THE UNKNOTTING NUMBER UNDER CONNECTED SUM AND MUTATION

SRIKHA CHAGANTI
ADVISED BY DR. MAGGIE MILLER

Abstract. There has been a long standing conjecture, implicitly stated in a 1937 paper by Wendt, that claims the unknotting number is additive under connected sum: For two knots K_1 and K_2 in $\mathbb{R}^3$, $u(K_1) + u(K_2) = u(K_1 \# K_2)$. This was recently disproven in Summer 2025 by Brittenham and Hermiller. The first example found was the connected sum of the $7_1((2,7)$-torus knot) and its mirror image, $\bar{7_1}$, which has an unknotting number of 5. The unknotting number of 7_1 is 3, and the unknotting number of the mirror image is 3. So, $u(K)+u(\bar{K}) = 6 > 5 \geq u(K\#\bar{K})$. The strategy to find the initial counterexample employed by Brittenham and Hermiller was motivated by the work they had done when attempting to disprove another conjecture that still remains open: Unknotting number is invariant under knot mutation. Knot mutation preserves various fundamental knot invariants, including the Alexander polynomial and the signature. Meanwhile, invariants that detect mutation are much more difficult to handle, making the open problem very difficult. In this thesis, we will explore the strategy employed to find the counterexample to the Unknotting Number Under Connected Sum conjecture, and the open problems that follow.

1. Introduction

When one generally thinks of a knot, they may usually picture a string tied around itself in a certain pattern, such as a slipknot or square knot. However, knots are mathematical objects of great interest. We can think of a mathematical knot as an embedding of the unit circle into $\mathbb{R}^3$. There are still various open problems regarding knots today. The problems we discuss in this thesis have to do with a measure on knots known as the unknotting number. The unknotting number describes how many crossing changes it takes to turn a knot back into the unknot. Although a very simple definition, it is deceptively difficult to calculate. The unknotting number of a knot was first defined by Wendt in a 1937 paper, where the following long standing conjecture (that has recently been answered) was implicitly stated. [14]

Conjecture 1.1. *For two knots K_1 and K_2 in R^3, $u(K_1) + u(K_2) = u(K_1 \# K_2)$.*

Upon initial glance, we can notice that one direction of the inequality is quite clear.

$$u(K_1 \# K_2) \leq u(K_1) + u(K_2)$$

If K_1 and K_2 can be unknotted in n and m crossing changes respectively, then we choose diagrams that can be unknotted with the minimum required crossing changes. Connect sum

Date: April 2026.

the diagrams, and the connect sum will turn into the unknot in $n+m$ changes. Less obvious, however, is the opposite direction; that $u(K_1 \# K_2) \geq u(K_1) + u(K_2)$. This would imply that the connect sum could not get 'rid of' any crossing changes.

A counterexample was found unexpectedly by Brittenham and Hermiller in 2025. [2] The authors of the paper find that the connect sum of the torus knot, $T(2,7)$, with its mirror image does not satisfy the conjecture, and also gives way to an infinite set of counterexamples. Surprisingly, they were attempting to answer a different conjecture.

Conjecture 1.2. *The unknotting number of a link is not invariant under mutation.*

This conjecture can be found in Kirby's problem list as Problem 1.69 (C), and was originally proposed by Boileau. [5]

Through the process of building a collection that would hopefully provide an example to prove Conjecture 1.2, they assume that Conjecture 1.1 is true. When they encounter a valid example, they realize that one of two things must be true. Either unknotting number is not invariant under mutation, or it is not additive under connect sum. Throughout this thesis, we will discuss the finding of the counterexample, and the still open mutation problem.

2. Definitions and Background

Definition 2.1. We define a mathematical **knot** to be an embedding of the unit circle into $\mathbb{R}^3$.

Definition 2.2. A **link** is the embedding of multiple circles into $\mathbb{R}^3$.

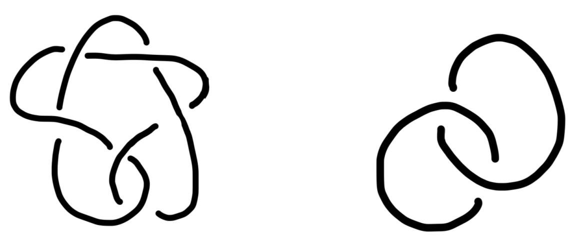


Figure 1. Knot 5_1 and the Hopf link

Definition 2.3. Let X and Y be topological spaces and f, g embeddings from X into Y. Then we call the continuous map $H : Y \times [0,1] \to Y$ an **ambient isotopy** if:

- $H(f,0) = f$
- $H(f,1) = g$
- Each $H(f,t)$ is homeomorphism from Y to itself, for all $t \in (0,1)$.

Knots K_1 and K_2 are said to be equivalent if there exists an ambient isotopy between the two.

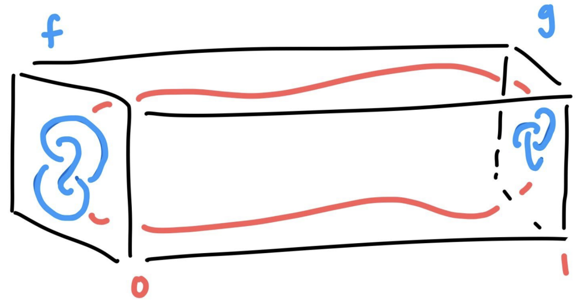


FIGURE 2. Isotopic Knots

We can notice that the definition of a knot requires an embedding of a circle, rather than an interval $[0,1]$ (like a 'string'). This is because otherwise, any two knots would be equivalent– they would all be isotopic to $[0,1]$.

The more visually intuitive way in which equivalent knot diagrams are related is through small elementary deformations known as Reidemeister moves. There are three possible Reidemeister moves, along with their inverses. Given two equivalent knots, there exists a sequence of Reidemeister moves deforming one diagram into the other. The three moves and their respective inverses are shown in Figure 3.

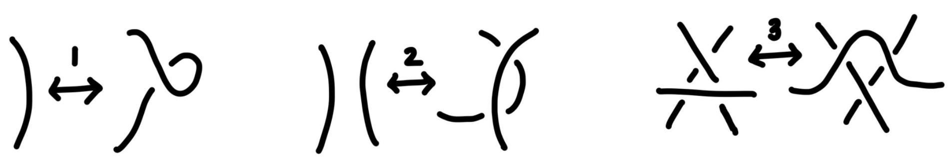


FIGURE 3. Reidemeister Moves

Definition 2.4. The **unknot** is the boundary of an embedded disk in $\mathbb{R}^3$.

We can think of the unknot as being a closed loop with no tangles in it.

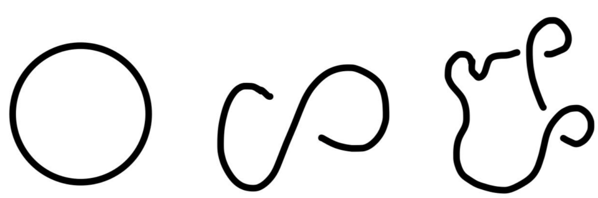


FIGURE 4. Various Examples of the Unknot

The crossings in a knot diagram can be labeled as positive or negative, shown in Figure 5.

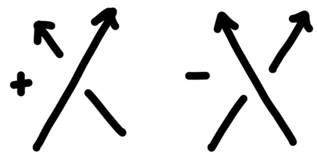


Figure 5. Positive and Negative Crossings

Definition 2.5. The **Dowker-Thistletwaite (DT) Code** of a knot is a sequence of even integers that represents a specific knot diagram.

It is used often in knot databases.

Definition 2.6. The **linking number** of two links, K and J, $lk(K, J)$ is the sum of the signs of all the crossings of K and J, divided by two.

A knot invariant assigns a value to a knot such that if two knots are equivalent, they must have the same invariant value. They are useful in distinguishing and classifying knots. Many of these knot invariants are most easily derived from a knot's Seifert surface.

Definition 2.7. For any arbitrarily oriented knot K, there exists an orientable, connected surface F in $\mathbb{R}^3$ such that K is the boundary of this surface. This surface is referred to as the **Seifert surface**.

Given a diagram for a knot K, we can decompose the diagram into simple closed curves. Each of these simple closed curves can be spanned by a disk. Then, attach positive or negative twist bands to the respective positive or negative crossing sites. This results in the orientable surface.

Using Seifert's algorithm described above, we can obtain the Seifert surface for any knot or link.

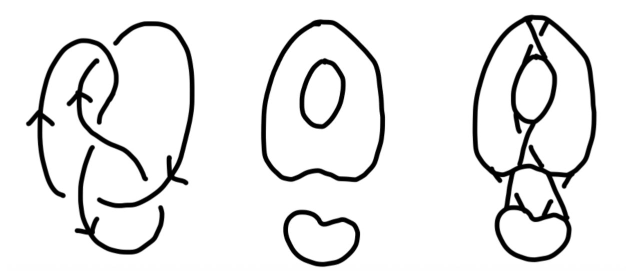


Figure 6. Construction of a Seifert Surface

To this surface, we can associate a matrix known as the Seifert matrix, from which many knot invariants are calculated.

Consider a fixed Seifert surface for our knot. If the surface has genus g, then we choose $2g$ simple oriented curves on the surface. For any simple oriented curve α on the surface, we can form a 'positive push off' curve, α^+. The positive push off, α^+, runs parallel to α, but pushed slightly in the positive direction. It is important to be careful when drawing the positive push off curves and their crossings with the other curves and the surface. The Seifert matrix, V, compares all curves to all push offs, i.e., $V_{ij} = lk(x_i, x_j^+)$. Although the Seifert matrix is not invariant, many invariants are calculated using the matrix. Figure 7 shows the drawing of these oriented curves and their positive push offs on the Seifert surface for a trefoil.

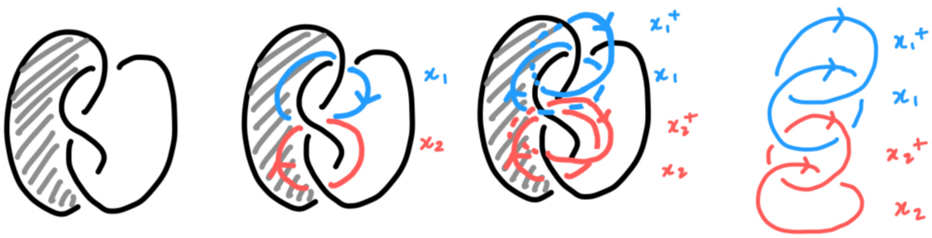


FIGURE 7. Forming positive push off curves on the trefoil surface

The Seifert matrix for the trefoil based off this diagram would be as follows.

$$V = \begin{bmatrix} lk(x_1, x_1^+) & lk(x_1, x_2^+) \\ lk(x_2, x_1^+) & lk(x_2, x_2^+) \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}$$

Definition 2.8. The **Alexander polynomial** or $\Delta(t)$ of a knot is $\det(V - tV^T)$ where V is the Seifert matrix associated to the knot.

Once again, we can continue with the trefoil example. Earlier, it is shown that a Seifert matrix for the trefoil is $V = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}$.

$$\Delta(t) = \det(V - tV^T) \tag{1}$$
$$= \det(\begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} - t\begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}) \tag{2}$$
$$= \det \begin{pmatrix} 1-t & -1 \\ t & 1-t \end{pmatrix} \tag{3}$$
$$= (1-t)^2 - (-t) \tag{4}$$
$$= 1 - 2t + t^2 + t \tag{5}$$
$$= 1 - t + t^2 \tag{6}$$

Therefore, the Alexander polynomial of the trefoil is $1 - t + t^2$.

The Alexander polynomial does not exactly determine the knot. Various knots have the trivial polynomial, meaning one cannot distinguish the unknot using the Alexander polynomial. Nevertheless, it is a fundamental measure that holds a lot of valuable information on the knot.

An important knot invariant that depends on the Seifert matrix of a knot is the signature, $\sigma(K)$.

From elementary algebra, it is known that given an $n \times n$ symmetric matrix A with entries in $\mathbb{R}$, there exists an invertible real matrix P that diagonalizes it; i.e. such that $PAP^T = B$, where B is diagonal. Naturally, we can assume that $\det P = \pm 1$, as there always exists an orthogonal P. Let $\{a_1, a_2, ..., a_n\}$ be the entries in the diagonal of B. The signature of A is the number of $a_i < 0$ subtracted from the number of $a_i > 0$.

$$\sigma(A) = \text{number of positive eigenvalues in A} - \text{number of negative eigenvalues in A}$$

Definition 2.9. The **signature** of a knot K, $\sigma(K)$ is the signature of $V + V^T$, where V is the knot's associated Seifert matrix.

The unknotting number is another example of a fundamental knot invariant.

Definition 2.10. The **unknotting number** $u(K)$ of a knot K is defined as the minimum number of crossing changes required to transform K to the unknot.

FIGURE 8. Unknotting Sequence of 5_1

We refer to the mirror image of a knot K as $\overline{K}$. It is clear that $u(K) = u(\overline{K})$. If knot K has unknotting number $u(K) = n$, there is a sequence involving n crossing changes that results in the unknot. By simply 'holding up a mirror' to the sequence, we obtain a sequence for the unknotting of the mirror image of K, $\overline{K}$.

Although the unknotting number is difficult to compute, there are some theorems that have furthered understanding surrounding the invariant. For example, it is a fact that signature offers a lower bound to the unknotting number. A rigorous proof can be found in Murasugi's paper where he first presents the theorem. [8]

$$|\sigma(K)|/2 \leq u(K)$$

Proof. Let V be a $2g \times 2g$ Seifert matrix for knot K, where g is the genus of the Seifert surface. The signature of a knot, $\sigma(K)$, is the signature of the symmetrized Seifert matrix, $A = V + V^T$.

As we know, $u(K) = n$ is the number of crossing changes required to turn K into the unknot. There exists an unknotting sequence $K, K_1, ..., K_n$ where K_n is the unknot, and K_i to K_{i+1} requires one crossing change.

Let us start with some knot K_i from the sequence. Recalling the manner in which we construct a Seifert surface, we can realize that a single crossing change on a knot flips the twist of the band corresponding to that crossing. This is equivalent to adding a band to the surface. Adding a single band to the surface modifies only the curves that pass directly through the new band. We obtain knot K_{i+1} after this single crossing change. The Seifert matrix of K_{i+1} is now $V' = V \pm vv^T$, changing the original Seifert matrix V by a rank 1 modification. The signature of a matrix is the number of positive eigenvalues minus the number of negative eigenvalues. A rank 1 change to the matrix changes the number of positive or negative eigenvalues by at most 1. Consequently, $\sigma(K) = sign(A)$ changes by at most 2: $|(1) - (-1)| = 2$. Therefore, we can see that a single crossing change (from K_i to K_{i+1}) changes $\sigma(K)$ by at most 2.

The signature of the unknot is 0. We need $u(K)$ crossing changes to unknot K, and a single crossing change changes $\sigma(K)$ by at most 2.

$$|\sigma(K)| \leq 2u(K)$$

$$\sigma(K)/2 \leq u(K)$$

□

It is natural to ask how one might combine two knots. One such way is the connected sum operation.

Definition 2.11. Suppose we are given the knot diagrams for two knots K_1 and K_2. By deleting a short arc from each disjoint knot diagram and connecting the endpoints using two new arcs naturally, we can get a diagram for the **connected sum,** $K_1 \# K_2$**.**

(A) Disjoint Knots K and J

(B) Connect Sum $K\#J$

It is using connected sum that the notion of prime knots is defined. A knot is said to be prime if it cannot be written as a connected sum of two nontrivial knots. Further, every knot can be decomposed into connected sums of prime knots. The connected sum operation is kind, with many invariants behaving well under it. For example, signature is additive under connected sum.

3. Unknotting Number Under Connected Sum

Conjecture 1.1 has been disproven by Brittenham and Hermiller recently. [2] Consider the knot $K = 7_1$, which is the $(2,7)$ torus knot. The connected sum $L = K\#\overline{K}$ is the original counter example to the conjecture.

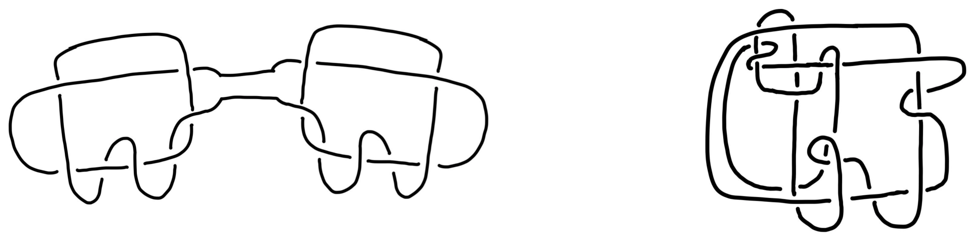


Figure 10. Diagrams of Connect Sum of $7_1\#\overline{7_1}$

K has signature (from Definition 2.9) $\sigma(K) = 6$. Because signature must be less than or equal to twice the unknotting number, it must be that unknotting number of $K = 7_1$ is at least 3. By looking at the diagram, there is an immediate unknotting sequence that requires only 3 crossing changes, so $u(K)$ is 3. Since mirror image preserves unknotting number, $u(\overline{K}) = 3$. If Conjecture 1.1 were to hold true, we would have that $u(K) + u(\overline{K}) = u(K\#\overline{K}) = 6$. Brittenham and Hermiller find that, surprisingly, $u(K\#\overline{K}) \leq 5$. [2] The explicit and careful sequence of crossing changes that results in the unknot can be found in their paper.

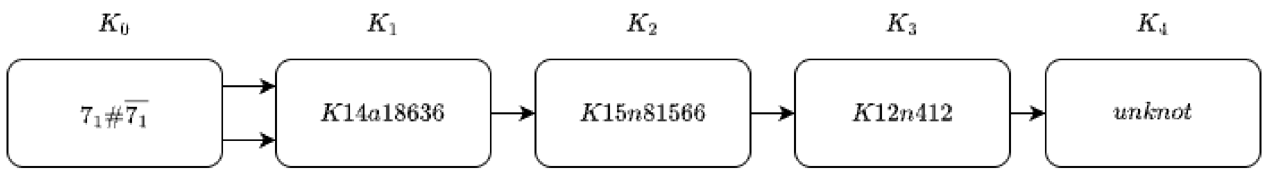


Figure 11. The Solution's Unknotting Sequence

In their paper providing a remark on Brittenham and Hermiller's findings, Wang and Zhang highlight all five crossing changes in a singular diagram of the original connected sum. [13] They leave it to the reader to verify that the final diagram is the unknot. We verify it ourselves. Below is our drawing of the final diagram, that is in fact the unknot, with the 5 crossings highlighted that were changed from the $7_1\#\overline{7_1}$ knot. We use this drawing in the PLink editor. SnapPy returns that the fundamental group of the knot is $\mathbb{Z}$, verifying that it is the unknot.

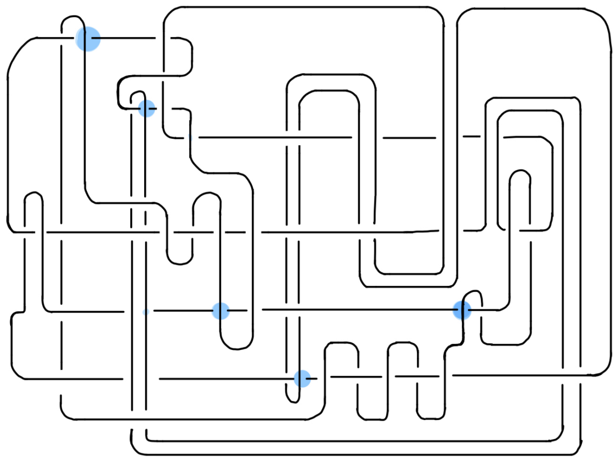


FIGURE 12. The unknot with crossing changes from $7_1\#\overline{7_1}$

Our simple code along with output is provided below.

```
import snappy

%gui tk
M = snappy.Manifold()

Starting the link editor.
Select Tools->Send to SnapPy to load the link complement.
2026-02-10 13:12:56.914 python[26159:2090330] The class 'NSOpenPanel'
overrides the method identifier.
This method is implemented by class 'NSWindow'

New triangulation received from PLink!

print(M.fundamental_group(simplify_presentation="True"))
K = M.link()
print(K)

Generators:
a
Relators:

<Link: 1 comp; 58 cross>
```

In their search for this counterexample, the authors assume the unknotting number is additive. They want to build a collection with unknotting number 4 but relatively small signature. Conveniently, signature is both additive under connected sum and negative under mirror image. When building connected sums of a knot with its mirror image, it is easy to get a relatively large unknotting number with a relatively small signature. They specifically built connected sums with knots that had unknotting number 3 and 4 respectively. Under their assumption, the connected sum would have unknotting number 7. They then change three crossings, to have connected sums with what 'should' be unknotting number 4. This results in 240 knots.

Through their search for a counterexample in their database, they find a mutant knot with unknotting number 3 that is one crossing change away from one of the 240 unknotting number 4 knots. The reason for this turns out to be that their assumption was false! One of the crossing changes when building their collection resulted in a knot (connect sum) with lower than expected unknotting number, and unknotting number did not hold under connected sum.

The knot that led to their initial counterexample was the connect sum $7_1\#10_{139}$. As $u(7_1) = 3$ and $u(10_{139}) = 4$, the connect sum should have unknotting number 7. (In this case, it is true.) As described above, they change three crossings to get to the desired unknotting number 4. However, the three crossing changes result in $K14a18636$, which we can recognize as the second knot from the unknotting sequence in Figure 11. From Figure 11, we can see only three crossing changes are required to transform to the unknot, implying that $K14a18636$ has unknotting number 3, rather than the expected 4.

The culprit was the intermediate $7_1\#\overline{7_1}$, that was a result of one of the three crossing changes between $7_1\#10_{139}$ and $K14a18636$.

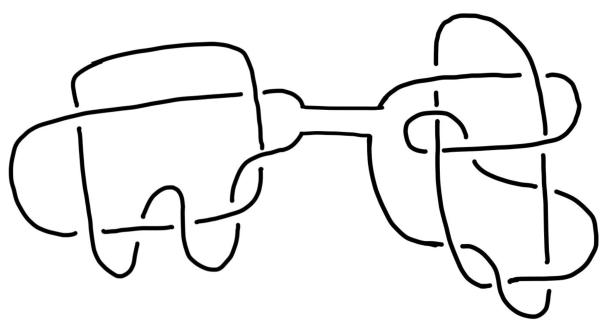


FIGURE 13. Diagram of $7_1\#10_{139}$

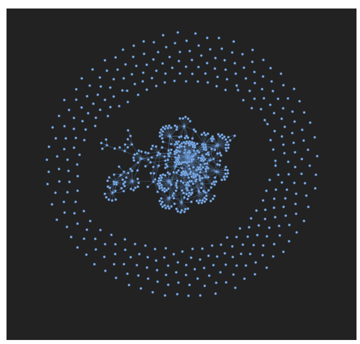
Definition 3.1. Knot K_1 is **Gordian adjacent** to knot K_2 if K_1 appears as an intermediate knot in the unknotting sequence of K_2.

So, $K14a18636$ and all other knots in the Figure 11 sequence are Gordian adjacent to $7_1\#\overline{7_1}$. Suppose 7_1 is Gordian adjacent to L_1, L_2. The minimal unknotting sequence for $L_1\#\overline{L_2}$ contains the unknotting sequence for $7_1\#\overline{7_1}$, and the unknotting sequence for $7_1\#\overline{7_1}$ is already smaller than conjecturally expected. So, $u(L_1\#\overline{L_2})$ will be less than $u(L_1) + u(\overline{L_2})$. In this way, the initial counterexample gives way to an infinite class of examples.

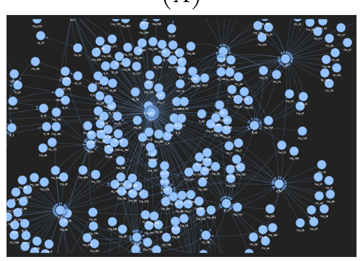
It is only natural to ask what led to this counterexample specifically. Was it chance or something special about the connected sum? To begin to explore Gordian adjacency across knots, we build a graph network where there is a directed edge from knot K_2 to the knot K_1 that is Gordian adjacent to it. The graph is available here. Our functions are all provided in Section 5. Figure 14 shows the the graph at different zoom levels. When the user's mouse hovers over a node on the graph, the knots that are adjacent to it are listed. The nodes can also be dragged around to view various edges more clearly.

We obtain all necessary knot data from KnotInfo. [7] After reading this data, it is split into python dictionaries of the knot name with respective unknotting number and DT code. We restrict our graph to prime knots with crossings under 11, for the purpose of compile time. This results in around 800 knots. However, we can easily expand to include all knots up to 15 crossings using the same code. We load all this knot information using the `load_knotinfo` function.

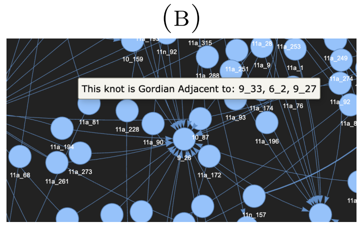
For each knot, we check whether a crossing change at each index in the DT code results in a gordian adjacent knot: this gives us around 7500 crossings to check. We use SnapPy to identify knots. Because SnapPy is unable to identify non-hyperbolic knots itself, we write a function that directs the `identify_knot_safe` function to the `identify_non_hyperbolic` function. For knots under 11 crossings and torus knots, the Alexander polynomial, described in Definition 2.8, acts as an invariant. Although SnapPy cannot identify the knot, given the DT code, it can find the Alexander polynomial. So, we generate an Alexander polynomial dictionary using KnotInfo, and use it to identify non hyperbolic knots, as well. We run this identification on every crossing change for a knot using the `find_neighbors` function, and run the whole batch using the `run_batch_safe` function.



(A)



(B)



(C)

FIGURE 14. The Gordian Adjacency Graph Network

Simple code to generate the graph is provided below.

```python
uk_dict, dt_dict, ap_dict, certain = attributes.load_knotinfo('KnotInfo Complete Data.csv', max_crossings=11)
edges, graphed_edges = attributes.run_batch(uk_dict, dt_dict, ap_dict)

from pyvis.network import Network
net = Network(notebook=True, cdn_resources='in_line', bgcolor='#222222', font_color='white', directed=True)

for k in dt_dict.keys():
    net.add_node(str(k), label=str(k))
net.add_edges(list(graphed_edges))

neighbor_map = net.get_adj_list()
for node in net.nodes:
    node_neighbors = neighbor_map[node['id']]
    node['title'] = 'This knot is Gordian Adjacent to: ' + ', '.join(map(str, node_neighbors))

net.toggle_physics(True)
net.show('gordianadj.html')
```

Using NetworkX and pyvis, one can also ask many interesting questions regarding this graph, such as what the most connected node is, or what the longest chain within the graph is.

4. Mutation

This novel result is surprising, but most interesting is the manner in which they came across it. The authors originally wanted to find a counterexample to Conjecture 1.2.

Definition 4.1. The part of a knot K inside some sphere that intersects K at exactly 4 points is known as a **tangle**. A **mutation** of K replaces the tangle by an 180 degree rotation of itself.

Figure 15 shows famous mutant knot pair, the Kinoshita Terasaka knot on the left and the Conway knot on the right.

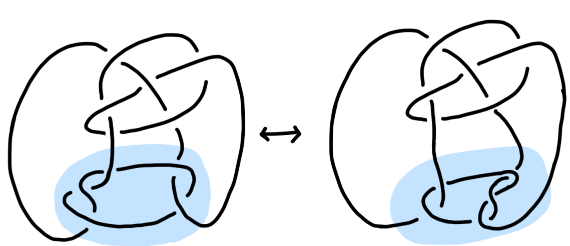


Figure 15. Mutation on the Kinoshita-Terasaka knot and Conway knot

Recall from the previous section that the authors originally set out to answer Conjecture 1.2 regarding mutation, and began by building a specific collection of 240 knots with what they thought to be unknotting number 4.

The authors also had a list of mutant pair target knots that were one crossing change away from the unknotting number 4 collection. If one of the mutant target knots had unknotting number 3 and was one crossing change away from some knot of the 240, this would mean unknotting number was not invariant under mutation. Or, this would mean their assumption when building the collection of 240 knots was wrong– that unknotting number was not additive under connected sum. As we have seen, this was a counterexample to Conjecture 1.1, and the mutation conjecture remains open.

Finding a solution to Conjecture 1.2 remains a very difficult problem. In solving Conjecture 1.1, Brittenham and Hermiller heavily use the fact that signature provides a lower bound to unknotting number. By building a collection where $\sigma(K)/2$ differs from $u(K)$ by more than one, they are able to find a counterexample. Due to signature being a well behaved invariant, the collection offers connected sums with relatively small signature and comparatively large unknotting number.

Upon initial inspection of the mutation problem, one may notice the similar lower bound that various mutation-invariant knot invariants offer. The τ invariant is an integer invariant originally computed from a knot's knot Floer homology. It is originally defined by Oszvath and Szabo in 2003. [11]

$$|\tau(K)| \leq u(K)$$

Similarly, the s invariant is an integer invariant defined by Rasmussen. It is derived from the Khovanov homology. [12]

$$|s(K)| \leq g_4(K) \leq u(K)$$

where g_4 is the slice genus of a knot.

Finding the counterexample required that signature was additive under connected sum and negative under mirror image to make knots with a gap of more than one between $|\sigma(K)/2|$ and $u(K)$. We can quickly realize that manipulating invariants s and τ in the same way would be much more difficult. Further, these invariants are much harder to compute. Meanwhile, other invariants do not detect mutation. It may be possible to explore reinforcement learning or deep learning methods. Shortly before the counterexample was found to the conjecture, a reinforcement learning strategy was employed to find minimal unknotting sequences for knots with up to 200 crossings. This offers an upper bound to the unknotting number. [1] Overall, the problem remains very mathematically difficult.

5. Code Reference

5.1. **Loading Knot Information.** Below are the functions used for loading knot information from KnotInfo in the desired format.

```
def clean_unknotting(val):
    val = str(val).strip()
    if val.isdigit():
        return int(val)
    else:
        return np.nan
```

```

def clean_dt(val):
    val = str(val).strip()
    clean_val = ast.literal_eval(val)
    return clean_val

def clean_ap(val):
    val = str(val).strip()
    dict = polynomial_to_dict(val)
    return dict

def crossing_change(dt_code, index):
    test_code = dt_code.copy()
    test_code[index] = -dt_code[index]
    return test_code

def load_knotinfo(path = ‘KnotInfo Complete Data.csv', max_crossings = 11):

    df = pd.read_csv('KnotInfo Complete Data.csv')
    df.drop(index = 0, inplace=True)
    df.columns = df.columns.str.strip().str.lower().str.replace(‘ ', ‘_')

    # Filter to knots with desired amount of crossings and make the smaller dataframe
    df_small = df[df['crossing_number'].astype(int) <= 11].copy()

    #Making dictionaries for dt and unknotting number
    dt_dict = df_small.set_index('name')['dt_notation'].to_dict()
    uk_dict = df_small.set_index('name')['unknotting_number'].to_dict()
    ap_dict = df_small.set_index('name')['alexander_polynomial'].to_dict()

    #making ‘certain' dataframe to make sure all entries have a forsure unknotting number
    df_small['certain_uk'] = df_small['unknotting_number'].apply(clean_unknotting)
    certain = df_small[df_small['certain_uk'].notna()].copy()
    #uncertain = df_small[df_small['certain_uk'].isna()].copy() #can print # that don't have certain unknotting number

    #cleaning dt code in smaller dataframe and certain dataframe
    df_small['dt_notation'] = df_small['dt_notation'].fillna('[]')
    df_small['dt_list'] = df_small['dt_notation'].apply(clean_dt)
    certain['dt_notation'] = certain['dt_notation'].fillna('[]')
    certain['dt_code'] = certain['dt_notation'].apply(clean_dt)

    certain['alexander_polynomial'] = certain['alexander_polynomial'].fillna('0')
    certain['alexander_polynomial'] = certain['alexander_polynomial'].apply(clean_ap)
```

```

    #update dictionaries
    uk_dict  = dict(zip(certain['name'], certain['certain_uk'].astype(int)))
    dt_dict = dict(zip(certain['name'], certain['dt_code']))
    ap_dict = dict(zip(certain['name'], certain['alexander_polynomial']))

    #unknot should be the ending in our graph
    uk_dict['0_1'] = 0
    dt_dict['0_1'] = []
    ap_dict['0_1'] = ‘1’

    return uk_dict, dt_dict, ap_dict, certain
```

5.2. **Knot and Neighbor Identification.** Below are the functions used to identify knots and their gordian adjacent neighbors, upto the desired number of crossings.

```
def polynomial_to_dict(poly_str):
    #function to turn polynomials into dictionary format where keys are exponents and values are coefficients

    poly_str = poly_str.replace(" ", "")
    poly_str = poly_str.replace("t", "x")


    pattern = r'([\+\-]?\d*)(x(?:\^(\d+))?)?'

    terms = re.findall(pattern, poly_str)
    poly_dict = {}

    for coeff, x_part, exp in terms:
        if not coeff and not x_part:
            continue

        # parse coefficient
        if coeff in (‘’, ‘+’):
            c = 1
        elif coeff == ‘-’:
            c = -1
        else:
            c = int(coeff)

        # parse exponent
        if not x_part:
            e = 0
        elif not exp:
            e = 1
        else:
            e = int(exp)

        poly_dict[e] = poly_dict.get(e, 0) + c
```

```

    return poly_dict

def identify_non_hyperbolic(K, ap_dict):
    #identifying non-hyperbolic knots using alexander polynomial, since
   snappy can't identify them
    try:
        ap = K.alexander_polynomial()
    except Exception:
        return None

    ap = polynomial_to_dict(str(ap))

    if ap in ap_dict.values():
        for knot, poly in ap_dict.items():
            if poly == ap:
                return knot

    return None

def identify_knot_safe(dt_code, ap_dict):
    if not dt_code:
        return '0_1'                    # if no dt code, return unknot

    K = snappy.Link('DT:' + str(dt_code))

    try:
        K.simplify("global")
    except Exception:
        pass

    M = K.exterior()
    sol = M.solution_type()
    matches = M.identify()

    if matches == []:
        return identify_non_hyperbolic(K, ap_dict)   # if no hyperbolic
   solution, try to identify as non-hyperbolic using alexander polynomial

    #if 'degenerate' in sol:
        return identify_non_hyperbolic(K, ap_dict)   # non hyperbolic

    #if 'not attempted' in sol:
        return None                     # snapPy gave up

    return matches[0].name() if matches else None

def find_neighbors(knot_name, uk_dict, dt_dict, ap_dict):
    u_K = uk_dict.get(knot_name) #get unknottig number of main knot
    dt_code = dt_dict.get(knot_name) #get dt code of main knot
```

```
    if u_K is None:
        return []
    neighbors = [] #list of (neighbor_name, crossing_index) pairs
    for i in range(len(dt_code)):     #for each value / crossing in the dt code, flip it and see if we get a neighbor with unknotting number one less
        test_code = crossing_change(dt_code, i)
        neighbor = identify_knot_safe(test_code, ap_dict) #identify the neighbor knot from the modified dt code
        if neighbor and uk_dict.get(neighbor) == (u_K - 1): #check if neighbor is valid and has unknotting number one less
            neighbors.append((neighbor, i))
    return neighbors

def run_batch(uk_dict, dt_dict, ap_dict):
    tasks = [] #all the crossing change tasks to process
    for name, dt in dt_dict.items():
        if not dt:
            continue
        tasks.append((name, dt))

    with tqdm.tqdm(total=len(tasks)) as pbar:
        edges = []
        graphed_edges = []

        for task in tasks:
            neighbors = find_neighbors(task[0], uk_dict, dt_dict, ap_dict)
            for neighbor in neighbors:
                edge = (task[0], neighbor[0], neighbor[1]) # (knot, neighbor, crossing change index)
                edges.append(edge)
                graphed_edges.append((task[0], neighbor[0]))
            pbar.update(1)

        graphed_edges = list(set(graphed_edges))

    return edges, graphed_edges
```


Acknowledgements

I would first like to thank thesis advisor, Dr. Maggie Miller, for teaching me everything I know about knots and helping me understand topics that otherwise felt overwhelming and incomprehensible. Thank you to thank my parents for pushing me to study the things I enjoy most. I have had the privilege of loving what I do every single day. Thank you to Ava and Vaishali for for listening to me ramble on about everything new I was learning. Finally, thank you to UT and the Dean's Scholars program for offering a mathematics education and facilitating an environment that has showed me the joys of the subject.

References

[1] Taylor Applebaum, Sam Blackwell, Alex Davies, Thomas Edlich, András Juhász, Marc Lackenby, Nenad Tomašev, and Daniel Zheng. The unknotting number, hard unknot diagrams, and reinforcement learning. *Experimental Mathematics*, 0(0):1–19, 2025.

[2] Mark Brittenham and Susan Hermiller. Unknotting number is not additive under connected sum. *arXiv e-prints*, page arXiv:2506.24088, June 2025.

[3] Marc Culler, Nathan M. Dunfield, Matthias Goerner, and Jeffrey R. Weeks. SnapPy, a computer program for studying the geometry and topology of 3-manifolds. Available at `http://snappy.computop.org` (DD/MM/YYYY).

[4] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

[5] Robion Kirby. Problems in low-dimensional topology. In Michael Freedman and Kirby Robion, editors, *Geometric topology*, AMS/IP Studies in Advanced Mathematics, pages 35–473. American Mathematical Society, Providence, RI, 1995.

[6] Xiao-Song Lin and Sam Nelson. On generalized knot groups, 2006.

[7] Charles Livingston and Allison H. Moore. Knotinfo: Table of knot invariants. URL: `knotinfo.org`, Current Month Current Year.

[8] Kunio Murasugi. On a certain numerical invariant of link types. *Transactions of the American Mathematical Society*, 117:387–422, 1965.

[9] SAM NELSON and WALTER D. NEUMANN. The 2-generalized knot group determines the knot. *Communications in Contemporary Mathematics*, 10(supp01):843–847, November 2008.

[10] Peter Ozsvath and Zolta Szabo. Knot floer homology, genus bounds, and mutation, 2004.

[11] Peter Ozsváth and Zoltán Szabó. Knot floer homology and the four-ball genus. *Geometry amp; Topology*, 7(2):615–639, October 2003.

[12] Jacob A. Rasmussen. Khovanov homology and the slice genus, 2004.

[13] Chao Wang and Yimu Zhang. A remark on the counterexample to the unknotting number conjecture, 2025.

[14] H. Wendt. Die gordische auflösung von knoten. *Mathematische Zeitschrift*, 42:680–696, 1937.