An Algorithm to Detect Flypes in Minimal Alternating Knot Diagrams

Trivan Menezes

Abstract

A *knot* is an embedding of a circle in 3-space. A knot can be represented by tangling a string and fusing its endpoints together. A knot diagram D is a projection of a knot onto a two-dimensional plane. In such a projection, the points of self-intersection are known as *crossings*. A knot is considered *alternating* if an alternating pattern of over and under passes are encountered at crossings while tracing diagram D in a single direction. Furthermore, a diagram D is considered minimal if the diagram contains the minimum number of crossings out of all diagrams of the same knot type. A tangle T in a knot diagram D is a region R bounded by a simple closed curve that intersects the knot diagram transversely at four points. A *flype* is a move in which a tangle is flipped 180 degrees such that a neighboring crossing moves to the other side of the tangle. A minimal diagram of a given alternating knot is related to any other minimal diagram of that knot type by a finite sequence of flypes. This paper outlines an algorithm which detects flypes in a knot diagram by processing an encoding of a knot diagram called planar-diagram code (PD code). Combining this algorithm with one which executes a flype in a knot diagram will yield a program which can perform all possible flypes in a given knot diagram and returns a list of all minimal diagrams of the given alternating knot. With such a list, functions which distinguish knots from one another, called *knot invariants*, can be calculated.

1 Introduction - Knot Theory Background

Knot theory is the study of knots. A *knot* is an embedding of a circle in 3-space, and a knot can be represented by tangling a string and fusing its endpoints together. This forms a closed curve in space that does not have any points of self-intersection. A knot diagram D is a projection of a knot onto a two-dimensional plane. In such a projection, the double points are known as *crossings*, and we indicate which strand passes over another strand in the diagram by a small break in the underpass, see Figure 1. We only allow a finite number of crossings in a given projection of a knot. A knot is considered *alternating* if an alternating pattern of over and under passes are encountered at crossings while tracing diagram D in a single direction (see Figure 1). Not all knots are alternating. However, given the set of knots of crossing number n, there are infinitely many alternating knots as n approaches infinity [9]. The knots that are not alternating are termed *non-alternating.* Knots are thought of as being made of elastic string. This string can be bent or deformed, yielding various configurations of the same knot. The set of all such configurations comprises the *knot type* [3]. Thus a knot type yields infinitely many different diagrams that can contain an arbitrarily large number of crossings. Furthermore, a diagram D is considered minimal if the diagram contains the minimum number of crossings out of all diagrams of the same knot type [1]. We note that a knot type can have multiple different minimal alternating diagrams. The goal of this research is to design and program a procedure which identifies all minimal diagrams of a given alternating knot type.

A tangle T in a knot diagram D is a region R bounded by a simple closed curve that

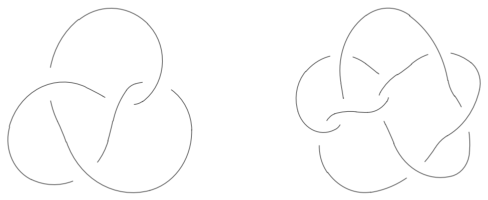

Figure 1: Left: A 4 crossing alternating knot diagram. Right: An 8 crossing non-alternating knot diagram.

intersects the knot diagram transversely at four points [1]. Tangle T contains two arcs that are part of the knot diagram. A *flype* is a move in which a tangle is flipped 180 degrees such that a neighboring crossing moves to the other side of the tangle (see Figure 2). Flypes relate the minimal diagrams of an alternating knot type. That is, a minimal diagram of a given alternating knot is related to any other minimal diagram of that knot type by a finite sequence of flypes [8].

Theorem The Tait Flyping Conjecture [7][8]: Given two reduced alternating projections of the same knot, they are equivalent on the sphere if and only if they are related by a series of flypes.

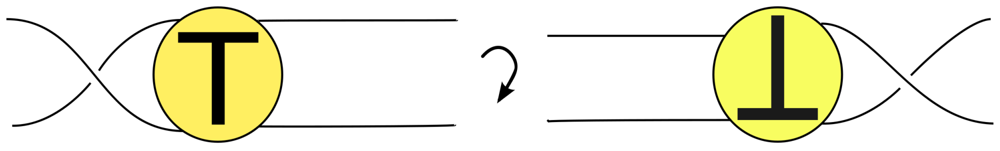

Figure 2: A flype of tangle T in a knot diagram.

A knot diagram is *oriented* if its strands are marked with arrows denoting a consistent direction of traversal (the orientation). A flyping circuit is a decomposition of a knot diagram into a set of tangles as shown in Figure 3. Furthermore, we assume no tangle in the circuit can be split into two tangles that increase the number of tangles in the flyping circuit. If one of tangles consists of a single crossing then this crossing can be flyped to occur between any two tangles in the flyping circuit. In an oriented knot diagram, a flyping circuit is considered either *parallel* or *anti-parallel*. A flyping circuit is *parallel* if any pair of strands connecting two tangles are oriented in the same direction (see Figure 3). Otherwise, the flyping circuit is considered *anti-parallel*.

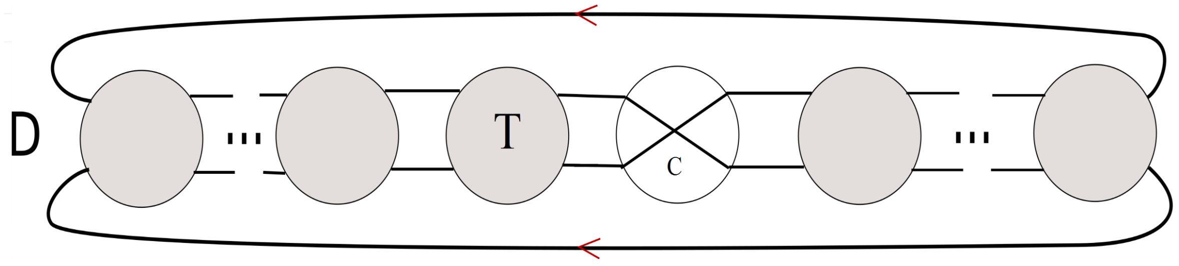

Figure 3: A parallel flyping circuit D (directions labeled).

Planar diagram code (PD code) is an encoding of a knot diagram that can be processed and manipulated by a computer. Particularly, the KnotTheory package for Wolfram Mathematica enables a user to perform various routines using planar diagram codes [2]. In order to construct such a code, begin by choosing an arbitrary point on a knot diagram D. Choose an orientation (direction of traversal), and traverse around the knot diagram until the starting point is reached again. Along the way, label the strands with increasing natural numbers, with the first strand having an index of 1. Once a crossing is encountered, label the next strand with the incremented index. For a diagram D with n crossings, each crossing is traversed twice (each crossing consists of two strands which overlap one another).

As a result, the upper bound of the index is $2n$. Because two strands appear to overlap at each crossing, we can associate a quadruple of numbers to each crossing. The n sets of four indices (one set for each crossing) comprise the planar diagram code. The first index in each quadruple set is the incoming understrand (according to the established orientation), and the order of the other three indices is determined by moving counter-clockwise around the crossing from the incoming understrand. For example, the PD code of the 6-crossing knot in Figure 4 is: $\{[1, 9, 2, 8], [3, 10, 4, 11], [5, 3, 6, 2], [7, 1, 8, 12], [9, 4, 10, 5], [11, 7, 12, 6]\}$.

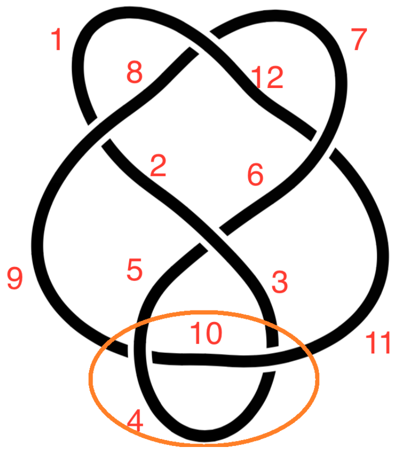

Figure 4: A 6 crossing knot diagram with strands labeled. The circle indicates a two-crossing tangle, with the following PD representation: $\{\{9, 4, 10, 5\}, \{3, 10, 4, 11\}\}$.

Knot invariants are functions that are used to classify and distinguish knots. In numerous cases, knot diagrams can be used to compute these functions. Knot invariants will return the same result when computed using two different knot diagrams of the same knot type. Thus, the result remains *invariant* across different representations and projections of the same knot. Given a knot invariant f and let K_1 and K_2 be two knot diagrams, if $f(K_1) \neq f(K_2)$, then K_1 and K_2 are different knots. However, the converse is not true. If

$f(K_1) = f(K_2)$, then K_1 and K_2 may not necessarily be the same knots, for different knots can have equivalent invariant values. The goal of this project is to create a computer program (using the KnotTheory package in Wolfram Mathematica) which detects all sequences of flypes in a given alternating knot diagram. These flypes will then be executed to obtain a list of all minimal diagrams of the given alternating knot type. Such a list will expedite the process of searching for counterexamples of conjectures and the process of verifying properties of knot invariants. [2]

2 Methods

The following explanation will outline an algorithm used for detecting flypes in a given knot diagram. The output will consist of tangle-crossing pairings, in which the given crossing is able to be flyped about the given tangle. An example of the output generated by the program for the knot diagram in Figure 4 is shown below. A crossing Cr is represented by a single quadruple set of integers from the planar diagram code of the knot diagram. Tangle T is represented by a list of quadruple sets of integers corresponding to the crossings which are encompassed by the given tangle.

$T = \{\{9, 4, 10, 5\}, \{3, 10, 4, 11\}\} \qquad Cr = \{5, 3, 6, 2\}$,

$T = \{\{1, 9, 2, 8\}, \{7, 1, 8, 12\}, \{11, 7, 12, 6\}\} \qquad Cr = \{5, 3, 6, 2\}$,

$T = \{\{1, 9, 2, 8\}, \{5, 3, 6, 2\}, \{7, 1, 8, 12\}, \{11, 7, 12, 6\}\} \qquad Cr = \{3, 10, 4, 11\}$

The tangle T = $\{\{9, 4, 10, 5\}, \{3, 10, 4, 11\}\}$ is highlighted in Figure 4.

2.1 The Algorithm for Flype Detection

Tree Generation

Given the PD-code of a knot diagram D, this algorithm produces a tree T associated to D in which each set of children for a given interior vertex represents a flyping circuit, each leaf of the tree T corresponds to a single crossing in D, and each interior vertex corresponds to a tangle in D. For example, the node 6 in the tree in Figure 5 has the children 3,4,5. Therefore, 3,4,5 represent a flyping circuit. In this case, 4 and 5 are single crossings (leaves in T) that can be flyped about the tangle 3 (an interior vertex in T). The tree T will then be used to obtain a list of the planar-diagram representation of the possible flypes in the given knot diagram.

The algorithm produces the tree T of flypes by tracing a hierarchy of tangles present in knot diagram D. A single crossing can be considered a tangle (by the definition in Section 1), and starting with tangles that are just single crossings, the algorithm detects tangles by searching for other tangles which share two arcs with previously detected tangles. The two tangles T_1 and T_2, which share two arcs, together form a new tangle, T_3 (see Figure 6). In the tree, T_1 and T_2 become the children of T_3. The algorithm constructs a tree (or forest of trees) based on this simple step.

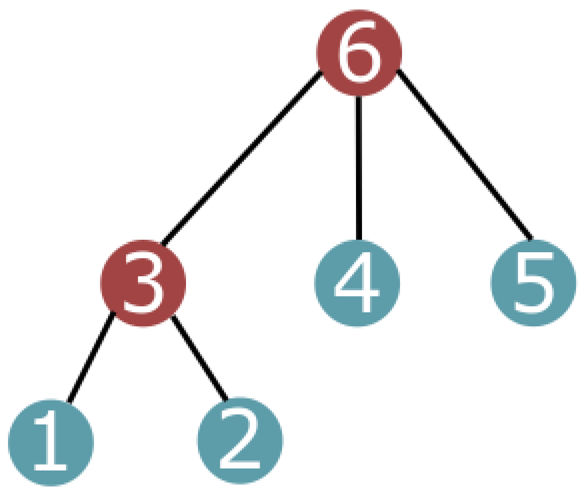

Figure 5: An example of a flype-circuit tree.

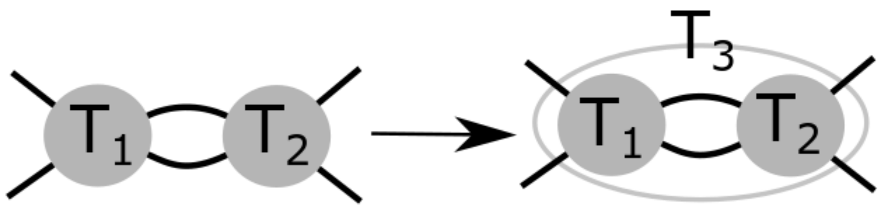

Figure 6: Two tangles which share two arcs can be classified as a single tangle.

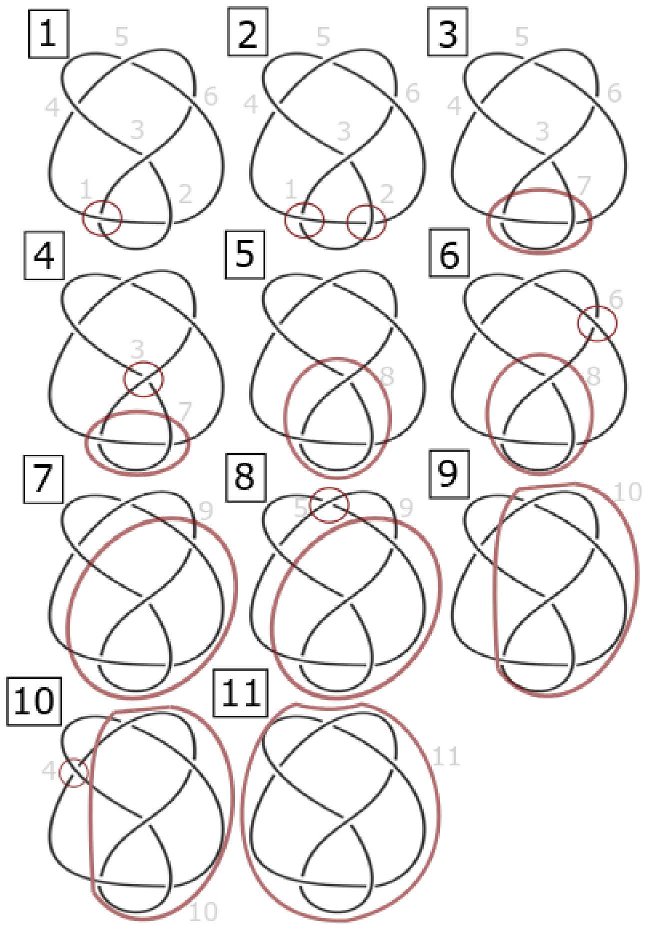

Figure 7: An illustration of the algorithm's execution for the six crossing knot in Figure 4. The corresponding tree is shown in Figure 8.

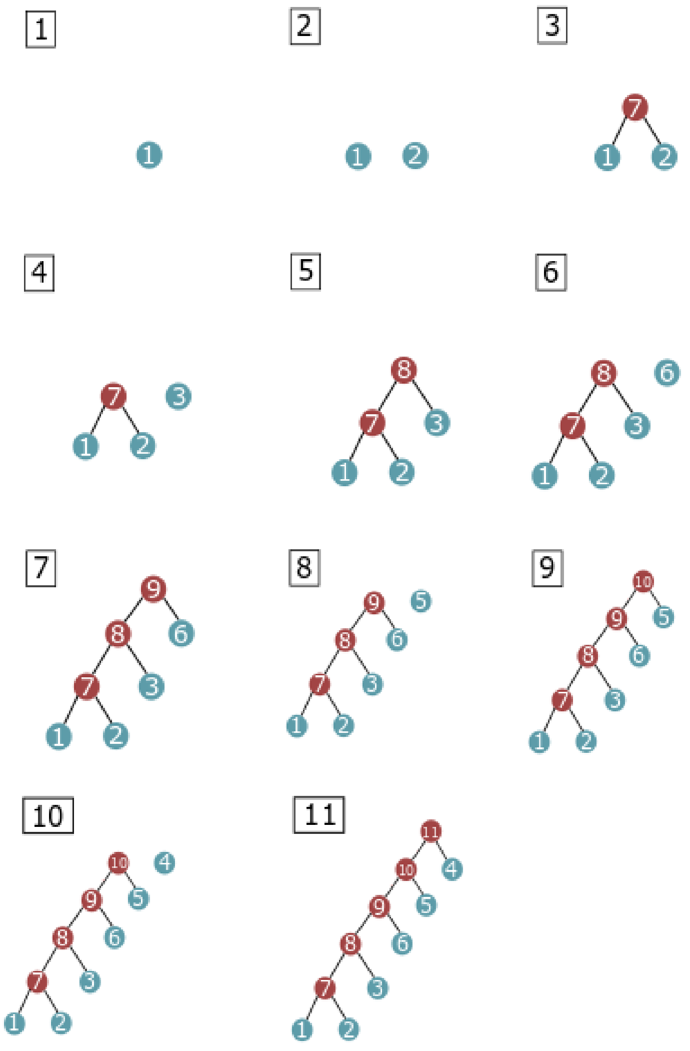

Figure 8: The generation process of the tree T using the steps shown in Figure 7.

An example of the algorithm for producing the flype-circuit tree is provided in Figures 7 and 8. The steps which iterate through the Figures are as follows: 1) An arbitrary crossing 1 is chosen. 2) A crossing is detected that shares two arcs with crossing 1. 3) These two crossings can be labelled as a tangle. This tangle is labelled 7, because the labels 1-6 are reserved for the single crossings in the diagram. In the tree, edges are created which lead from 7 to 1 and 7 to 2. 4) Crossing 3 shares two arcs with tangle 7. 5) Crossing 3 and tangle 7 are grouped into a single tangle, 8. In the tree, edges are created from 8 to 3 and 8 to 7. 6) Crossing 6 shares two arcs with tangle 8. 7) Crossing 6 and tangle 8 are grouped into a single tangle, 9. 8) Crossing 5 shares two arcs with tangle 9. 9) Crossing 5 and tangle 9 are grouped into a single tangle, 10. 10) Crossing 4 shares two arcs with tangle 10. 11) Crossing 4 and tangle 10 are grouped into a single tangle, 11. 12) All tangles in the diagram have been labelled.

Tree Reduction

The tree T produced in Figure 8 is a binary tree that does not yet convey the flyping circuits in the given knot diagram D shown in Figure 7. For example, for the tree shown in Figure 7, the crossing 4 can be flyped about the tangle 8, but the tree T does not convey this. Some edges of the tree T must be eliminated in order for the tree to completely show the flyping circuits in D.

In order to determine which edges of a tree must be eliminated, the position of each tangle, relative to the tangles to which it is connected, must be considered. The relative position of a tangle is determined by tracking the *bounds* of the tangle.

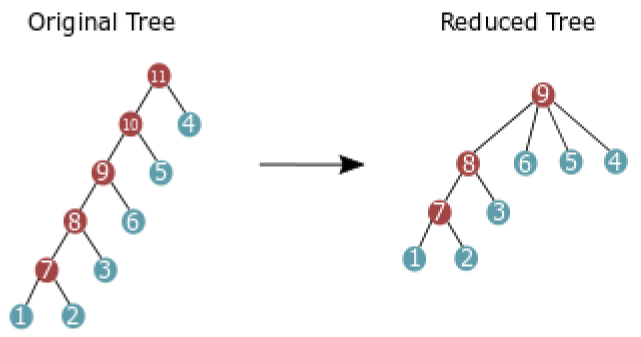

Figure 9: The reduction of the tree created in Figures 7 and 8.

Bounds of a tangle

All tangles that are not single crossings have a pair of bounds. Figure 10 demonstrates examples for the determination of a tangle's bounds. The top shows the initial step: combining two single crossings into a tangle. The bottom shows the step in which the new tangle consists of two other tangles. In both cases, the pair $\{a, b\}, \{e, f\}$ are the bounds of the new tangle, where the indices are the same integer labels as used in the PD-code of the diagram.

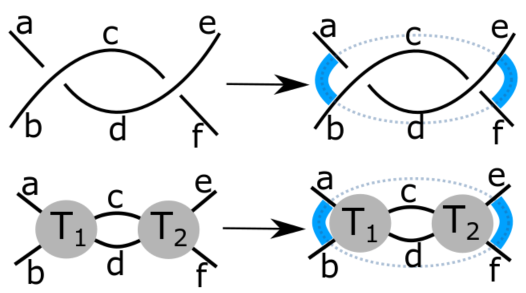

Figure 10: Bounds = {a,b}, {e,f}. The labels represent the numbering of arcs in the PD code.

Eliminating Edges

Assume the tangles T_1 and T_2 are connected to the the tangle T_3. That is, T_1 and T_2 are children of T_3. Also assume that T_3 has its bounds labeled as {a,b}, {e,f}. If {a,b} is also included in T_1's bounds, then the edge connecting T_3 to T_1 will be eliminated. An example is shown in Figure 11. It is possible for the edge from T_3 to T_2 to also be eliminated, depending on whether or not T_2 shares a bounds label with T_3.

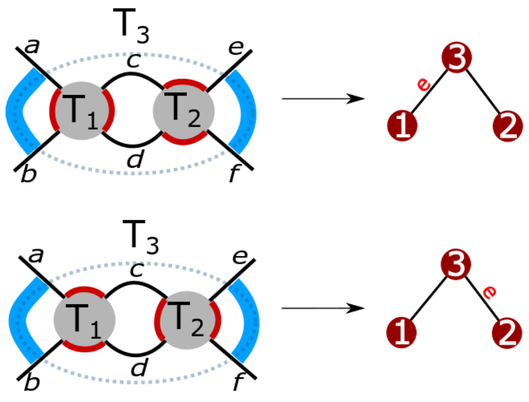

Figure 11: The bounds of T_3 are indicated in blue. The bounds of T_1 and T_2 are each indicated in red.

Top - T_3 and T_1 share the bounds {a,b}. The edge from T_3 to T_1 is marked for elimination.

Bottom - T_3 and T_2 share the bounds {e,f}. The edge from T_3 to T_2 is marked for elimination.

If a tangle T_i is a single crossing and a child of a tangle T_p, then the edge from T_p to T_i is never eliminated.

For the examples shown in Figures 7 and 8, the following changes are made.

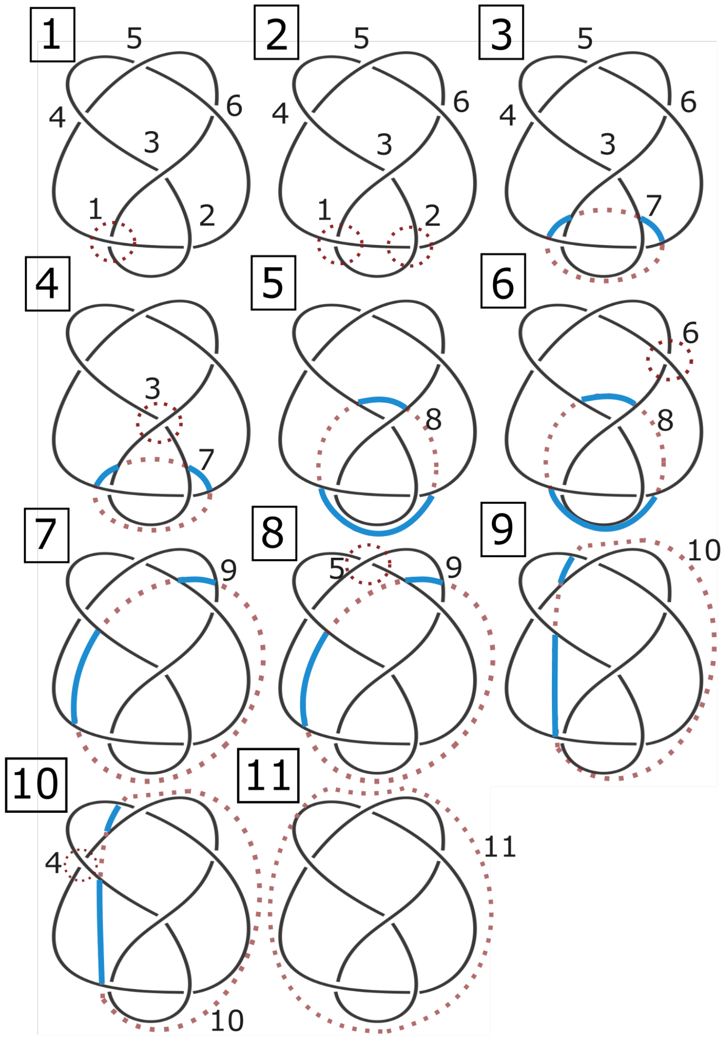

Figure 12: An illustration of the algorithm following the steps of Figure 7 with the addition of the tangles' bounds indicated by blue arcs.

The very last edge detected in the tree (the edge from 11 to 10 in the case of Figure 13) will always be eliminated, regardless of boundary pairs. After all necessary edges have been eliminated, we call T a reduced tree T_r.

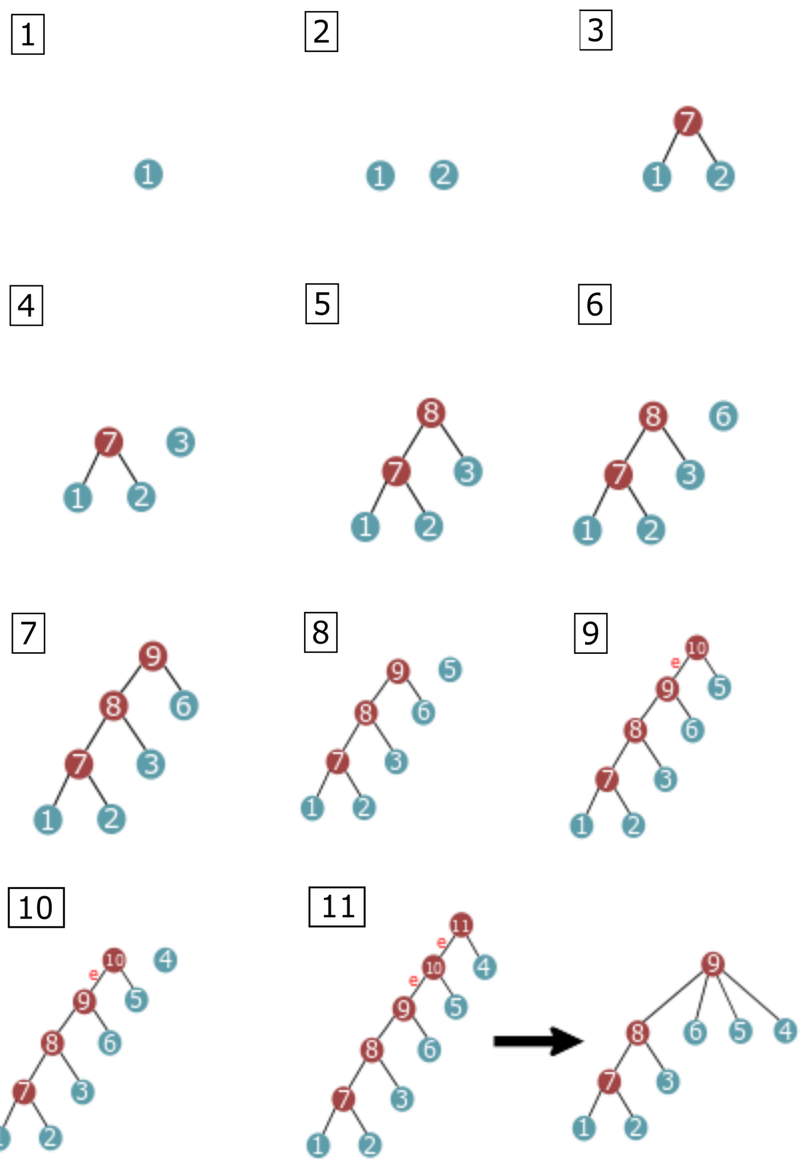

Figure 13: An illustration of the algorithm's execution for the steps shown in Figure 12 with edges marked for elimination based on bounds.

Flype Retrieval

In a reduced tree T_r, the set of children of any interior vertex corresponds to a flyping circuit. Figure 14 displays possible flyping circuits evident in the reduced tree created in Figure 13.

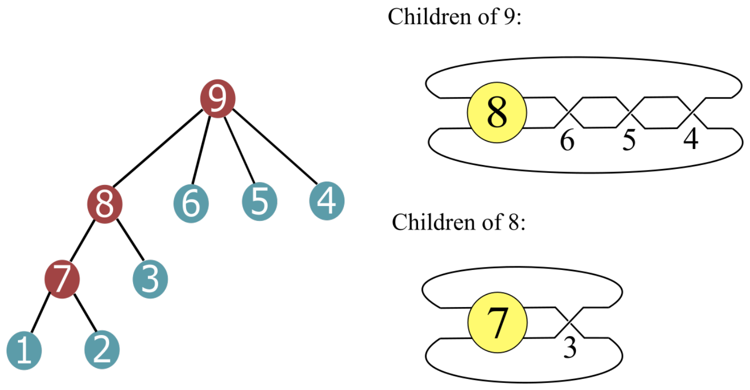

Figure 14: Since there are three interior vertices in the given reduced tree, there is a possibility of three flyping circuits evident in the tree. The children of tangle 7 do not form a flyping circuit since there is no tangle in this set of children.

The reduced tree T_r may not convey all of the possible flypes in the given knot diagram D. In order to see the remaining flypes, we must redraw T_r using different interior vertices as the root of the tree (Figure 15). After considering the variations of T_r where each interior vertex of T_r is made the root of the tree, we have a complete list of the flyping circuits that are possible in the knot diagram D. In the flyping circuits shown in the tree, each index of a tangle or crossing is converted to its corresponding PD code, and a list of all possible flypes in the circuits is generated.

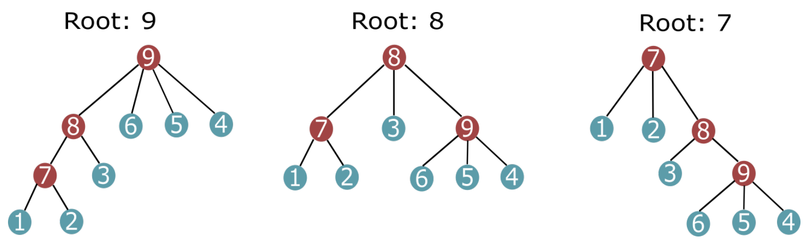

Figure 15: The variations of the reduced tree created in Figure 13. Each of the interior vertices (7, 8, 9) can become the root of the tree.

2.2 Enumeration of All Possible Flypes

In a reduced tree T_r, consider an interior vertex that has k children. Assume that x of these children are single crossings and y of these children are tangles containing more than one crossing. Thus, $x + y = k$. The k children form a flyping circuit consisting of y tangles with a cyclic order and x indistinguishable crossings. Since a flyping circuit is a cyclic structure, there are y regions created by the y tangles in which the x crossings can be distributed (Figure 16). This gives rise to at most $\binom{x+y}{x}$ different states of the flyping circuit. This can be recognized by observing that there are y possible regions which a crossing can occupy, and that the regions are separated by y dividers. Consequently, the number of available positions for the x crossings to occupy is $x+y$, which is the sum of the x number of crossings and the y dividers (Figure 16). If some of the tangles are not distinct, then there may exist fewer possible states of the flyping circuit.

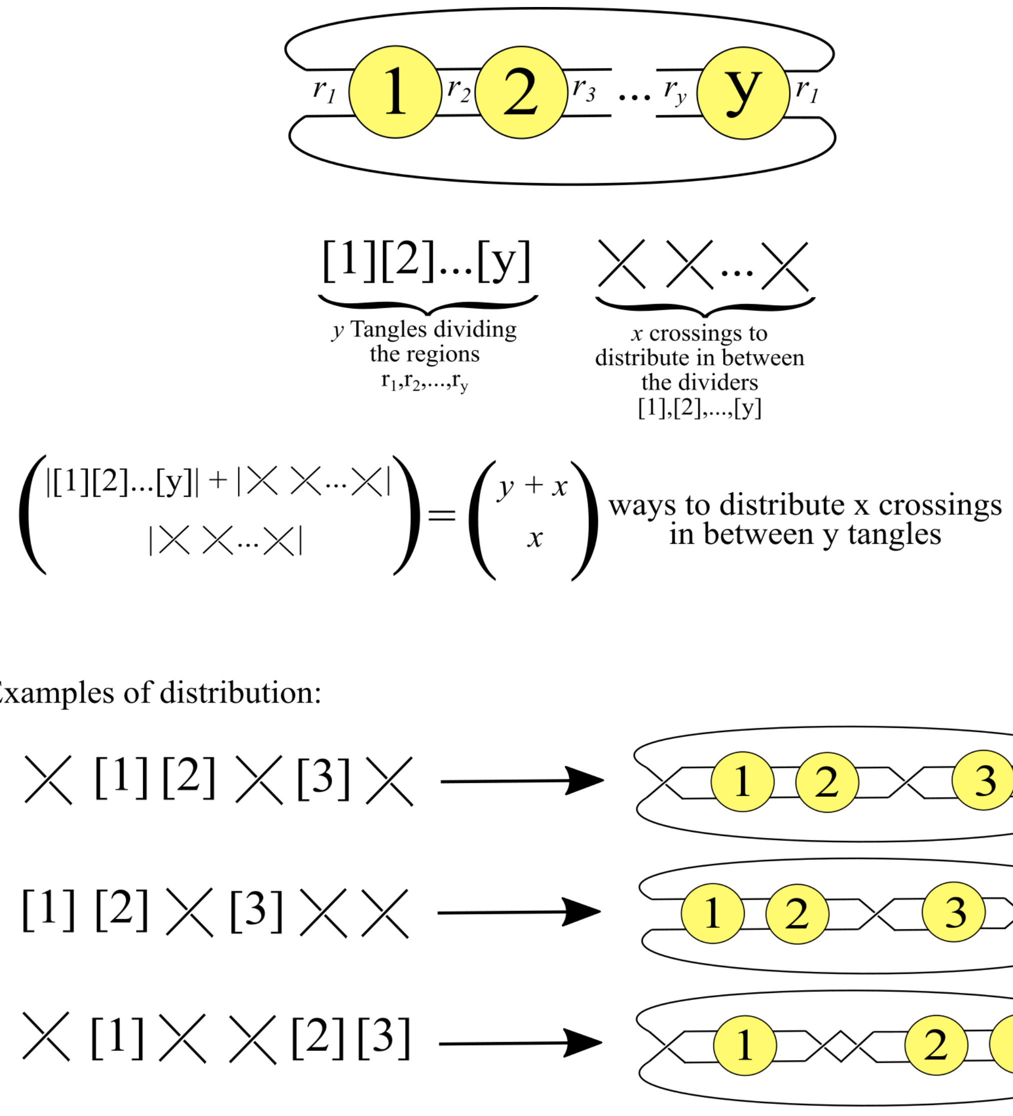
r_1 1 r_2 2 r_3 … r_y y r_1
[1][2]...[y]
y Tangles dividing the regions $r_1, r_2, \ldots, r_y$
x crossings to distribute in between the dividers [1],[2],...,[y]
ways to distribute x crossings in between y tangles
Examples of distribution:
[1] [2] [3]
1 2 3

Figure 16:

y tangles divide y regions, with the first region r_1 being the unbounded region bordering tangle 1 and tangle y.

In the distribution examples shown, note that the first example is the same as the second example. Both have one crossing in the region bounded by tangle 2 and tangle 3, and two crossings in the region bounded by tangle 1 and tangle 3.

3 Potential for Future Work

This algorithm was implemented as a program written in Wolfram Mathematica. The overall goal of this project is to have a single program in which one can input the PD code of a minimal alternating knot diagram and a list of all other minimal diagrams of the knot will be generated. Seeing that this algorithm and program detects the possible flypes which can occur in a minimal alternating knot diagram, the next step is to combine this program with one that executes flypes in a knot diagram. The result will be a single program which executes all possible flypes on a minimal alternating knot diagram and returns all minimal diagrams of the given knot. With a list of all minimal diagrams of a knot, knot invariants can be calculated. Furthermore, other quantities can be computed which can become knot invariants once the given computation is performed over all minimal diagrams of a knot. An example of such a quantity is the diagrammatic *nullification number* [4] [5] [6]. Therefore, this project will enable the computation of certain knot invariants that are defined by taking the minima or maxima of a given computation over all minimal diagrams of a given knot.

References

[1] Adams, Colin *The Knot Book*, American Mathematical Soc., (2004).

[2] Bar-Natan, Dror; Morrison, Scott; et al. *The Knot Atlas*, http://katlas.org

[3] Cromwell, Peter *Knots and Links*, Cambridge University Press, (2004).

[4] Diao, Yuanan; Ernst, Claus; Montemayor, Anthony *Nullification of knots and links*, Knot Theory Ramifications 21 (2012), no. 6, 1250046, 24 pp.

[5] Ernst, Claus; Montemayor, Anthony. *Nullification numbers of knots with up to 10 crossings*, Knot Theory Ramifications 25 (2016), no. 7, 1650037, 20 pp.

[6] Ernst, Claus; Montemayor, Anthony. *Nullification of torus knots and links*, Knot Theory Ramifications 23 (2014), no. 11, 1450058, 19 pp.

[7] Menasco, William; Thistlethwaite, Morwen. *The classification of alternating links*, Ann. of Math. (2) 138 (1993), no. 1, 113171.

[8] Menasco, William; Thistlethwaite, Morwen *The Tait flyping conjecture*, Bull. Amer. Math. Soc. (N.S.) 25 (1991), no. 2, 403412.

[9] Sundberg, Carl; Thistlethwaite, Morwen *The rate of growth of the number of prime alternating links and tangles.* Pacific J. Math. 182 (1998), no. 2, 329358.