Unknot recognition and the elusive polynomial time algorithm

Benjamin Burton

The University of Queensland

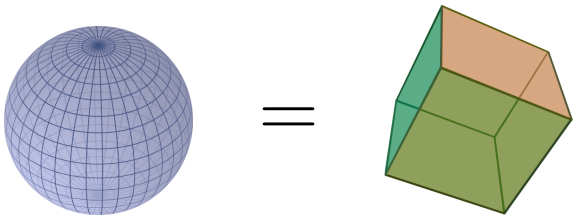
May 18, 2012

What is geometric topology?

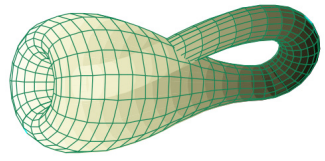
Geometric topology is essentially “rubber-sheet geometry”.

Two topological objects are considered equivalent if we can “bend or stretch” one to make the other.

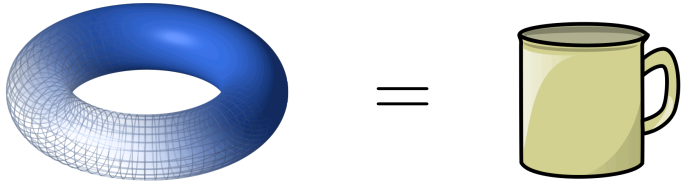
Examples from 2-manifolds (2-dimensional surfaces):



Sphere



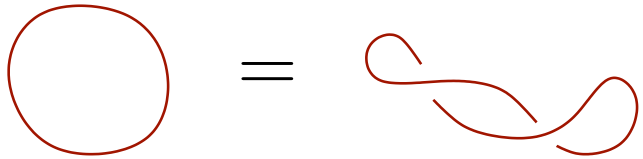
Klein bottle



Torus

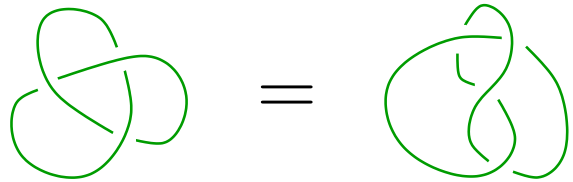
What is geometric topology (ctd.)

Examples from knot theory:



Unknot

Figure 8



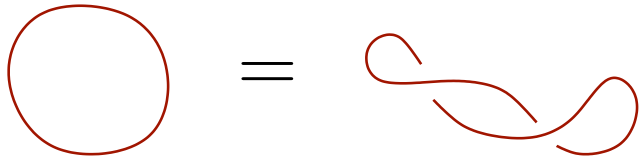
Trefoil

Much research is driven by decision problems:

- Are the 2-manifolds M, N equivalent? . . . Easy!
- Are the knots K, L equivalent? . . . Difficult
- Are the 3-manifolds M, N equivalent? . . . Very difficult

What is geometric topology (ctd.)

Examples from knot theory:



Unknot

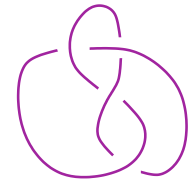
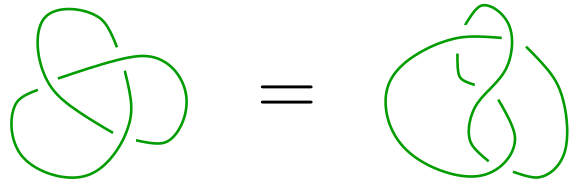


Figure 8



Trefoil

Much research is driven by decision problems:

- Are the 2-manifolds M, N equivalent? . . . Easy!
- Are the knots K, L equivalent? . . . Difficult
- Are the 3-manifolds M, N equivalent? . . . Very difficult
- Are the 4-manifolds M, N equivalent? . . . Undecidable! [Markov, 1960]

We study the simplest cases: Does $M \equiv$ *sphere*? Does $K \equiv$ *unknot*?

2-sphere recognition

Is the 2-manifold (surface) M equivalent to the 2-sphere?

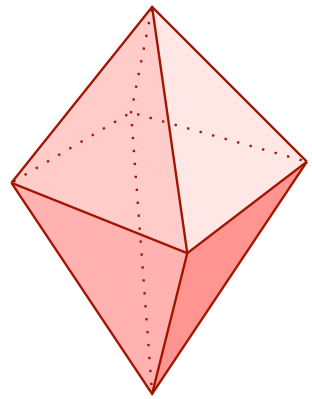
Theorem

For every triangulation of the 2-sphere:

$$\textit{vertices} - \textit{edges} + \textit{faces} = 2.$$

For any triangulation of any other 2-manifold:

$$\textit{vertices} - \textit{edges} + \textit{faces} < 2.$$



$6 - 12 + 8 = 2$

2-sphere recognition algorithm

Triangulate M and test whether $\textit{vertices} - \textit{edges} + \textit{faces} = 2$.

Simple to implement and very fast (small polynomial time).

Unknot and 3-sphere recognition

Is the knot K equivalent to the unknot?

- First algorithm based on normal surface theory [Haken, 1961]
- Later algorithm based on diagram simplification [Dynnikov, 2003]

Is the 3-manifold M equivalent to the 3-sphere?

- First algorithm used almost normal surfaces [Rubinstein, 1992]
- Later algorithm based on Pachner moves [Mijatović, 2003]

Most are messy to implement.

All have at least exponential time in the worst case.

Complexity classes

Do these algorithms need to run in exponential time?

What do we know?

- Unknot recognition is in NP [Hass-Lagarias-Pippenger, 1999]
- 3-sphere recognition is in NP [Schleimer, 2004]
- Knot genus in an arbitrary 3-manifold is NP-complete [Agol-Hass-Thurston, 2002]

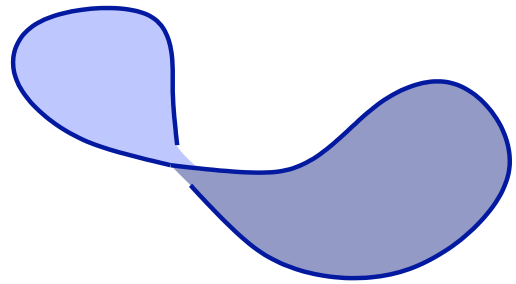
There are hints that unknot / 3-sphere recognition might lie in P . . .

- Unknot recognition also in co-NP (needs GRH) [Kuperberg, 2011]
- . . . as is 3-sphere recognition [Hass-Kuperberg, 2012]
- Bad cases are extremely rare [B., 2010]
- Several "near miss" polynomial-time algorithms, with linear programming as a key tool

Approach #1: Normal surface theory

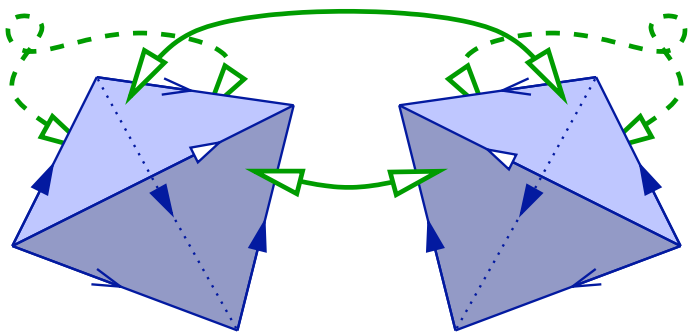
The key idea is to look for interesting surfaces within a 3-D space.

Haken's unknot recognition algorithm:
Find the 2-dimensional disc that the unknot surrounds.



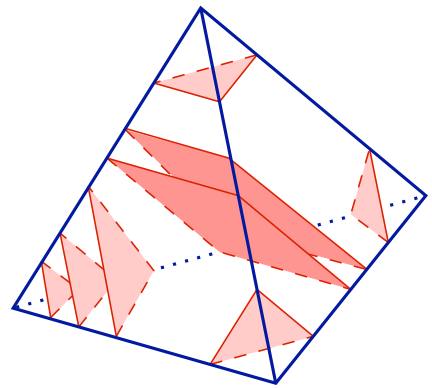
Input: A triangulation of a 3-D space (e.g., drill out the knot from $\mathbb{R}^3$)

- Glue together faces of n tetrahedra (n is the input size).
- Tetrahedra may be "bent" and/or self-identified.

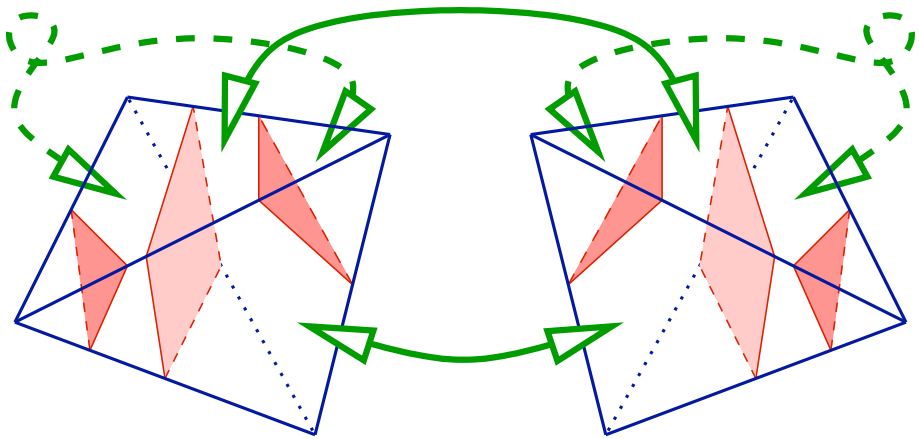


Searching for normal surfaces

We look for embedded normal surfaces.

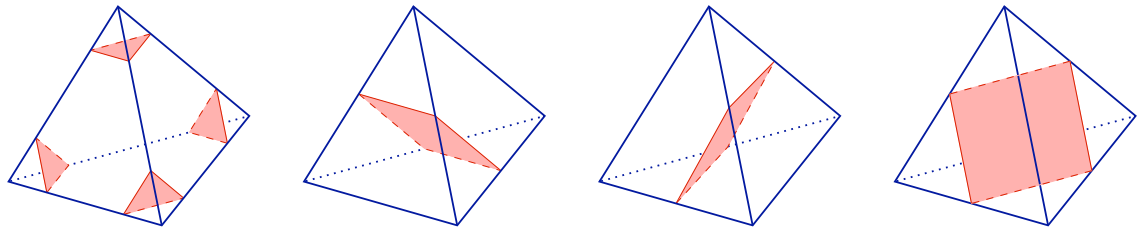


These slice through tetrahedra in triangles and quadrilaterals with no self-intersections.



Normal surfaces as integer vectors

A normal surface can be described by a sequence of $7n$ integers.
These count the discs of each type in each tetrahedron.



This vector uniquely identifies the normal surface.

Theorem (Haken, 1961)

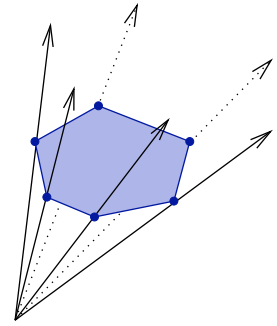
A vector $\mathbf{x} \in \mathbf{Z}^{7n}$ *represents an embedded normal surface if & only if:*

- $\mathbf{x}$ *is non-negative;*
- $\mathbf{x}$ *satisfies a series of linear homogeneous matching equations;*
- $\mathbf{x}$ *uses at most one quadrilateral type per tetrahedron (the quadrilateral constraints).*

The magic

The projective solution space is a cross-section of the cone described by $\mathbf{x} \geq \mathbf{0}$ and the matching equations.

This is a rational polytope.



Theorem (Haken, 1961; Jaco-Tollefson, 1984)

If the knot spans a disc, then it spans a normal disc that projects to a vertex of the projective solution space.

Unknot recognition algorithm

Enumerate the vertices of the projective solution space.
If a vertex satisfies the quadrilateral constraints, reconstruct the surface and test whether it is the disc that we are looking for.

3-sphere recognition uses similar techniques.

Can we do this in polynomial time?

We cannot enumerate all vertices in polynomial time:

Pathological cases exist with $O(17^{n/4})$ vertices that all satisfy the quadrilateral constraints. [B., 2010]

Lemma

For every polygonal decomposition of a disc,

$$\textit{vertices} - \textit{edges} + \textit{faces} = 1.$$

For any polygonal decomposition of any other bounded surface,

$$\textit{vertices} - \textit{edges} + \textit{faces} \leq 0.$$

Observation: *vertices* $-$ *edges* $+$ *faces* is linear on the solution space!

Corollary

We have the unknot if and only if $\max(\textit{vertices} - \textit{edges} + \textit{faces}) > 0$ *under the quadrilateral constraints.*

Linear programming and the projective solution space

This sounds like a job for linear programming!

The problem is the quadrilateral constraints, which are non-linear and have a non-convex solution set.

Workarounds:

- Run 3^n distinct linear programs on the 3^n convex pieces that make up this solution set. [Casson, ∼2002]

  This is always slow, since all 3^n steps are necessary if the input knot is non-trivial.

- Add integer and binary variables to enforce the quadrilateral constraints. [B.-Ozlen, 2011]

  This is fast in practice, but requires integer programming which is non-polynomial in general.

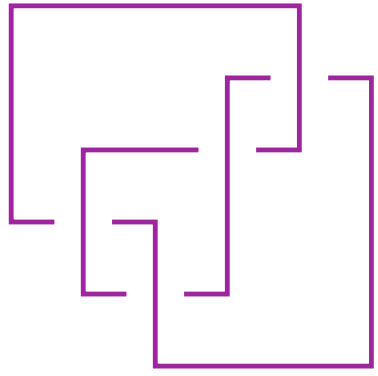
Approach #2: Diagram simplification

Try to monotonically simplify a knot diagram / triangulation into its simplest possible form.

Grid diagrams for knots:

Constructed from n horizontal rods and n vertical rods.

Vertical rods always cross above horizontal rods.

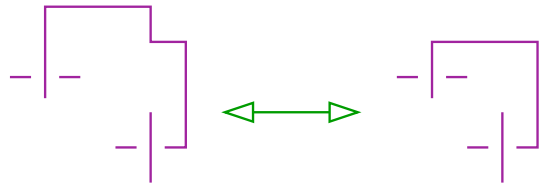


Theorem (Dynnikov, 2003)

Any two grid diagrams of the same knot can be related by elementary moves.

Simplifying grid diagrams

Some elementary moves reduce n:



Some elementary moves leave n unchanged:

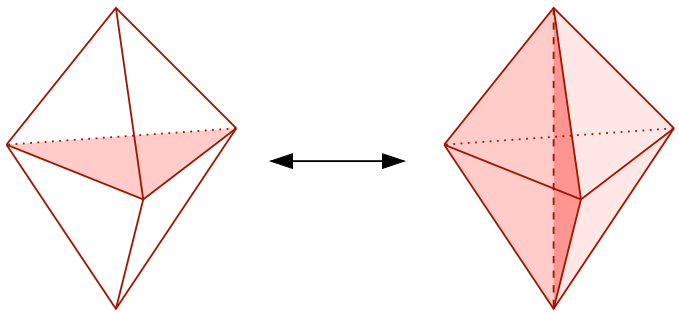
Theorem (Dynnikov, 2003)

For any grid diagram of the unknot, there is a non-strict monotonic sequence of simplification moves that reduces the diagram to the trivial square.

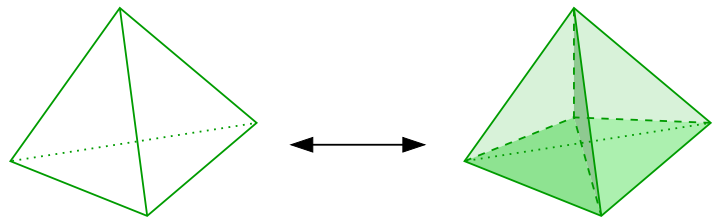
If non-strict could be made strict, this would yield a polynomial time algorithm!

3-sphere recognition by Pachner moves

Any two triangulations of the same 3-manifold
can be related by Pachner moves: [Pachner, 1991]



2-3 / 3-2 move



1-4 / 4-1 move

The same is true if we consider only
one-vertex triangulations and 2-3 / 3-2 moves. [Matveev, 1988]

Simplifying by Pachner moves

In theory: Triangulations might become much larger along the way.

Current best bound: $6 \cdot 10^6 n^2 2^{2 \cdot 10^4 n^2}$ moves [Mijatović, 2003]

In practice:

Computer theorem (B., 2011)

For all $n = 3, \ldots, 9$, any 3-sphere triangulation of size n can be simplified in ≤ 9 moves, and by passing through ≤ 2 extra tetrahedra.

This was shown by enumerating and analysing all $149,676,922$ distinct 3-manifold triangulations of size $n \leq 9$.

Does this help?

If we could turn these experimental bounds into theoretical bounds...

3-sphere recognition algorithm

Try all possible sequences of $\leq B$ moves, where B is our theoretical bound.

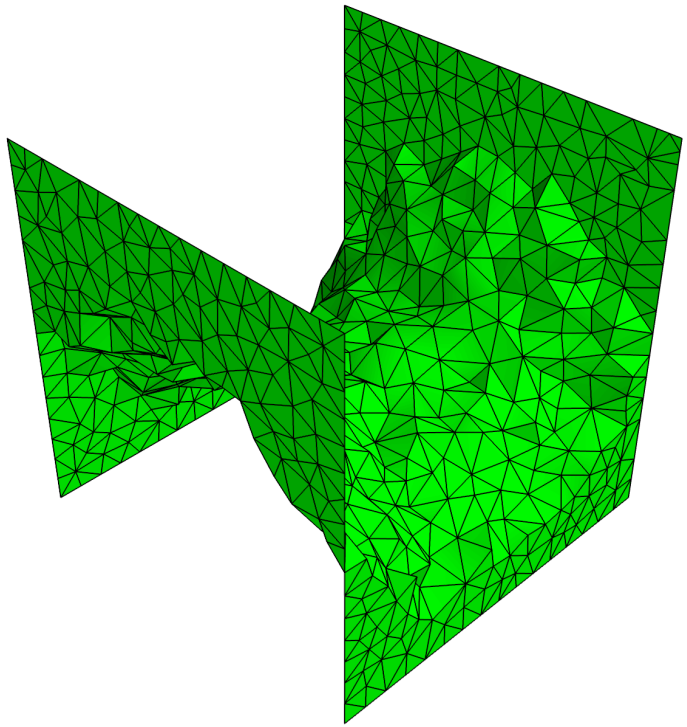
If this simplifies the triangulation, repeat. If not, "read off" whether we have a 3-sphere.

If B grows slower than $O(n/\log n)$, this yields a sub-exponential time algorithm.

If B grows like $O(1)$, this yields a polynomial time algorithm!

Approach #3: Integer programming over homology

New problem: **Least area surface** bounded by a knot



SOURCE: DUNFIELD AND HIRANI, 2010

Consider a discrete version:

- triangulate the space so the knot follows edges;
- find a least area surface built from faces of the triangulation.

Finding the least area surface

Theorem (Dunfield-Hirani, 2010)

In this discrete setting, the least area surface can be found in polynomial time.

Basic idea:

- Describe a surface as a sum of faces (triangles).

  If our triangulation has n faces, this gives an integer vector in $\mathbb{Z}^n$.
- Express "the triangles form a surface" using linear constraints:

  Each triangle going into an edge must meet some triangle going out of an edge.
- Express area as a linear functional on $\mathbb{Z}^n$.
- Minimise this linear functional using integer programming.

Achieving polynomial time

Theorem (Dey-Hirani-Krishnamoorthy, 2010)

In this setting, the constraint matrix for the integer program is totally unimodular.

This means that we can relax the integer program to a linear program, which can be solved in polynomial time.

Unfortunately:

Theorem (Hass-Snoeyink-Thurston, 2003)

Even if the knot spans a disc, the least area surface might not be a disc.

If only we could express $\textit{vertices} - \textit{edges} + \textit{faces}$ as a linear functional on triangles, we could recognise the unknot in polynomial time!

Generic complexity

If at first you fail . . .

Generic complexity: ignore a few bad cases, where $\Pr(\text{bad}) \to 0$ as $n \to \infty$.

Exhaustive analysis of all $1,537,582,427$ closed 1-vertex triangulations with $n \leq 10$ suggests:

$$\Pr(\text{bad}) \in O(1/n^c) \quad \text{for all } c > 0,$$

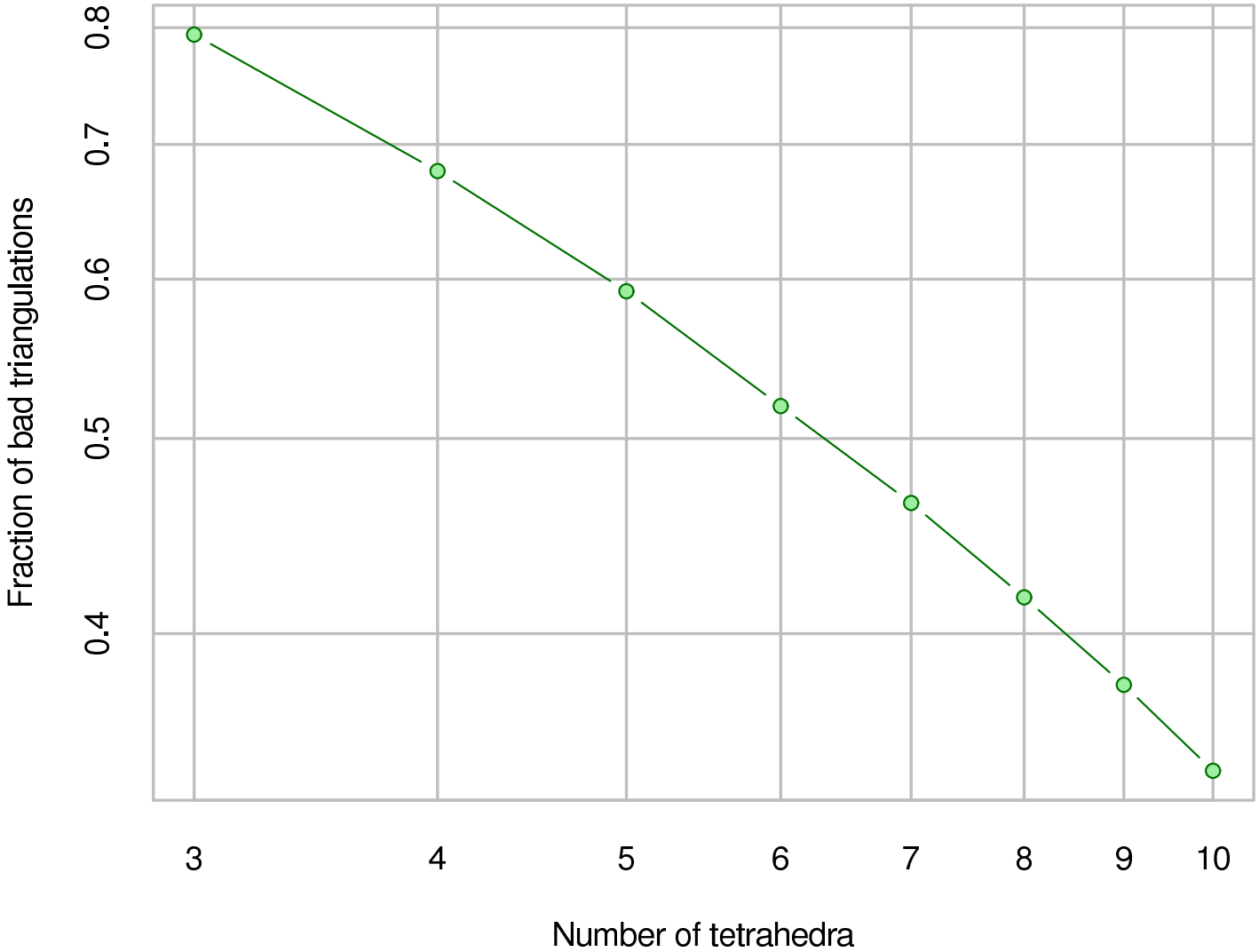
where "bad" means "does not simplify immediately". That is:

Experimental observation

Generic triangulations simplify immediately!

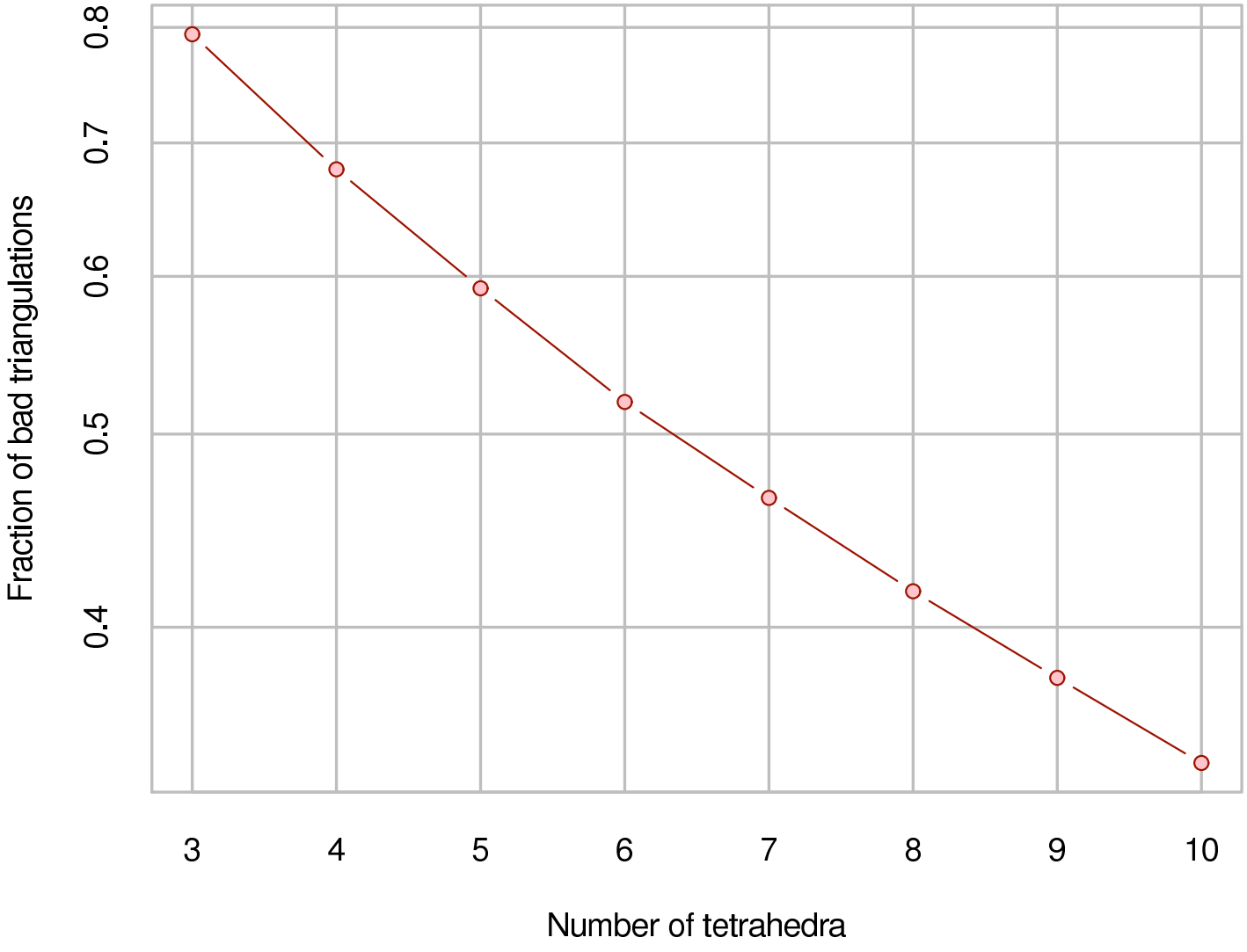
Triangulations that do NOT simplify immediately

$1,537,582,427$ closed 1-vertex triangulations, log-log scale:



Triangulations that do NOT simplify immediately

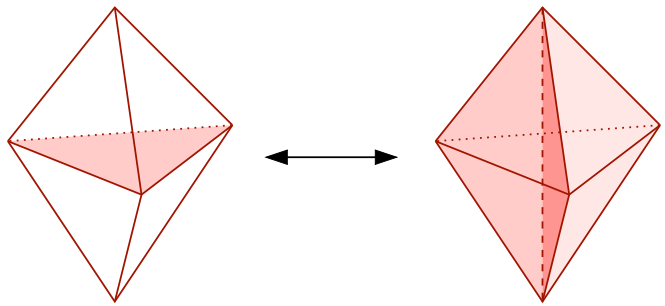
1, 537, 582, 427 closed 1-vertex triangulations, linear-log scale:



Aggressive simplification

In practice, simplification is an extremely effective heuristic in 3-sphere recognition and related problems.

To find a k-move simplification requires $O(n^k)$ steps.



Observation

Suppose we allow $O(n^k)$ time to simplify from $t \to t-1$ tetrahedra.

As t drops, the number of moves can grow:

$$n^k = t^{(k \cdot \log n / \log t)}$$

Aggressive simplification (ctd.)

Observation (previous slide)

As t drops, the number of moves can grow:

$$n^k = t^{(k \cdot \log n / \log t)}$$

That is, we can become more aggressive in our simplification as the triangulation shrinks.

$\longrightarrow$ The difficult small cases become simpler!

Under the right "approximate independence" assumptions:

Conjecture

Generic 3-sphere triangulations can be simplified to the trivial case $n = 2$ in polynomial time.

Want to know more?

Normal surface algorithms

Hass, Lagarias & Pippenger, *The computational complexity of knot and link problems*, J. ACM **46** (1999), no. 2, 185–211

B. & Ozlen, *Computing the crosscap number of a knot using integer programming and normal surfaces*, ACM TOMS, `arXiv:1107.2382`

Simplification-based algorithms

Dynnikov, *Recognition algorithms in knot theory*, Russian Math. Surveys **58** (2003), no. 6(354), 45–92

B., *The Pachner graph and the simplification of 3-sphere triangulations*, SoCG 2011, `arXiv:1011.4169`

Least area surface

Dunfield & Hirani, *The least spanning area of a knot and the optimal bounding chain problem*, SoCG 2011, `arXiv:1012.3030`