Computing the crosscap number of a knot using integer programming and normal surfaces

Burton, Benjamin; Ozlen, Melih

Burton, B., & Ozlen, M. (2012). Computing the crosscap number of a knot using integer programming and normal surfaces. ACM Transactions on Mathematical Software, 39(1), 1–19.
https://doi.org/10.1145/2382585.2382589
Document Version: Accepted Manuscript

Published Version: https://doi.org/10.1145/2382585.2382589

Computing the crosscap number of a knot using integer programming and normal surfaces

Benjamin A. Burton and Melih Ozlen

March 4, 2012

Abstract

The crosscap number of a knot is an invariant describing the non-orientable surface of smallest genus that the knot bounds. Unlike knot genus (its orientable counterpart), crosscap numbers are difficult to compute and no general algorithm is known. We present three methods for computing crosscap number that offer varying trade-offs between precision and speed: (i) an algorithm based on Hilbert basis enumeration and (ii) an algorithm based on exact integer programming, both of which either compute the solution precisely or reduce it to two possible values, and (iii) a fast but limited precision integer programming algorithm that bounds the solution from above.

The first two algorithms advance the theoretical state of the art, but remain intractable for practical use. The third algorithm is fast and effective, which we show in a practical setting by making significant improvements to the current knowledge of crosscap numbers in knot tables. Our integer programming framework is general, with the potential for further applications in computational geometry and topology.

1 Introduction

Knot invariants lie at the heart of computational knot theory. Nevertheless, computing invariants can be challenging: algorithms often require complex implementations and exponential running times, and for some invariants no general algorithm is known.

In this paper we focus on invariants of knots in $\mathbb{R}^3$ that relate to 2-dimensional surfaces: *knot genus* and *crosscap number*. In essence, these measure the simplest embedded orientable and non-orientable surfaces respectively that have a single boundary curve following the knot.

Knot genus is well-studied: algorithms are known [15], and precise values have been computed for all prime knots with ≤ 12 crossings [10]. In contrast, although crosscap numbers can be computed for special classes of knots [17, 18, 27], no algorithm is known for computing them in general. Of all 2977 non-trivial prime knots with ≤ 12 crossings, only 289 have crosscap numbers that are known precisely [10].

The crosscap number displays unusual behaviours that knot genus does not, and embodies different information [11, 26]. It is therefore desirable to compute crosscap numbers in a general setting. Here we develop three algorithms with varying precision-to-speed trade-offs, which yield both theoretical and practical advances.

Our algorithms do not guarantee to compute the crosscap number precisely for every input (this remains an open problem), but they do come close. The first two algorithms are significant theoretical advances: they either compute the crosscap number precisely or reduce it to one of

two possible values. The third algorithm is a significant practical achievement: although it only outputs an upper bound on the crosscap number, its strong practical performance combined with known lower bounds allows us to make significant improvements to crosscap numbers in existing tables of knots.

The first algorithm, described in Section 4, uses Haken's normal surface theory [14] to reduce the computation of crosscap number to a Hilbert basis enumeration over a high-dimensional polyhedral cone. Although this general approach follows a common template in computational topology, there are complications that cause the usual theoretical techniques to fail. To address this, we introduce a special class of triangulations called *suitable triangulations*, described in Section 3, with which we are able to solve these theoretical problems.

The second and third algorithms, described in Sections 5 and 6, draw on techniques from discrete optimisation theory. Here we develop an integer programming framework that allows us to approach problems in computational topology using off-the-shelf optimisation software. The difference between the second and third algorithms is that the second requires expensive exact integer arithmetic; the third makes concessions that allow us to use fast off-the-shelf solvers based on floating point computation, but with the side-effect that it only outputs an upper bound.

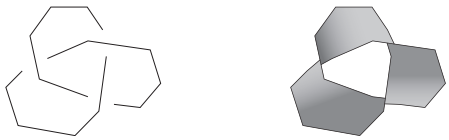
The first two algorithms remain intractable for all but the simplest knots, though each has different strengths through which it may become practical with the growth of supporting software in algebra and optimisation. In contrast the third algorithm is extremely fast, and in Section 7 we run it over the full 12-crossing knot tables. The results are extremely pleasing: of the 2688 knots with unknown crossing number, for 747 we can improve the best-known bounds, and for a further 27 we can combine our output with known lower bounds to compute the crosscap number precisely.

The integer programming framework that we introduce here is general, and has significant potential for use elsewhere in computational geometry and topology. We discuss these matters further in Section 7.

2 Preliminaries

Here we give a short summary of concepts from knot theory and normal surface theory that appear within this paper. This outline is necessarily brief; for details see the excellent overview in [15].

For our purposes, a knot is a piecewise linear closed curve embedded in $\mathbb{R}^3$. We also treat knots as being embedded in the 3-sphere S^3, where S^3 is the one-point compactification $\mathbb{R}^3 \cup \{\infty\}$. A knot K is typically given as a *knot diagram*, which is a general position projection of the knot onto the plane as illustrated in Figure 1(a). See [15] for more precise definitions of these concepts.



(a) (b)

Figure 1: A knot diagram and a spanning surface for the trefoil knot

We are interested in properties of knots related to two-dimensional surfaces. In this paper all surfaces are piecewise linear, and might be disconnected unless otherwise stated. A surface with no boundary curves (such as a sphere or a torus) is called *closed*, and a surface with boundary curves (such as a disc or a Möbius band) is called *bounded*.

Every closed orientable surface is topologically a 2-sphere with $g \geq 0$ orientable "handles" attached (i.e., a g-holed torus); the *orientable genus* of such a surface is g. Every closed non-orientable surface is a 2-sphere with $g \geq 1$ "crosscaps" attached; the *non-orientable genus* of such a surface is g. We simply use the word *genus* when orientability is clear from context. If an (orientable or non-orientable) surface S has boundary, then the (orientable or non-orientable) genus of S is the genus of the closed surface obtained by filling each boundary curve of S with a disc.

Let $\mathcal{P}$ be a polygonal decomposition of a surface S. The *Euler characteristic* of $\mathcal{P}$ is defined as $V - E - F$, where V, E and F represent the number of vertices, edges and faces of $\mathcal{P}$ respectively. Euler characteristic is a topological invariant of a surface, and is denoted by $\chi(S)$. The Euler characteristic of an orientable surface of genus k with b boundary curves is $2-2k-b$, and the Euler characteristic of a non-orientable surface of genus k with b boundary curves is $2-k-b$.

The knot invariants that we study here relate to surfaces embedded in 3-dimensional space, which leads us to study 3-manifolds. A *3-manifold* is a higher-dimensional analogue of a surface: each interior point of a 3-manifold M has a local neighbourhood that is topologically similar to $\mathbb{R}^3$, and each point on the boundary of M has a local neighbourhood that is topologically similar to the closed half-space $\mathbb{R}^3_{z\geq 0}$. Again, see [15] for precise details. A surface $S \subset M$ is *embedded* in M if it has no self-intersections, and *properly embedded* if in addition the boundary of S lies within the boundary of M, and the interior of S lies within the interior of M.

Consider a knot K embedded in S^3. A *spanning surface* for K is a connected embedded surface in S^3 whose boundary is precisely K. Figure 1(b) illustrates a spanning surface for the trefoil knot (this surface is a Möbius band, with non-orientable genus 1).

We can now define the following two invariants of a knot K. The *genus* of K, denoted $g(K)$, is the smallest k for which there exists an orientable genus k spanning surface for K. Likewise, the *crosscap number* of K, denoted $C(K)$, is the smallest k for which there exists a non-orientable genus k spanning surface for K. As a special case, the crosscap number of the trivial knot (also called the *unknot*) is defined to be 0. The unknot is the only knot with genus 0, and the only knot with crosscap number 0. A key relation between genus and crosscap number is the following [11]:

Theorem 2.1 (Clark's inequality)**.** *For any knot K, it is true that $C(K) \leq 2g(K) + 1$.*

Let $K \subset S^3$ be a knot, and let R be a small regular neighbourhood of K in S^3. The *complement* of K, denoted $\overline{K}$, is the closure of $S^3 \backslash R$. This is a 3-manifold with boundary, obtained by "eating away" the knot from S^3, as illustrated in Figure 2(a). The boundary surface of $\overline{K}$ is a torus, and any curve on this torus that bounds a disc in R is called a *meridian*.

We can reformulate spanning surfaces in terms of knot complements. Consider a knot $K \subset S^3$ and some meridian m on the boundary of $\overline{K}$. A *spanning surface* in $\overline{K}$ is a connected, properly embedded surface $S \subset \overline{K}$ with precisely one boundary curve c, where c and m have algebraic intersection number ± 1 on the torus boundary of $\overline{K}$.[1] See Figure 2(b) for an illustration. This is essentially the same as our previous definition, since any such surface can be extended through

[1] Essentially we require that the boundary of S cuts m precisely once. Using algebraic intersection number allows us to account for any extra trivial "wiggles" back and forth across m.

Figure 2: The complement of the trefoil knot and a spanning surface within it

R to give a connected embedded surface bounded by K and vice versa, assuming that the neighbourhood R is sufficiently small.

All of our algorithms work with triangulations of the complement $\overline{K}$. In this paper, a *triangulation* of a 3-manifold is a collection of n tetrahedra, some of whose $4n$ faces are affinely identified in pairs. This broad definition allows for smaller triangulations than a traditional simplicial complex; in particular, most triangulations in this paper are *one-vertex triangulations*, where all $4n$ tetrahedron vertices are (as a result of the face gluings) identified to a single point in $\overline{K}$.

More generally, if $\mathcal{T}$ is a 3-manifold triangulation, all tetrahedron vertices that are identified to a single point in $\mathcal{T}$ are collectively referred to as a single *vertex* of $\mathcal{T}$; similarly for edges and faces. Any vertex, edge or face that lies in the boundary of the underlying 3-manifold is called a *boundary* vertex, edge or face; all others are referred to as *internal*. Note that the boundary faces of $\mathcal{T}$ are precisely those tetrahedron faces that are not paired with some partner face.

In our algorithms we describe spanning surfaces using normal surface theory. A *normal surface* in $\mathcal{T}$ is a properly embedded surface that meets each tetrahedron Δ of $\mathcal{T}$ in a disjoint collection of triangles and quadrilaterals, each running between distinct edges of Δ, as illustrated in Figure 3. There are four *triangle types* and three *quadrilateral types* according to which edges they meet. Within each tetrahedron there may be several triangles or quadrilaterals of any given type; collectively these are referred to as *normal discs*.

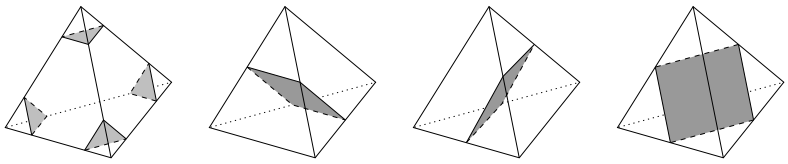


Figure 3: Normal triangles and quadrilaterals within a tetrahedron

The *vector representation* of a normal surface S is the $7n$-dimensional integer vector

$$\mathbf{v}(S) = (t_{1,1}, t_{1,2}, t_{1,3}, t_{1,4},\ q_{1,1}, q_{1,2}, q_{1,3}\ ;\ t_{2,1}, t_{2,2}, t_{2,3}, t_{2,4},\ q_{2,1}, q_{2,2}, q_{2,3}\ ;\ \ldots, q_{n,3}\ ) \in \mathbb{Z}^{7n},$$

where $t_{i,j}$ is the number of triangles in the ith tetrahedron of the jth type, and $q_{i,k}$ is the number of quadrilaterals in the ith tetrahedron of the kth type ($1 \leq i \leq n$, $1 \leq j \leq 4$, $1 \leq k \leq 3$).

Theorem 2.2 (Haken, 1961)**.** *A vector $\mathbf{v} = (t_{1,1}, \ldots, q_{n,3}) \in \mathbb{R}^{7n}$ is the vector representation of a normal surface if and only if: (i) all elements of $\mathbf{v}$ are non-negative integers; (ii) $\mathbf{v}$ satisfies*

a certain set of homogeneous linear equations derived from the triangulation, and (iii) for each i, at most one of the three quadrilateral coordinates $q_{i,1}, q_{i,2}, q_{i,3}$ is non-zero.

The homogeneous linear equations in (ii) are called the *matching equations*, and condition (iii) is called the *quadrilateral constraints*. The set of all points in $\mathbb{R}^{7n}$ with non-negative coordinates that satisfy the matching equations is called the *normal surface solution cone*.

Given a vector that satisfies all of the constraints of Theorem 2.2, the corresponding normal surface can be reconstructed uniquely (up to normal isotopy). If X and Y are both normal surfaces in some triangulation, the *normal sum* $X + Y$ is the normal surface with vector representation $\mathbf{v}(X) + \mathbf{v}(Y)$. It is possible that $\mathbf{v}(X) + \mathbf{v}(Y)$ does not satisfy the quadrilateral constraints, in which case (for our purposes) the sum $X + Y$ is not defined.

A *fundamental normal surface* S is one that cannot be written as $S = X + Y$ for non-empty normal surfaces X, Y. There are finitely many fundamental normal surfaces in a triangulation, corresponding precisely to the vectors in the Hilbert basis of the normal surface solution cone.

3 Suitable triangulations

To overcome theoretical difficulties that arise with non-orientable spanning surfaces, we introduce a special class of triangulations for our algorithms to use.

Definition. A *suitable triangulation* $\mathcal{T}$ of a knot complement is one for which:

(i) $\mathcal{T}$ has precisely one vertex (and therefore the torus boundary of $\mathcal{T}$ contains this one vertex, three edges and two faces);[2]

(ii) one of the boundary edges of $\mathcal{T}$ is a meridian.

Given a triangulation $\mathcal{T}$ of a knot complement, it is easy to test whether $\mathcal{T}$ is suitable. Condition (i) can be verified by grouping vertices of tetrahedra into equivalence classes under identification. For condition (ii), we can verify that a boundary edge e is a meridian by attaching a solid torus in an appropriate fashion and testing whether the resulting closed manifold is a 3-sphere.[3]

Our first two algorithms for computing crosscap number require a further condition:

Definition. An *efficient suitable triangulation* $\mathcal{T}$ of a knot complement is a suitable triangulation that contains no embedded normal 2-spheres.

This extra 2-sphere condition is related to, though weaker than, the 0-efficiency criterion of Jaco and Rubinstein [21]. Testing for efficient suitability is possible but slow: to verify the absence of embedded normal 2-spheres one must typically enumerate all extreme rays of the high-dimensional normal surface solution cone [21].

We give two algorithms for producing a suitable triangulation of a knot complement. The first is general but slow; the second is heuristic in nature but works extremely well in practice.

Algorithm 3.1. *Given a knot diagram describing the knot $K \subseteq S^3$, the following procedure will output an efficient suitable triangulation of $\overline{K}$.*

[2] This follows by a standard Euler characteristic argument.

[3] The solid torus must be attached so that its meridional disc is bounded by e; this can be done using a $(1, 1, 0)$ layered solid torus as described in [22]. Although the subsequent 3-sphere test requires worst-case exponential time, it is found to run surprisingly fast in practice when the right algorithms and simplification heuristics are used [5, 6].

1. *Test whether K is the unknot [14, 24]. If so, output a pre-constructed efficient suitable triangulation of the unknot complement and terminate immediately.*

2. *Run the procedure of Hass, Lagarias and Pippenger [15, Lemma 7.2] to obtain a triangulation $\mathcal{T}$ of $\overline{K}$, along with a meridian expressed as a path that follows boundary edges of $\mathcal{T}$.*

3. *Run the procedure of Jaco and Rubinstein [21, Proposition 5.15 and Theorem 5.20] to convert this into a one-vertex triangulation with no embedded normal 2-spheres. Keep track of the location of the meridian as the Jaco-Rubinstein procedure runs.*

4. *Apply layerings of extra tetrahedra to alter the boundary edges until the meridian consists of a single boundary edge, and output the resulting triangulation.*

   Each layering attaches a new tetrahedron along two boundary faces as illustrated in Figure 4. The number and locations of these layerings are determined by a continued fraction calculation, as described by Jaco and Rubinstein [22, 23].

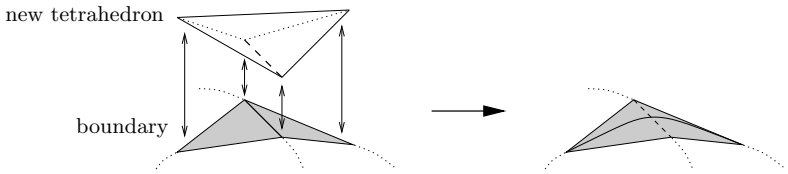


Figure 4: Layering a new tetrahedron onto the boundary

The unknot test in step 1 is necessary because the Jaco-Rubinstein procedure in step 3 requires $\overline{K}$ to be both irreducible and boundary irreducible, which is only true for non-trivial knots [16]. If K is the unknot, the one-tetrahedron solid torus [21] can be used as a pre-constructed solution.

Theorem 3.2. *Algorithm 3.1 is correct, i.e., it produces an efficient suitable triangulation of $\overline{K}$.*

Proof. For the correctness of each sub-procedure we refer the reader to the source papers [14, 15, 21, 22, 23, 24]. Here we simply prove that the requirements for an efficient suitable triangulation are satisfied.

The Jaco-Rubinstein procedure in step 3 gives a triangulation with no embedded normal 2-spheres and just one vertex. The only missing requirement is that some boundary edge is a meridian. The layering process in step 4 fixes this, and does not break the other requirements:

- Each layering preserves the number of vertices of the triangulation.

- The layering process does not introduce any new embedded normal 2-spheres. Each time we layer a new tetrahedron onto the boundary, every normal triangle or quadrilateral in this new tetrahedron meets the boundary of the new triangulation. Therefore any embedded normal 2-sphere in the new triangulation cannot use these new normal discs, and must have been an embedded normal 2-sphere in the old triangulation also. □

Steps 1 and 3 of the previous algorithm are slow; although step 1 can be avoided (e.g., by using prior information that the input knot is non-trivial), the Jaco-Rubinstein procedure of step 3 remains too inefficient to use with all but the simplest knot complements.

We therefore offer an alternative, heuristic algorithm that uses only fast (small polynomial time) operations. The drawback is that this heuristic algorithm might not produce any solution at all; however, experience shows this to be a rare occurrence, as discussed below.

Algorithm 3.3. *Given a knot diagram describing the knot $K \subseteq S^3$, the following heuristic procedure will either output a suitable triangulation of $\overline{K}$ or terminate with no output at all.*

1. *Run the procedure of Hass, Lagarias and Pippenger [15, Lemma 7.2] to obtain a triangulation $\mathcal{T}$ of $\overline{K}$, along with a meridian expressed as a path that follows boundary edges of $\mathcal{T}$.*

2. *Simplify the triangulation using fast local operations (such as edge collapses, Pachner moves, book closing moves and related operations [4, 9]) to reduce the number of tetrahedra and the number of boundary faces as far as possible.*

3. *Test whether the resulting triangulation $\mathcal{T}'$ is suitable. If so then output $\mathcal{T}'$, and otherwise terminate with no output.*

The choice of local simplification operations in step 2 is not important; see [4, 9] for details. Although the suitability test in step 3 requires 3-sphere recognition (which runs in worst-case exponential time), we can sidestep this by tracking the location of the meridian throughout step 2, avoiding the need to verify the meridian condition (and test for 3-spheres) in step 3.

In Section 7 we observe that this heuristic algorithm outputs a suitable triangulation for all 2977 non-trivial prime knots with ≤ 12 crossings, showing it to be extremely effective in practice.

We finish this section with some useful properties of suitable triangulations.

Lemma 3.4. *Let $\mathcal{T}$ be an efficient suitable triangulation of a knot complement. Then every closed normal surface embedded in $\mathcal{T}$ has Euler characteristic ≤ 0.*

Proof. The efficiency criterion ensures that there are no embedded normal 2-spheres. The only other closed surface of positive Euler characteristic is the projective plane, which does not embed in $\mathbb{R}^3$ and so cannot embed in any knot complement. □

Lemma 3.5. *Let K be any knot, let $\mathcal{T}$ be any suitable triangulation of its complement with meridional boundary edge m, and let S be any normal surface in $\mathcal{T}$. Then S is a spanning surface for K if and only if S has no closed components and S meets edge m in precisely one point.*

Proof. Let $\partial\mathcal{T}$ denote the two-triangle torus on the boundary of $\mathcal{T}$, and let ∂S denote the collection of boundary curves of S. Each curve of ∂S is a *normal curve* on $\partial\mathcal{T}$; that is, a union of arcs between distinct edges of $\partial\mathcal{T}$ as illustrated in Figure 5.

If S is spanning then S is connected and has no closed components; moreover, ∂S consists of a single normal curve c whose algebraic intersection number with m is ± 1. It is a property of normal curves on a two-triangle torus that every such curve meets m in precisely one point [23].

Conversely, suppose that S has no closed components and meets m in precisely one point. Then S has some boundary curve c whose algebraic intersection number with m is ± 1. Because

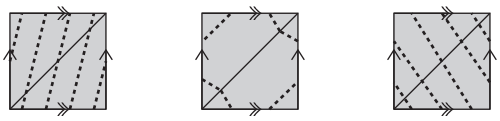


Figure 5: Examples of normal curves on a two-triangle torus

the boundary curves of S are disjoint and $\partial\mathcal{T}$ is a torus, any other boundary curve c' of S must be parallel to c or trivial in $\partial\mathcal{T}$, both of which would generate additional intersections with m (note that the only trivial *normal* curve on a two-triangle torus cuts each edge twice). Therefore c is the only boundary curve of S.

Since S has no closed components it follows that S is a connected surface with boundary c, and since c has algebraic intersection number ± 1 with m it follows that S is spanning. □

4 A Hilbert basis algorithm

Our first algorithm for computing the crosscap number $C(K)$ follows a common pattern for topological algorithms: it is based on the enumeration of fundamental normal surfaces.

It is worth revisiting the algorithm for the orientable counterpart of $C(K)$, the knot genus $g(K)$, as described by Hass, Lagarias and Pippenger [15]. Their algorithm is based on the following result:

Theorem 4.1 (Hass, Lagarias and Pippenger, 1999)**.** *Let K be any knot, and $\mathcal{T}$ be any triangulation of its complement. Then there is a fundamental normal orientable spanning surface of genus $g(K)$.*

The algorithm for computing $g(K)$ is then to enumerate all fundamental normal surfaces in $\mathcal{T}$, and to observe the smallest genus orientable spanning surface that appears.

For computing crosscap number, things are less straightforward: it is not even known whether there must be a *normal* non-orientable spanning surface of non-orientable genus $C(K)$, let alone a fundamental normal surface. The arguments of Hass, Lagarias and Pippenger use the fact that any minimal genus orientable spanning surface is essential[4]; however, these arguments do not translate to the non-orientable case since there are knots for which *every* minimal genus non-orientable spanning surface is non-essential [2, 19].

Our solution is twofold: we work with efficient suitable triangulations, which allow us to obtain precise results in many cases, and for those cases that remain we use Clark's inequality to reduce the solution to one of two possible values.

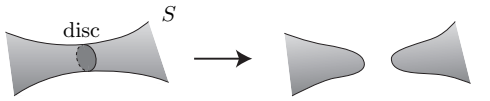
Lemma 4.2. *Let K be any non-trivial knot, and $\mathcal{T}$ be any suitable triangulation of its complement. Then either there is a normal non-orientable spanning surface of non-orientable genus $C(K)$, or else $C(K) = 2g(K) + 1$.*

Proof. Let S be any non-orientable spanning surface of non-orientable genus $C(K)$, and let m be the meridional boundary edge of $\mathcal{T}$. Since the boundary of S has algebraic intersection number ± 1 with m, we can isotope S so that the boundary of S meets m in precisely one point (a simple operation on the two-triangle torus boundary of $\mathcal{T}$).

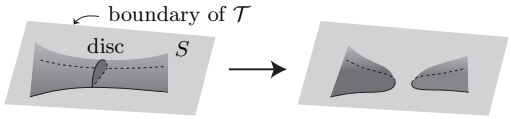
[4]That is, both incompressible and boundary incompressible. Further details are not required here.

We now follow the standard *normalisation procedure* that converts an arbitrary properly embedded surface into an embedded normal surface (possibly with different topology). We do not reiterate the details of this procedure here; for full details the reader is referred to a standard reference such as [20] or [25]. Instead we highlight some of its key aspects:

- Most steps in the procedure are isotopies, which preserve the fact that S is a non-orientable spanning surface of genus $C(K)$. However, three types of step can alter the topology of S:
  - *internal compressions*, which involve surgery on a disc that is bounded by a curve on S, as illustrated in Figure 6(a);
  - *boundary compressions*, which involve surgery on a disc that is bounded by an arc on S and an arc on the boundary of $\mathcal{T}$, as illustrated in Figure 6(b);
  - *deletion of trivial components*, where we remove components of S that are trivial spheres (bounding a ball in $\overline{K}$) or trivial discs (parallel into the boundary of $\overline{K}$).



(a) An internal compression



(b) A boundary compression

Figure 6: Examples of normalisation moves that can alter the topology of S

- For any edge e of $\mathcal{T}$, no step in the procedure will ever increase the number of intersections between the surface and e. Moreover, each internal compression and each boundary compression can be performed in a manner that never reduces this number of intersections either.

We now analyse how internal compressions and boundary compressions can affect our non-orientable spanning surface S. At each stage, we assume that S is a non-orientable spanning surface of minimum genus $C(K)$.

- For internal compressions, there are two cases to consider:
  - If the boundary of the compression disc is a separating curve in S, then the compression splits S into two disjoint pieces S_1 and S_2, where S is the connected sum $S_1 \# S_2$. Without loss of generality, we assume that the boundary curve of S stays with S_1, and so S_1 is also a spanning surface and S_2 is a closed surface.

    Since $S = S_1 \# S_2$, at least one of S_1 and S_2 must be non-orientable, and since no closed non-orientable surface can embed in a knot complement, this non-orientable piece must be S_1. Moreover, a simple Euler characteristic calculation gives $\chi(S) = \chi(S_1) + \chi(S_2) - 2$, and since $\chi(S_2) \leq 2$ it follows that S_1 (like S) must have the smallest possible non-orientable genus $C(K)$. Therefore we can simply delete S_2,

replace S with the non-orientable spanning surface S_1 of genus $C(K)$, and continue the process.

- If the boundary of the compression disc is not separating in S, then the resulting surface is some spanning surface U for which $\chi(U) = \chi(S) + 2$. U cannot be non-orientable, since its non-orientable genus would be less than the minimum $C(K)$. Therefore U is an orientable spanning surface with orientable genus $\frac{1}{2}(1 - \chi(U)) = \frac{1}{2}(1 - \chi(S) - 2) = \frac{1}{2}(C(K) - 2)$. This gives $g(K) \leq \frac{1}{2}(C(K) - 2)$ in contradiction to Clark's inequality (Theorem 2.1), and so this case can never occur.

- For boundary compressions we first note that, since boundary compressions do not change the number of intersections with the meridional edge m, the resulting (possibly disconnected) surface still cuts m precisely once.

  - If the compression separates S into two disjoint surfaces S_1 and S_2, we may assume that S_1 cuts edge m precisely once and S_2 does not cut m at all. Therefore the boundary of S_2 must be either a meridian or a trivial curve on the boundary torus; either way, it can be filled with a disc in $S^3 \backslash \overline{K}$ to yield a closed surface in S^3. Since no closed non-orientable surface can embed in S^3, it follows that S_2 must be orientable. Therefore the piece S_1 is non-orientable. This time the Euler characteristic argument gives $\chi(S) = \chi(S_1) + \chi(S_2) - 1$ with $\chi(S_2) \leq 1$. Therefore the piece S_1 is again a non-orientable spanning surface with genus $C(K)$, and we simply delete S_2, replace S with S_1 and continue.
  - If the compression does not separate S, the result is a spanning surface U with $\chi(U) = \chi(S) + 1$. In this case the normalisation process might fail. As before, U cannot have non-orientable genus less than $C(K)$; therefore U is an *orientable* spanning surface with orientable genus $\frac{1}{2}(1 - \chi(U)) = \frac{1}{2}(C(K) - 1)$. This gives $g(K) \leq \frac{1}{2}(C(K) - 1)$, and combined with Clark's inequality we obtain $C(K) = 2g(K) + 1$ precisely.

If we follow the arguments above, deletion of trivial components is never required: we delete unwanted extra components as they appear during internal and boundary compressions, and the spanning surface itself is never a trivial sphere nor a trivial disc.

In conclusion, either the normalisation procedure yields an embedded *normal* non-orientable spanning surface of genus $C(K)$, or else we have a situation in which $C(K) = 2g(K) + 1$. □

We can now move from normal surfaces to *fundamental* normal surfaces, giving us the main theorem of this section.

Theorem 4.3. *Let K be any non-trivial knot, and $\mathcal{T}$ be any efficient suitable triangulation of its complement. Then either there is a* fundamental *normal non-orientable spanning surface of non-orientable genus $C(K)$, or else $C(K) \in \{2g(K),\ 2g(K) + 1\}$.*

Proof. If there is no *normal* non-orientable spanning surface of genus $C(K)$ then $C(K) = 2g(K) + 1$, by Lemma 4.2. Assume then that there is some normal non-orientable spanning surface of genus $C(K)$, and let S be such a surface containing the fewest possible normal discs. Let m denote the meridional boundary edge of $\mathcal{T}$.

If S is not fundamental then $S = U + V$ for some non-empty normal surfaces U and V. By Lemma 3.5, S cuts the edge m in precisely one point. Since edge intersections are additive under normal sum[5], we may assume without loss of generality that U cuts m precisely once

[5]This refers to absolute numbers of intersections, not algebraic intersection number.

and V does not cut m at all. We may also assume that U is connected (since any components that do not meet m can be moved over to V), whereby Lemma 3.5 shows that U is a spanning surface also.

The surface V is a union of closed components and/or bounded components. Every boundary curve of V is disjoint from m and is therefore a meridian (since any other *normal* boundary curve, even the trivial curve, must cut m at least once). This means that no bounded component of V can be a disc (because the meridian has non-trivial homology in $\overline{K}$, and so no disc in $\overline{K}$ can have a meridian as boundary). Therefore every bounded component of V has non-positive Euler characteristic; combined with Lemma 3.4 this gives us $\chi(V) \leq 0$. Since Euler characteristic is additive under normal sum we have $\chi(S) = \chi(U) + \chi(V)$, and therefore $\chi(U) \geq \chi(S)$.

If U is non-orientable, it follows that U is a normal non-orientable spanning surface of genus at most $C(K)$. By minimality of $C(K)$ this genus is precisely $C(K)$; moreover, since $S = U + V$ we see that U contains fewer normal discs than S. This contradicts our initial choice of S.

Therefore U is orientable, and has genus $\leq \frac{1}{2}(1 - \chi(S)) = \frac{1}{2}C(K)$. Since U is spanning this gives $2g(K) \leq C(K)$, and by Clark's inequality it follows that $C(K) \in \{2g(K),\ 2g(K)+1\}$. □

Combining Theorems 4.1 and 4.3 gives our first algorithm for computing crosscap number.

Algorithm 4.4. *Given a knot diagram describing the knot $K \subseteq S^3$, the following procedure will either (i) output the crosscap number $C(K)$, or (ii) output a pair of integers one of which is $C(K)$.*

1. *Construct an efficient suitable triangulation $\mathcal{T}$ of the complement $\overline{K}$ using Algorithm 3.1.*
2. *Enumerate the set $\mathcal{F}$ of all fundamental normal surfaces in $\mathcal{T}$.*
3. *For each surface $S \in \mathcal{F}$, test whether S is a spanning surface for K and whether S is orientable or non-orientable. Let g_n be the minimum non-orientable genus amongst all non-orientable spanning surfaces in $\mathcal{F}$, or ∞ if no non-orientable spanning surfaces are found. Let g_o be the minimum orientable genus amongst all orientable spanning surfaces in $\mathcal{F}$.*
4. *If $g_o = 0$ then output 0. If $g_n \leq 2g_o$ then output g_n. Otherwise output $\{2g_o,\ 2g_o + 1\}$.*

As outlined in [15], the enumeration in step 2 involves a high-dimensional Hilbert basis computation, and can be performed using standard software such as *Normaliz* [3]. In step 3, we can test for orientability and genus by reconstructing the normal surface, and we can test whether S is a spanning surface for K using Lemma 3.5.

Theorem 4.5. *Algorithm 4.4 is correct, i.e., the crosscap number $C(K)$ is one of the solutions that it outputs in step 4.*

Proof. By Theorem 4.1, $g_o = g(K)$. If K is the unknot then $g_o = 0$ and the algorithm outputs the correct value $C(K) = 0$. Assume then that K is non-trivial, which means that $g_o > 0$.

If some non-orientable spanning surface of genus $C(K)$ does appear in the set $\mathcal{F}$ then we also have $g_n = C(K)$. If $g_n \leq 2g_o$ then we output the correct value $g_n = C(K)$. Otherwise Clark's inequality (Theorem 2.1) gives $C(K) = 2g_o + 1$, and so the correct value of $C(K)$ appears in the output set $\{2g_o,\ 2g_o + 1\}$.

If no non-orientable spanning surface of genus $C(K)$ appears in $\mathcal{F}$ then $g_n > C(K)$, and by Theorem 4.3 we also have $C(K) \in \{2g(K),\ 2g(K)+1\} = \{2g_o,\ 2g_o+1\}$. In this case $2g_o < g_n$, and again our output set $\{2g_o,\ 2g_o + 1\}$ contains the correct value of $C(K)$. □

5 An exact integer programming algorithm

The bottleneck in Algorithm 4.4 is the Hilbert basis computation (the enumeration of all fundamental normal surfaces), which remains intractable for all but the simplest knots. For this reason we seek alternate algorithms that do not require an enumeration of surfaces. Instead we attempt to locate a minimal genus non-orientable spanning surface directly using integer programming.

We begin this section with several results that allow us to formulate topological constraints numerically. We finish with Algorithm 5.4, which gives the full procedure for computing $C(K)$.

The first lemma is due to Hass, Lagarias and Pippenger [15], and allows us to place upper bounds on the coordinates of fundamental normal surfaces.

Lemma 5.1 (Hass, Lagarias and Pippenger, 1999)**.** *Let $\mathcal{T}$ be a 3-manifold triangulation with n tetrahedra, and let S be a fundamental normal surface in $\mathcal{T}$. Then every coordinate of the vector representation $\mathbf{v}(S)$ is at most $n \cdot 2^{7n+2}$.*

This bound plays an important role in our algorithm, and it is desirable to improve it. Techniques based on linear programming offer potential; for instance, in [8] the authors significantly reduce this bound for related problems.[6] We do not pursue these matters further here.

Our next result is the well-known observation that Euler characteristic is linear on the normal surface solution cone. Our proof includes details on how such a linear function can be constructed.

Lemma 5.2. *Let $\mathcal{T}$ be an n-tetrahedron 3-manifold triangulation. Then there is a linear function $\chi\colon \mathbb{R}^{7n} \to \mathbb{R}$ such that, for any normal surface S in $\mathcal{T}$, $\chi(\mathbf{v}(S))$ is the Euler characteristic of S.*

Proof. Let $\mathbf{x} \in \mathbb{R}^{7n}$. We define $\chi(\mathbf{x}) = V - E - F$, where:

- F is the sum of all $4n$ triangular coordinates and all $3n$ quadrilateral coordinates of $\mathbf{x}$.
- E is defined as a sum over edges of normal discs: if e is one of the three edges of a normal triangle or one of the four edges of a normal quadrilateral, then e contributes $+1$ to this sum if e lies on the boundary of $\mathcal{T}$, or $+1/2$ if e is internal to $\mathcal{T}$.
- V is defined as a sum over vertices of normal discs: if v is one of the three vertices of a normal triangle or one of the four vertices of a normal quadrilateral, then v contributes $+1/d$ to this sum, where d is the degree of the *edge* of $\mathcal{T}$ that v lies within.

It is clear that V, E and F can all be expressed as sums of normal coordinates, and so χ is linear on $\mathbb{R}^{7n}$. Moreover: each vertex of degree d in a normal surface S contributes a total of $+1/d \times d = +1$ to V (since it meets d normal discs, and it must lie within an edge of $\mathcal{T}$ of degree d); each boundary edge of S contributes $+1$ to E; each internal edge of S contributes $+1/2 \times 2 = +1$ to E (since it meets two normal discs); and each triangular or quadrilateral face of S contributes $+1$ to F. Therefore $\chi(\mathbf{v}(S)) = V - E + F$ is indeed the Euler characteristic of S. □

It should be noted that there are many linear functions $\chi\colon \mathbb{R}^{7n} \to \mathbb{R}$ with this property; the formulation above was chosen for its simple proof. There are sparser formulations that may be preferable for computation; again we do not pursue this matter here.

[6] The paper [8] considers *vertex* (not fundamental) normal surfaces, and works in a different coordinate system.

Our final lemma allows us to arithmetically express the condition that a given normal surface is a spanning surface for our knot.

Lemma 5.3. *Let K be any knot, and let $\mathcal{T}$ be any suitable triangulation of its complement with meridional boundary edge m. Let Δ_i be any tetrahedron of $\mathcal{T}$ that contains the boundary edge m, and let $t_{i,a}$, $t_{i,b}$, $q_{i,c}$ and $q_{i,d}$ be the coordinates describing the four normal disc types in Δ_i that touch m, as illustrated in Figure 7.*

For any normal surface S in $\mathcal{T}$, S is a spanning surface for K if and only if S has no closed components and the vector representation of S satisfies $t_{i,a} + t_{i,b} + q_{i,c} + q_{i,d} = 1$. We call this equation the spanning equation *for $\mathcal{T}$.*

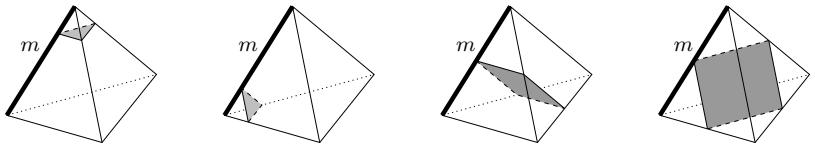


Figure 7: The four normal disc types in Δ_i that touch the meridional boundary edge m

Proof. The sum $t_{i,a} + t_{i,b} + q_{i,c} + q_{i,d}$ counts the number of times that the surface S cuts the meridional boundary edge m (this is true regardless of which tetrahedron Δ_i we chose). The result then follows immediately from Lemma 3.5. □

Algorithm 5.4. *Given a knot diagram describing the knot $K \subseteq S^3$, the following procedure will either (i) output the crosscap number $C(K)$, or (ii) output a pair of integers one of which is $C(K)$.*

1. *Construct an efficient suitable triangulation $\mathcal{T}$ of the complement $\overline{K}$ using Algorithm 3.1. Let n be the number of tetrahedra in $\mathcal{T}$.*
2. *Optimise the following integer program using an exact arithmetic integer programming solver:*
   - *(i) define the $7n$ non-negative integer variables $t_{i,j}$ for $1 \leq i \leq n$, $1 \leq j \leq 4$ and $q_{i,j}$ for $1 \leq i \leq n$, $1 \leq j \leq 3$;*
   - *(ii) define the $3n$ binary variables $b_{i,j}$ for $1 \leq i \leq n$, $1 \leq j \leq 3$;*
   - *(iii) add constraints for the matching equations and the spanning equation (Lemma 5.3);*
   - *(iv) add constraints $n \cdot 2^{7n+2} \cdot b_{i,j} \geq q_{i,j}$ for $1 \leq i \leq n$, $1 \leq j \leq 3$;*
   - *(v) add constraints $b_{i,1} + b_{i,2} + b_{i,3} \leq 1$ for $1 \leq i \leq n$;*
   - *(vi) maximise the Euler characteristic function (Lemma 5.2).*
3. *Let $\mathbf{x}$ be an optimal solution, and let S be the corresponding normal surface with any closed components removed. If S is non-orientable then output $1 - \chi(S)$. Otherwise output the pair $\{1 - \chi(S), 2 - \chi(S)\}$.*

It is crucial that the integer programming solver be based on exact integer arithmetic; see [1, 12] for examples of such tools. With traditional floating-point solvers, round-off errors can creep in, especially when working with coefficients as large as $n \cdot 2^{7n+2}$. Such round-off errors can result in a sub-optimal solution, or even a solution $\mathbf{x}$ that does not represent a spanning surface at all.

Theorem 5.5. *Algorithm 5.4 is well-defined (i.e., the integer program in step 2 is not unbounded), and is correct (i.e., the crosscap number $C(K)$ is one of the solutions output in step 3).*

Proof. The integer variables $t_{i,j}$ and $q_{i,j}$ correspond to the usual triangle and quadrilateral coordinates. Each binary variable $b_{i,j}$ has the following effect under condition (iv): if $b_{i,j} = 0$ then it forces $q_{i,j} = 0$, and if $b_{i,j} = 1$ then we can have any $q_{i,j}$ in the range $0 \leq q_{i,j} \leq n \cdot 2^{7n+2}$.

First consider any feasible solution to the integer program (optimal or otherwise). Because any such solution satisfies all constraints from step 2, we know that the variables $t_{i,j}$ and $q_{i,j}$ are non-negative integers, satisfy the matching equations and the spanning equation (condition (iii)), and satisfy the quadrilateral constraints (condition (v)). By Theorem 2.2 and Lemma 5.3 it follows that any feasible solution to our integer program describes a normal surface in $\mathcal{T}$, which is the disjoint union of a spanning surface and zero or more additional closed components.

If the integer program is unbounded then such a union can have arbitrarily large Euler characteristic. Because any spanning surface has Euler characteristic ≤ 1, it follows that $\mathcal{T}$ contains some closed normal surface of positive Euler characteristic. However, this is impossible by Lemma 3.4, and so the integer program is bounded and Algorithm 5.4 is well-defined.

We now turn to the proof of correctness. By the argument above, the surface S obtained in step 3 (with closed components removed) must be a spanning surface. If S is non-orientable then its non-orientable genus is $1 - \chi(S)$, and we have $C(K) \leq 1 - \chi(S)$. If S is orientable then its orientable genus is $\frac{1}{2}(1 - \chi(S)) \geq g(K)$, and by Clark's inequality we have $C(K) \leq 2g(K) + 1 \leq 2 - \chi(S)$.

Having bounded $C(K)$ from above, we now bound it from below. Let S_o be the smallest-genus orientable normal surface whose normal coordinates are all at most $n \cdot 2^{7n+2}$, and let S_n be the smallest-genus non-orientable normal surface whose normal coordinates are at most $n \cdot 2^{7n+2}$. By Theorem 4.1 and Lemma 5.1, the surface S_o exists and has genus $g(K)$. By Theorem 4.3 and Lemma 5.1, we have one of two cases for S_n: either (a) S_n exists and has non-orientable genus $C(K)$, or (b) $C(K) \geq 2g(K)$ (in which case S_n might or might not exist).

Since S_o and S_n are normal spanning surfaces with coordinates $\leq n \cdot 2^{7n+2}$, they satisfy all of the constraints in step 2 of the algorithm (here each $b_{i,j} = 0$ or 1 according to whether the corresponding $q_{i,j}$ is zero or non-zero). If S' is the normal surface corresponding to the optimal solution $\mathbf{x}$, it follows that $\chi(S_o) \leq \chi(S')$, and that $\chi(S_n) \leq \chi(S')$ if S_n exists. As before, $\mathcal{T}$ contains no closed normal surfaces of positive Euler characteristic, and so after removing any closed components of S' we obtain $\chi(S_o) \leq \chi(S)$, and $\chi(S_n) \leq \chi(S)$ if S_n exists.

We can now piece our various results together. In case (a) above where S_n exists with non-orientable genus $C(K)$, we have $C(K) = 1 - \chi(S_n) \geq 1 - \chi(S)$. In case (b) we have $C(K) \geq 2g(K) = 1 - \chi(S_o) \geq 1 - \chi(S)$. Either way we obtain the lower bound $1 - \chi(S) \leq C(K)$.

In conclusion: if S is orientable then $1 - \chi(S) \leq C(K) \leq 2 - \chi(S)$, and if S is non-orientable then $1 - \chi(S) \leq C(K) \leq 1 - \chi(S)$. Therefore the output of our algorithm is correct. □

6 A limited precision integer programming algorithm

The exact integer programming algorithm in the previous section avoids an expensive Hilbert basis enumeration, but it has its own drawbacks. Exact integer programming solvers are rarer and less well-developed than their limited precision floating-point cousins, and the exponentially large constraint coefficients $n \cdot 2^{7n+2}$ can have a crippling effect on performance. Moreover, constructing an *efficient* suitable triangulation remains slow, as discussed in Section 3.

Our final algorithm avoids these performance problems: we allow floating-point solvers with limited precision, and we replace each large coefficient $n \cdot 2^{7n+2}$ in our integer program with the arbitrarily chosen small coefficient 10000. We also drop the efficiency requirement and allow just suitable triangulations, which are faster to construct.

These concessions bring about a loss of information and precision from our solution. In response, we add tests to ensure that the solution to our integer program is valid (i.e., represents a spanning surface), and we interpret the final output value as just an upper bound on $C(K)$ (since we cannot be sure that a better solution was inadvertently missed). The complete algorithm is as follows.

Algorithm 6.1. *Given a knot diagram describing the knot $K \subseteq S^3$, the following procedure will output an upper bound U for which $C(K) \leq U$.*

1. *Construct a suitable triangulation $\mathcal{T}$ of the complement $\overline{K}$ using the fast but heuristic-based Algorithm 3.3. If this algorithm produces no triangulation then output ∞ and terminate immediately. Otherwise let n be the number of tetrahedra in $\mathcal{T}$.*

2. *Optimise the following integer program using a fast solver based on floating-point arithmetic:*

   (i) *define the $7n$ non-negative integer variables $t_{i,j}$ for $1 \leq i \leq n$, $1 \leq j \leq 4$ and $q_{i,j}$ for $1 \leq i \leq n$, $1 \leq j \leq 3$;*

   (ii) *define the $3n$ binary variables $b_{i,j}$ for $1 \leq i \leq n$, $1 \leq j \leq 3$;*

   (iii) *add constraints for the matching equations and the spanning equation (Lemma 5.3);*

   (iv) *add constraints $10000 \cdot b_{i,j} \geq q_{i,j}$ for $1 \leq i \leq n$, $1 \leq j \leq 3$;*

   (v) *add constraints $b_{i,1} + b_{i,2} + b_{i,3} \leq 1$ for $1 \leq i \leq n$;*

   (vi) *maximise the Euler characteristic function (Lemma 5.2).*

3. *If this integer program is unbounded then output ∞ and terminate.*

4. *Let $\mathbf{x}$ be an optimal solution. Using exact integer arithmetic, test whether $\mathbf{x}$ satisfies the constraints laid out in step 2. If not then output ∞ and terminate.*

5. *Let S be the normal surface described by $\mathbf{x}$ with any closed components removed. Output $1 - \chi(S)$ if S is non-orientable, or $2 - \chi(S)$ if S is orientable.*

Theorem 6.2. *Algorithm 6.1 is correct, i.e., $C(K)$ is less than or equal to the output value.*

Proof. This is a much simpler variant of the proof for Algorithm 5.4. If we output ∞ then the algorithm is clearly correct. Otherwise we have a solution $\mathbf{x}$ that satisfies the constraints of step 2 (as shown by the test in step 4), but with no guarantee that this solution is optimal.

As in Algorithm 5.4, because $\mathbf{x}$ satisfies the constraints from step 2, the surface S (with closed components removed) must be a spanning surface for K. If S is non-orientable then its non-orientable genus is $1-\chi(S)$, and so $C(K) \leq 1-\chi(S)$. If S is orientable then its orientable genus is $\frac{1}{2}(1-\chi(S)) \geq g(K)$, and by Clark's inequality we have $C(K) \leq 2g(k)+1 \leq 2-\chi(S)$. □

7 Discussion and computational results

As discussed in the introduction, Algorithms 4.4 and 5.4 remain too slow for practical use for all but the simplest knots. Algorithm 4.4 requires the enumeration of a Hilbert basis for a high-dimensional polyhedral cone, an extremely expensive procedure. Algorithm 5.4 requires exact arithmetic integer programming in problems with extremely large coefficients (such as $n \cdot 2^{7n+2}$), which remains intractable with currently available tools.

Despite this, both algorithms have certain benefits. Unlike our final algorithm, they produce no more than two possible solutions; moreover, further analysis shows that if $C(K) < 2g(K)$ then both Algorithms 4.4 and 5.4 guarantee a unique solution. They also rely on substantially different underlying computational problems (Hilbert basis enumeration versus exact integer programming), both of which have supporting software that continues to enjoy significant advances in efficiency [1, 3, 12]; in this sense we are able to "hedge our bets". A further avenue for improving the efficiency of Algorithm 5.4 is to lower the coordinate bounds in Lemma 5.1.

An important observation is that none of the algorithms in this paper are able to give a unique solution in cases where Clark's inequality is tight, i.e., $C(K) = 2g(K) + 1$.

In contrast to the first two algorithms, the limited-precision Algorithm 6.1 is fast and effective. By combining the upper bounds from this algorithm with lower bounds from existing knot tables, we are able to make significant improvements to these tables.

Specifically: the *KnotInfo* project contains a rich body of invariant data for all 2977 prime non-trivial knots with ≤ 12 crossings. Of these knots, 289 have known crosscap numbers, and the remaining 2688 knots are listed with best-known upper and lower bounds.

For each of these knots, we begin with the corresponding triangulation of the complement from the *SnapPy* census [13], use *Regina* [4, 7] to convert it into a triangulation with boundary faces, and then run Algorithm 6.1 to obtain a bound on the crosscap number. For the crucial optimisation step we use IBM's *ILOG CPLEX* package (version 12.2). The triangulations range from 5 to 50 tetrahedra in size (with an average of 33.27), and the total running time over all triangulations is roughly 4.5 hours on a quad-core 2.93 GHz Intel Core i7 CPU.

Our first observation is that Algorithm 6.1 never outputs ∞. That is: for *every* knot in the tables, the heuristic Algorithm 3.3 produces a suitable triangulation, the integer program is not unbounded (giving strong evidence that these triangulations may in fact be *efficient* suitable triangulations), and the optimal solution to the integer program satisfies all of the necessary constraints to produce a valid spanning surface.

For 27 of the 2688 knots with unknown crossing number, the upper bound produced by Algorithm 6.1 is equal to the lower bound listed in the *KnotInfo* tables; as a result we can identify the crosscap number precisely. These 27 knots and their crosscap numbers are listed in Table 1. For another 747 knots, our upper bound improves upon the *KnotInfo* upper bound: on average we reduce the number of possible solutions from 4.18 to 2.86. Detailed results from all of these computations, including a *Regina* data file listing the final spanning surfaces, can be found at `http://www.maths.uq.edu.au/~bab/code/`.

To conclude, we note that the integer programming framework given in Algorithms 5.4 and

KnotInfo name	Dowker-Thistlethwaite name	Genus	Previous bounds on $C(K)$	New value of $C(K)$
8_{20}	$8n_1$	2	$[2, 4]$	2
10_{125}	$10n_{15}$	3	$[2, 4]$	2
10_{126}	$10n_{17}$	3	$[2, 4]$	2
10_{139}	$10n_{27}$	4	$[2, 3]$	2
10_{140}	$10n_{29}$	2	$[2, 4]$	2
10_{142}	$10n_{30}$	3	$[2, 4]$	2
10_{145}	$10n_{14}$	2	$[2, 4]$	2
10_{161}	$10n_{31}$	3	$[2, 5]$	2
$11n_{102}$	$11n_{102}$	2	$[2, 4]$	2
$11n_{104}$	$11n_{104}$	4	$[2, 4]$	2
$11n_{135}$	$11n_{135}$	3	$[2, 5]$	2
$12n_{0121}$	$12n_{0121}$	2	$[2, 4]$	2
$12n_{0233}$	$12n_{0233}$	4	$[2, 4]$	2
$12n_{0235}$	$12n_{0235}$	4	$[2, 4]$	2
$12n_{0242}$	$12n_{0242}$	5	$[2, 3]$	2
$12n_{0404}$	$12n_{0404}$	2	$[2, 4]$	2
$12n_{0474}$	$12n_{0474}$	4	$[2, 4]$	2
$12n_{0475}$	$12n_{0475}$	3	$[2, 4]$	2
$12n_{0522}$	$12n_{0522}$	3	$[2, 4]$	2
$12n_{0575}$	$12n_{0575}$	4	$[2, 4]$	2
$12n_{0581}$	$12n_{0581}$	3	$[2, 4]$	2
$12n_{0582}$	$12n_{0582}$	2	$[2, 4]$	2
$12n_{0591}$	$12n_{0591}$	4	$[2, 5]$	2
$12n_{0721}$	$12n_{0721}$	4	$[2, 4]$	2
$12n_{0725}$	$12n_{0725}$	5	$[2, 3]$	2
$12n_{0749}$	$12n_{0749}$	3	$[2, 5]$	2
$12n_{0851}$	$12n_{0851}$	3	$[2, 5]$	2

Table 1: The 27 knots with newly-computed crosscap numbers

6.1 extends beyond the specific problem of computing crosscap numbers—it can be used as a general framework for locating normal surfaces with various properties. For instance, by optimising Euler characteristic this integer programming framework can be used to test for 0-efficiency [21], compute connected sum decompositions [21], and perform 3-sphere recognition [5]. Given current advances in exact integer programming solvers, this potential for applying optimisation techniques to problems in low-dimensional topology is only beginning to be explored.

Acknowledgements

The authors are grateful to the Queensland Cyber Infrastructure Foundation and RMIT University for the use of their high-performance computing facilities. The first author is supported by the Australian Research Council under the Discovery Projects funding scheme (projects DP1094516 and DP110101104).

References

[1] David L. Applegate, William Cook, Sanjeeb Dash, and Daniel G. Espinoza, *Exact solutions to linear programming problems*, Oper. Res. Lett. **35** (2007), no. 6, 693–699.

[2] Katsuji Bessho, *Incompressible surfaces bounded by links*, Master's thesis, Osaka University, 1994.

[3] Winfried Bruns and Bogdan Ichim, *Normaliz: Algorithms for affine monoids and rational cones*, J. Algebra **324** (2010), no. 5, 1098–1113.

[4] Benjamin A. Burton, *Introducing Regina, the 3-manifold topology software*, Experiment. Math. **13** (2004), no. 3, 267–272.

[5] ———, *Quadrilateral-octagon coordinates for almost normal surfaces*, Experiment. Math. **19** (2010), no. 3, 285–315.

[6] ———, *The Pachner graph and the simplification of 3-sphere triangulations*, SCG '11: Proceedings of the Twenty-Seventh Annual Symposium on Computational Geometry, ACM, 2011, pp. 153–162.

[7] Benjamin A. Burton, Ryan Budney, William Pettersson, et al., *Regina: Software for 3-manifold topology and normal surface theory*, `http://regina.sourceforge.net/`, 1999–2011.

[8] Benjamin A. Burton and Melih Ozlen, *A tree traversal algorithm for decision problems in knot theory and 3-manifold topology*, Preprint, `arXiv:1010.6200`, October 2010.

[9] Benjamin A. Burton, J. Hyam Rubinstein, and Stephan Tillmann, *The Weber-Seifert dodecahedral space is non-Haken*, Trans. Amer. Math. Soc. **364** (2012), no. 2, 911–932.

[10] Jae Choon Cha and Charles Livingston, *KnotInfo: Table of knot invariants*, `http://www.indiana.edu/~knotinfo`, accessed June 2011.

[11] Bradd Evans Clark, *Crosscaps and knots*, Internat. J. Math. Math. Sci. **1** (1978), no. 1, 113–123.

[12] William Cook, Thorsten Koch, Daniel E. Steffy, and Kati Wolter, *An exact rational mixed-integer programming solver*, Integer Progamming and Combinatorial Optimization, Lecture Notes in Comput. Sci., vol. 6655, Springer, Berlin, 2011, pp. 104–116.

[13] Marc Culler, Nathan M. Dunfield, and Jeffrey R. Weeks, *SnapPy, a computer program for studying the geometry and topology of 3-manifolds*, `http://snappy.computop.org/`, 1991–2011.

[14] Wolfgang Haken, *Theorie der Normalflächen*, Acta Math. **105** (1961), 245–375.

[15] Joel Hass, Jeffrey C. Lagarias, and Nicholas Pippenger, *The computational complexity of knot and link problems*, J. Assoc. Comput. Mach. **46** (1999), no. 2, 185–211.

[16] Geoffrey Hemion, *The classification of knots and 3-dimensional spaces*, Oxford Science Publications, Oxford University Press, Oxford, 1992.

[17] Mikami Hirasawa and Masakazu Teragaito, *Crosscap numbers of 2-bridge knots*, Topology **45** (2006), no. 3, 513–530.

[18] Kazuhiro Ichihara and Shigeru Mizushima, *Crosscap numbers of pretzel knots*, Topology Appl. **157** (2010), no. 1, 193–201.

[19] Kazuhiro Ichihara, Masahiro Ohtouge, and Masakazu Teragaito, *Boundary slopes of non-orientable Seifert surfaces for knots*, Topology Appl. **122** (2002), no. 3, 467–478.

[20] William Jaco and J. Hyam Rubinstein, *PL equivariant surgery and invariant decompositions of 3-manifolds*, Adv. Math. **73** (1989), no. 2, 149–191.

[21] ———, *0-efficient triangulations of 3-manifolds*, J. Differential Geom. **65** (2003), no. 1, 61–168.

[22] ———, *Layered-triangulations of 3-manifolds*, Preprint, `arXiv:math/0603601`, March 2006.

[23] William Jaco and Eric Sedgwick, *Decision problems in the space of Dehn fillings*, Topology **42** (2003), no. 4, 845–906.

[24] William Jaco and Jeffrey L. Tollefson, *Algorithms for the complete decomposition of a closed* 3*-manifold*, Illinois J. Math. **39** (1995), no. 3, 358–406.

[25] Sergei Matveev, *Algorithmic topology and classification of 3-manifolds*, Algorithms and Computation in Mathematics, no. 9, Springer, Berlin, 2003.

[26] Hitoshi Murakami and Akira Yasuhara, *Crosscap number of a knot*, Pacific J. Math. **171** (1995), no. 1, 261–273.

[27] Masakazu Teragaito, *Crosscap numbers of torus knots*, Topology Appl. **138** (2004), no. 1-3, 219–238.

Benjamin A. Burton
School of Mathematics and Physics, The University of Queensland
Brisbane QLD 4072, Australia
(bab@maths.uq.edu.au)

Melih Ozlen
School of Mathematical and Geospatial Sciences, RMIT University
GPO Box 2476V, Melbourne VIC 3001, Australia
(melih.ozlen@rmit.edu.au)