Theorems, Algorithms and Brute Force: Building a Census of 3-Manifolds

Benjamin Burton

School of Mathematical and Geospatial Sciences

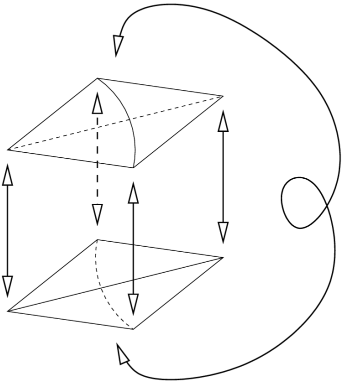
RMIT University
Melbourne, Australia

Outline

Overview: Definitions

Triangulations:

- A 2-dimensional surface can be built by gluing edges of triangles.
- Similarly, a 3-manifold can be built by gluing faces of tetrahedra:



Triangulation of real projective space $\mathbb{R}P^3$

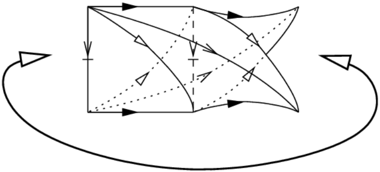
Overview: Definitions

Minimal Triangulations:

- Let M be some 3-manifold. There are many different triangulations that represent M.
- A *minimal triangulation* is a triangulation of M that uses *as few tetrahedra as possible*.

Examples:

- Real projective space $\mathbb{R}P^3$: 2 tetrahedra (from previous slide)
- Non-orientable product $\mathbb{R}P^2 \times S^1$: 3 tetrahedra (below)



- Smallest closed hyperbolic 3-manifold ($v = 0.943$): 9 tetrahedra

Why minimal triangulations?

- Many algorithms in 3-manifold topology are very slow (exponential in the number of tetrahedra)
  $\Rightarrow$ Small triangulations are essential
- Minimal triangulations often have very nice combinatorial structures
  $\Rightarrow$ Useful for studying the underlying 3-manifold

Overview: A Census of 3-Manifolds

The problem:

- List *all 3-manifolds* that can be built using $\leq n$ tetrahedra (like making tables of knots)
- List *all minimal triangulations* of these 3-manifolds

Why?

- Useful for seeking patterns and testing hypotheses
- Required for proving that triangulations are minimal
- Helps with recognising 3-manifolds that you have obtained through other calculations

Difficulties:

- Computations are *very, very slow*
- Not easy to recognise the 3-manifolds from the triangulations

Overview: Previous Census Work

Early work:

- 1989: Cusped hyperbolic manifolds (Hildebrand & Weeks, extended in 1999 with Callahan, data shipped with *SnapPea*)
- 1994: Closed hyperbolic manifolds (Hodgson & Weeks, interested in smallest hyperbolic volume)

Closed orientable manifolds:

- 1998: $\leq$ 6 tetrahedra (Matveev)
- 2001: $\leq$ 9 tetrahedra (Martelli & Petronio)
- 2005: $\leq$ 10 tetrahedra (Matveev / Martelli)

Closed non-orientable manifolds:

- 2002: $\leq$ 6 tetrahedra (Amendola & Martelli)
- 2003: $\leq$ 7 tetrahedra (Amendola & Martelli / Burton)
- 2005: $\leq$ 9 tetrahedra (Burton)

Overview: The Plan

We need to avoid a very large, very slow computer search.

- *Theorems:* Use mathematical theorems to find constraints that minimal triangulations must satisfy;
- *Algorithms:* Combine these theorems with techniques from computer science to improve the efficiency of the search;
- *Brute force:* Throw it all at a very big computer.

And wait…

- 10-tetrahedron non-orientable census: $3\frac{2}{3}$ years CPU time
- In reality, $\sim$ 2 months real time on a large cluster

Theorems: Conditions for Minimality

All results refer to triangulations that are:

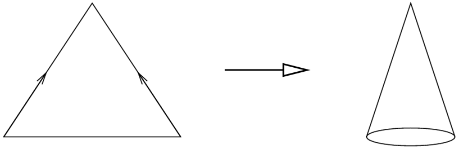
- closed
- minimal
- either orientable or non-orientable
- built from ≥ 3 tetrahedra (avoid small special cases)
- represent irreducible and P^2-irreducible manifolds

Only some theorems are shown here — more results of a similar nature can be proven.

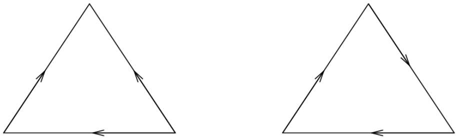
More results $\Rightarrow$ faster algorithms!

Watch how tetrahedron faces become wrapped together in the overall triangulation.

- No face has two of its edges joined together to form a cone:

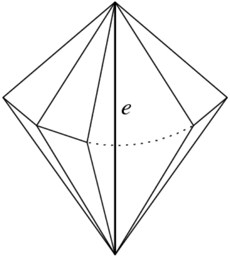


- No face has all three edges joined together:



Theorems: Edge Degrees

The *degree* of an edge is the number of times it appears as an edge of a tetrahedron.



- No edge has degree 1 or 2.
- No edge of degree 3 can meet three different tetrahedra.

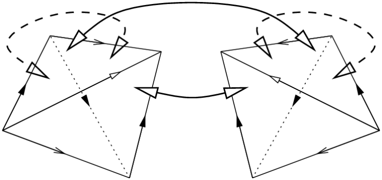
Theorems: Face Pairing Graphs

A *face pairing graph* shows how tetrahedron faces are joined together:

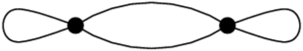
- Graph vertices represent tetrahedra
- Graph edges represent gluings between faces
- Each graph vertex has degree four

Example:

- 2-tetrahedron triangulation of the product $S^2 \times S^1$:

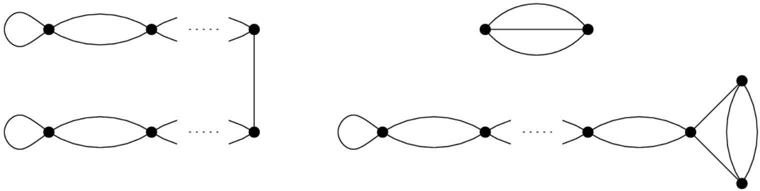


- Corresponding face pairing graph:

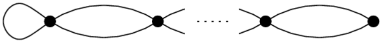


Theorems: Face Pairing Graphs

- No face pairing graph can contain any of the following structures:



- If a face pairing graph contains the following structure, the corresponding tetrahedra are joined to form a *layered solid torus*:



Algorithms: Overall Structure

The overall census algorithm is structured as follows:

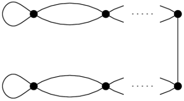
1. Find all possible face pairing graphs.
2. For each face pairing graph, try all possible rotations and reflections for joining pairs of faces together:
   - Six symmetries of the triangle
     $\Rightarrow$ six possibilities for each pair of faces
   - 6^{2t} total possibilities for each face pairing graph

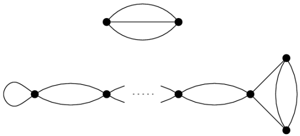
Part (1) is quite fast. Part (2) is extremely slow.

Algorithms: Face Pairing Improvements

Use the face pairing graph theorems:

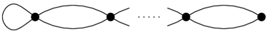
- If a graph contains a bad structure, do not process it at all.





  $\sim$ 50–60% of graphs contain bad structures
  $\Rightarrow$ eliminate $\sim$ 50–60% of running time

- Each time this structure appears in a graph, run through all 2^k layered solid tori instead of all $\frac{1}{6}36^k$ possible gluings of faces:



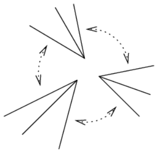
  Even better: reduces asymptotic complexity of running time

Overall improvement (6-tetrahedron non-orientable census):

- 5 weeks $\rightarrow$ 15 hours

Algorithms: Tracking Vertex Links

The neighbourhood of each vertex in the triangulation should be a ball.



Each time we join two faces, calculate new neighbourhoods of the relevant vertices.

- These neighbourhoods will be incomplete, but should be fillable to make a ball
  $\Rightarrow$ neighbourhoods must be orientable
- The final triangulation must have only one vertex (Jaco & Rubinstein / Martelli & Petronio, 2002)
  $\Rightarrow$ make sure that no neighbourhoods are filled in completely before we finish

Algorithms: Tracking Vertex Links

Difficulty:

- Calculating vertex links is slow — we don't want to do this every time we join two faces together!

Solution:

- Use a modification of the *union find* algorithm.

Union find is a sophisticated algorithm for finding connected components in a graph.

- Works by reading in one graph edge at a time and keeping an internal tree structure for each graph component.
- When a graph edge joins two components together, the two trees are merged.

Algorithms: Tracking Vertex Links

Union find has been modified to:

- Allow graph edges to be removed (i.e., allow backtracking in our topological computer search)
- Keep track of useful properties such as orientability of the vertex neighbourhood, how much of the neighbourhood remains to be filled in, etc.

A modified union find can also be used to eliminate low-degree edges and conical faces (see earlier theorems).

Overall improvement (6-tetrahedron non-orientable census):

- 15 hours $\rightarrow 1\frac{1}{2}$ hours using vertex links
- 15 hours $\rightarrow$ 46 seconds using both vertices and edges

Brute Force

Current non-orientable census running times (*hh*:*mm*:*ss*):

Tetrahedra	≤ 5	6	7	8	9	10
Time	0:02	0:46	21:38	17:44:37	28 days	$3\frac{2}{3}$ years
# Manifolds	0	5	3	10	33	≤ 87
# Triang.s	0	24	17	59	307	≤ 983

Code is parallelised to make large cases feasible:

- May run on a cluster of machines
- Embarrassingly parallel
  $\Rightarrow$ k machines means $\sim 1/k$ running time

Work in progress:

- Analysing data from the 10-tetrahedron census
- Improving algorithms to make > 10 tetrahedra feasible

Introducing *Regina*

All computational work done using *Regina*.

- Software package for 3-manifold topology
- Offers full GUI, Python scripting, and command-line tools
- Linux-based (Debian, Fedora, Mandrake, SuSE, others)
- Reads and writes *SnapPea* files
- Full documentation
- Open-source (`regina.sourceforge.net`)

Computes:

- algebraic invariants (π_1, H_1, Turaev-Viro)
- subdivisions, simplifications and decompositions
- combinatorial analysis and recognition of structures
- normal surfaces and angle structures

Further Reading

Census results:

- B. B., *Observations from the 8-tetrahedron non-orientable census*, to appear in Experiment. Math., `math.GT/0509345`, 2005.

Algorithms:

- B. B., *Face pairing graphs and 3-manifold enumeration*, J. Knot Theory Ramifications **13** (2004), 1057–1101.

Software:

- B. B., *Introducing Regina, the 3-manifold topology software*, Experiment. Math. **13** (2004), 267–272.
- *Regina* website, `http://regina.sourceforge.net/`.

Questions?