depends only on the number of boundary faces, which is bounded in terms of the bag size and thereby $\mathrm{tw}(G)$.

4.1 A generalisation to simple properties

Here we generalise Theorem 14 to many other settings. For this we define a *simple property* of a partial 3-manifold triangulation (see below).

We extend boundary configurations to include an extra piece of data ϕ based on the partial triangulation that helps test our property. For instance, if p is the simple property that the triangulation contains ≤ 3 internal vertices, then ϕ might encode the number of internal vertices thus far in the partial 3-manifold triangulation (here ϕ takes one of the values $0, 1, 2, 3, \texttt{too\_many}$).

As before: for a simple property p, we say that a boundary configuration C is *p-viable* for a graph G if there exists some partial-3-manifold triangulation $\mathcal{T}$ with property p, with $\Gamma(\mathcal{T}) = G$ and with C as its boundary configuration.

Shortly we solve the problem of testing whether a graph G admits any closed 3-manifold triangulation with property p, for any simple property p. The basic idea is as before: for each subtree of our tree decomposition of G, we compute all viable configurations and propagate this information up the tree.

Definition 15 (Simple property). *A boolean property p of a partial-3-manifold triangulation is called* simple *if all of the following hold. Here all configurations have $\leq b$ boundary faces, and f, g, h are some computable functions.*

1. *The extra data ϕ in the boundary configuration satisfies $\phi \in P$ for some universal set P with $|P| \leq f(b)$.*
2. *We can determine whether a triangulation satisfies p based only on its boundary configuration (including the extra data ϕ).*
3. *Given any viable configuration and a new face identification π between two of its boundary faces, we can in $O(g(b))$ time test whether introducing this identification yields another viable configuration and, if so, calculate the corresponding value of ϕ.*
4. *Given viable configurations for two disjoint triangulations, we can in $O(h(b))$ time test whether the configuration for their union is also viable and, if so, calculate the corresponding value of ϕ.*

The four conditions above can be respectively interpreted as meaning:

1. the upper bound on the number of viable configurations (including the data ϕ) still depends on b but not the number of tetrahedra;
2. we can still test property p without examining the full triangulation;
3. new face identifications can still be checked for p-viability in $O(g(b))$ time;
4. configurations for disjoint triangulations can be combined in $O(h(b))$ time.

Example 16. Let p be the property that a triangulation contains at most x internal vertices (i.e., vertices with links homeomorphic to a 2-sphere), for some fixed integer x. Then p is simple.

Here we define $\phi \in P = \{0, 1, \ldots, x, \texttt{too\_many}\}$ to be the number of vertices in our partial 3-manifold triangulation with 2-sphere links. This clearly satisfies conditions 1 and 2. For condition 3: when identifying two faces together, a new vertex acquires a 2-sphere link if and only if the identification closes off all punctures in the link (which we can test from the edge and vertex configurations). Condition 4 is easily satisfied by summing ϕ over the disjoint configurations.

The case when $x = 1$ is highly relevant: much theoretical and computational work has gone into 1-vertex triangulations of 3-manifolds [19, 23], and these are of particular use when searching for 0-efficient triangulations [20].

We can now state the main result of this paper:

Problem 17 p-ADMISSIBILITY(G) *Let p be a simple property. Given a connected 4-regular multigraph G, determine whether there exists a closed 3-manifold triangulation $\mathcal{T}$ with property p such that $\Gamma(\mathcal{T}) = G$.*

Theorem 18. *Let p be a simple property. Given a connected 4-regular multigraph G on n nodes with treewidth $\leq k$, and a corresponding tree decomposition with $O(n)$ nodes where each bag has at most two children, we can solve p-ADMISSIBILITY(G) in $O(n \cdot f(k))$ time for some computable function f.*

Our requirement for such a tree decomposition is not restrictive: Bodlaender [4] gives a fixed parameter tractable algorithm to find a tree decomposition of width $\leq k$ for fixed k, and Kloks [21] gives an $O(n)$ time algorithm to transform this into a tree decomposition where each bag has at most two children. The "two children" constraint can be relaxed; we use it here to keep the proof simple.

A full proof appears in the full version of this paper; the main ideas are as follows. For each bag ν of the tree decomposition we define a corresponding subgraph G_ν of G, which contains precisely those nodes of G that do *not* appear in bags outside the subtree rooted at ν. As before we use a dynamic programming approach, working from the leaves of the tree decomposition up to the root: for each ν we construct all viable configurations for G_ν, by combining the viable configurations at the child nodes of ν and analysing any new face identifications that might appear. We bound the running time at each ν by a function of the bag size, using the properties of Definition 15 and the observation that any partial triangulation admitted by G_ν must have $\leq 4(\mathrm{tw}(G)+1)$ boundary faces.

Once we reach the root node of the tree decomposition, the final list contains a p-viable configuration if and only if G admits a closed 3-manifold triangulation with property p.

5 Implementation and experimentation

The algorithm was implemented Java, using the treewidth library from [15]. Although our theoretical bound on the number of configurations is extremely large (Lemma 11), we store all configurations using hash maps to exploit situations where in practice the number of viable configurations is much smaller. As seen below, we find that such a discrepancy does indeed arise (and significantly so).

We also introduce another modification that yields significant speed improvements in practice. The algorithm builds up a complete list of all viable configurations at each bag ν of the tree decomposition. However, for an affirmative answer to the problem, only a small subset of these may be required. We take advantage of this as follows.

For any bag ν with no children, configurations are computed as normal. Once a viable configuration is found, it is immediately propagated up the tree in a depth-first manner. This means that, rather than calculating every possible viable configuration for every subgraph G_ν, the improved algorithm can identify a full triangulation with property p quickly and allow early termination.

We implemented the program with p defined to be *one-vertex and possibly minimal*, using criteria on the degrees of edges from [6]. This allowed us to compare both correctness and timing with the existing software *Regina* [9]. We ran our algorithm on all 4-regular graphs on 4, 5 or 6 nodes to verify correctness. We see that the average time to process a graph increases with treewidth, as expected. We also see that the number of viable configurations is indeed significantly lower than the upper bound of Lemma 11, as we had hoped.

Regina significantly outperforms our algorithm on all of these graphs, though these are small problems for which asymptotic behaviour plays a less important role. What matters more is performance on larger graphs, where existing software begins to break down.

We therefore ran a sample of 12-node graphs through our algorithm, selected randomly from graphs which cause significant slowdown in existing software. This "biased" sampling was deliberate—our aim is not for our algorithm to *always* outperform existing software, but instead to seek new ways of solving those difficult cases that existing software cannot handle. Here we do find success: our algorithm was at times 600% faster at identifying non-admissible graphs than *Regina*, though this improvement was not consistent across all trials. More detailed experiments will appear in the full version of this paper.

In summary: for larger problems, our proof-of-concept code already exhibits far superior performance for some cases that *Regina* struggles with. With more careful optimisation (e.g., for dealing with combinatorial isomorphism), we believe that this algorithm would be an important tool that complements existing software for topological enumeration.

The full source code for the implementation of this algorithm is available at `http://www.github.com/WPettersson/AdmissibleFPG`.

6 Applications and extensions

We first note that our meta-theorem is useful: here we list several simple properties p that are important in practice, with a brief motivation for each.

1. *One-vertex triangulations* are crucial for computation: they typically use very few tetrahedra, and have desirable combinatorial properties. This is especially evident with 0-efficient triangulations [20].
2. Likewise, *minimal triangulations* (which use the fewest possible tetrahedra) are important for both combinatorics and computation [6, 7]. Although minimality is not a simple property, it has many simple necessary conditions, which are used in practical enumeration software [7, 23].
3. *Ideal triangulation of hyperbolic manifolds* play a key role in 3-manifold topology. An extension of Theorem 18 allows us to support several necessary conditions for hyperbolicity, which again are used in real software [12, 18].

Finally: a major limitation of all existing 3-manifold enumeration algorithms is that they cannot "piggyback" on prior results for fewer tetrahedra, a technique that has been remarkably successful in other areas such as graph enumeration [25]. This is not a simple oversight: it is well known that we cannot build all "larger" 3-manifold triangulations from smaller 3-manifold triangulations. The techniques presented here, however, may allow us to overcome this issue—we can modify the algorithm of Theorem 18 to store entire families of triangulations at each bag of the tree decomposition. We would lose fixed parameter tractability, but for the first time we would be able to cache and reuse partial results across different graphs and even different numbers of tetrahedra, offering a real potential to extend census data well beyond its current limitations.

References

1. Arnborg, S.: Efficient algorithms for combinatorial problems on graphs with bounded decomposability—a survey. BIT 25(1), 2–23 (1985)
2. Arnborg, S., Lagergren, J., Seese, D.: Easy problems for tree-decomposable graphs. J. Algorithms 12(2), 308–340 (1991)
3. Berend, D., Tassa, T.: Improved bounds on Bell numbers and on moments of sums of random variables. Probab. Math. Statist. 30(2), 185–205 (2010)
4. Bodlaender, H.L.: A linear-time algorithm for finding tree-decompositions of small treewidth. SIAM J. Comput. 25(6), 1305–1317 (1996)
5. Bodlaender, H.L., Kloks, T.: Efficient and constructive algorithms for the pathwidth and treewidth of graphs. J. Algorithms 21(2), 358–402 (1996)
6. Burton, B.A.: Face pairing graphs and 3-manifold enumeration. J. Knot Theory Ramifications 13(8), 1057–1101 (2004)
7. Burton, B.A.: Enumeration of non-orientable 3-manifolds using face-pairing graphs and union-find. Discrete & Computational Geometry 38(3), 527–571 (2007)
8. Burton, B.A.: The cusped hyperbolic census is complete. Preprint, `arXiv:1405.2695` (2014)
9. Burton, B.A., Budney, R., Pettersson, W.: Regina: Software for 3-manifold topology and normal surface theory (1999-2013), `http://regina.sourceforge.net/`
10. Burton, B.A., Lewiner, T., Paixão, J., Spreer, J.: Parameterized complexity of discrete Morse theory. In: SCG '13: Proceedings of the 29th Annual Symposium on Computational Geometry. pp. 127–136. ACM (2013)
11. Burton, B.A., Spreer, J.: The complexity of detecting taut angle structures on triangulations. In: SODA '13: Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, 2013, pp. 168-183
12. Callahan, P.J., Hildebrand, M.V., Weeks, J.R.: A census of cusped hyperbolic 3-manifolds. Math. Comp. 68(225), 321–332 (1999)
13. Courcelle, B., Makowsky, J.A., Rotics, U.: On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic. Discrete Appl. Math. 108(1-2), 23–52 (2001)
14. Courcelle, B.: The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. Inform. and Comput. 85(1), 12–75 (1990)
15. van Dijk, T., van den Heuvel, J.P., Slob, W.: Computing treewidth with LibTW (2006)
16. Downey, R.G., Fellows, M.R.: Parameterized complexity. Monographs in Computer Science, Springer-Verlag, New York (1999)
17. Dunfield, N.M., Thurston, W.P.: Finite covers of random 3-manifolds. Invent. Math. 166(3), 457–521 (2006)
18. Hildebrand, M., Weeks, J.: A computer generated census of cusped hyperbolic 3-manifolds. In: Computers and mathematics (Cambridge, MA, 1989), pp. 53–59. Springer, New York (1989)
19. Jaco, W., Letscher, D., Rubinstein, J.H.: Algorithms for essential surfaces in 3-manifolds. In: Topology and geometry: commemorating SISTAG, Contemp. Math., vol. 314, pp. 107–124. Amer. Math. Soc., Providence, RI (2002)
20. Jaco, W., Rubinstein, J.H.: 0-efficient triangulations of 3-manifolds. J. Differential Geom. 65(1), 61–168 (2003)
21. Kloks, T.: Treewidth, Lecture Notes in Computer Science, vol. 842. Springer-Verlag, Berlin (1994), Computations and approximations

22. Martelli, B., Petronio, C.: Three-manifolds having complexity at most 9. Experimental Mathematics 10(2), 207–236 (2001)
23. Matveev, S.: Algorithmic topology and classification of 3-manifolds, Algorithms and Computation in Mathematics, vol. 9. Springer, Berlin, second edn. (2007)
24. Matveev, S.V.: Computer recognition of three-manifolds. Experiment. Math. 7(2), 153–161 (1998)
25. McKay, B.D.: Isomorph-free exhaustive generation. J. Algorithms 26(2), 306–324 (1998)
26. Robertson, N., Seymour, P.D.: Graph minors. II. Algorithmic aspects of tree-width. J. Algorithms 7(3), 309–322 (1986)
27. Thistlethwaite, M.: Cusped hyperbolic manifolds with 8 tetrahedra. `http://www.math.utk.edu/~morwen/8tet/` (Oct 2010)