# EXPANDING THE LEGENDRIAN KNOT ATLAS

A Thesis

Submitted to the Faculty

in partial fulfillment of the requirements for the

degree of

Bachelor of Arts

in

Mathematics

by

Noah Schwartz

DARTMOUTH COLLEGE

Advisor

Ina Petkova

Hanover, New Hampshire

May 2023

# Abstract

In this thesis, we begin working on a classification of all Legendrian knots of arc index $n = 10$. Our approach uses a combinatorial presentation of a knot, called a grid diagram. We begin by generating a set of grid diagrams guaranteed to contain at least one grid representing every non-destabilizable Legendrian knot of arc index $n = 10$. Next, we refine this set by removing destabilizable knots, finding pairs of Legendrian-isotopic knots, and sorting by various topological and Legendrian invariants. Finally, for each topological knot in our set, we compute its "mountain range," a graphical representation of the knot's Legendrian representatives, plotted in space depending on their classical invariants $tb$ and $r$.

# Acknowledgements

This project would not have been possible without the help of so many people. I'd like to start by thanking my advisor, Professor Ina Petkova, for her guidance and encouragement throughout this process. I would also like to thank the Dartmouth College Math Department for giving me the opportunity and resources to study math these past four years. Next, I want to acknowledge Mitchell Jubeir for his help in developing a grid generation algorithm, and Chuck Livingston for his help in topological identification of grids. Last and certainly not least, I would like to thank my friends and family. In particular, I would like to thank Ella Marden, Nat Alden, Emma Suppatapone, and Ian Gill, who have listened to me endlessly ramble about this project for the last nine months. Finally, I want to thank my parents, who have always been there for me. I will be forever grateful for your love and support.

# Contents

Chapter 1

# Introduction

This thesis tackles one of the simplest questions in knot theory: What knots are out there? Unsurprisingly, this question is (way) too broad to be answered by a single undergraduate thesis. Therefore, the specific goal of this thesis is to catalogue every non-destabilizable Legendrian knot of arc index 10. By the end of this introduction, I hope that readers will begin to understand what this means and why we care.

In a topological setting, knots are exactly what you might imagine: starting with a length of string, tangle it up in any way you want, and then fuse the ends. Two knots are 'isotopic' if you can stretch, twist, or otherwise deform the first knot into the exact shape of the second. Since there is no limit to the amount of tangling, there are infinitely many knots even in a purely topological setting. Therefore, to restrict the scope of this thesis, we restricted ourselves to a certain class of 'small' knots. Specifically, we focused on knots of arc index 10, meaning the set of knots that can be represented by 2-dimensional pictures composed of vertical and horizontal strands arranged on a $10 \times 10$ grid.

Now, the set of topological knots of arc index 10 has already been tabulated; see for example [JP10]. All but one have 16 of fewer crossings, and are catalogued on KnotInfo [LM23]. However, the same cannot be said for the set of *Legendrian*

knots of arc index 10. A Legendrian knot is a knot that satisfies certain geometric constraints from contact topology. Two Legendrian knots are 'Legendrian isotopic' if they can be twisted and deformed into one another, while always satisfying the same geometric constraints. Physically, this restricts the ways in which you are allowed to manipulate the knots. As a result, two Legendrian knots might be topologically isotopic but not Legendrian isotopic! In fact, every topological knot has infinitely many distinct Legendrian representatives. To handle this issue, we focused only on non-destabilizable Legendrian knots, which function as a generating set from which any Legendrian representative can be easily built. In [CN13], Chongchitmate and Ng and catalogued all non-destabilizable knots of arc index 9 or fewer; this thesis extends their work to arc index 10.

As mentioned above, the problem of classification of Legendrian knots is hard. Only a few special families of Legendrian knots have been fully classified. The number of distinct knots increases rapidly with arc index, and searching for isotopies between these knots requires exponentially-growing memory and time. As a result, so far only a few small knots not in the special families mentioned above are well-understood. This thesis presents a conjectural classification of all 604 distinct topological knots of arc index 10, taking mirroring and orientation into account. These results are a major step towards a complete classification of Legendrian knots of low crossing number. We hope that these results will be useful to mathematicians working on conjectures who are looking for examples.

The rest of this thesis is organized as follows. Chapter 2 provides a crash course in the mathematical background needed to understand this thesis. Chapter 3 details the algorithms and strategies employed to generate a minimal set of grids containing at least one destabilizable Legendrian knot of arc index 10. Chapter 4 explains how we computed successive layers of the mountain range for each topological knot type.

Finally, Chapter 5 provides a big-picture overview of results, and takes a deep-dive into the results for a single topological type.

Thank you for reading all the way to the end of the introduction, and enjoy the rest of the paper!

Chapter 2

# Background

In this chapter, we introduce the mathematical background that underlies this project. The first section of this chapter discusses the basics of knot theory in a topological setting, and the second section discusses the basics of knot theory in a Legendrian setting.

Section 2.1

# Topological Knot Theory

**Definition 2.1.1.** A knot $K$ is a smooth embedding of the closed loop $S^1$ into another manifold, typically $\mathbb{R}^3$ or $S^3$.

Two knots $K_1$ and $K_2$ are equivalent if they are ambiently isotopic, which formalizes the notion of smoothly manipulating the knot strands without allowing them to cross one another. Isotopy equivalence classes under this relation are called knot types. Some familiar knot types include the unknot, the trefoil knot, and the figure-8 knot.

Mathematicians have developed countless tools to describe and characterize knots. In this section, we will discuss the relevant topics of knot theory in a topological setting. We will begin by describing knot diagrams and the Reidemeister moves. Next, we will discuss grid diagrams and the grid moves. We will then finish by defining the Alexander polynomial.

## Knot Diagrams

A *knot diagram* $\mathcal{D}$ is a projection of a knot in $\mathbb{R}^3$ onto a two-dimensional plane, such that three line segments never intersect at a single point. We write the knot $K$ described by diagram $\mathcal{D}$ as $K(\mathcal{D})$. To determine when different diagrams represent the same knot, we introduce the three Reidemeister moves, illustrated in Figure 2.1. Two knot diagrams $\mathcal{D}_1$ and $\mathcal{D}_2$ represent topologically isotopic knots if and only if $\mathcal{D}_1$ may be transformed into $\mathcal{D}_2$ via a sequence of Reidemeister moves and planar isotopies.

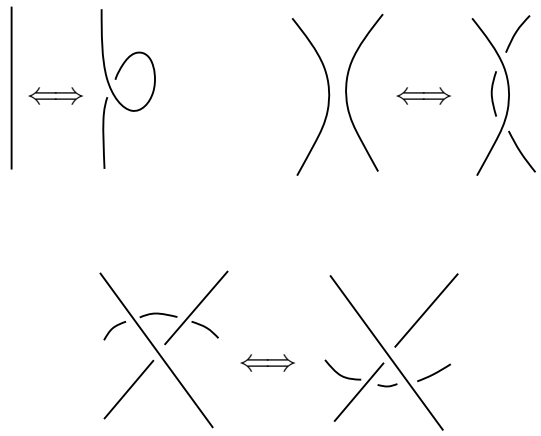

Figure 2.1: The top-left diagram illustrates a Type I Reidemeister move. The top-right diagram illustrates a Type II Reidemeister move. The bottom-center diagram illustrates a Type III Reidemeister move. Note that these moves do not depend on strand orientation. Also note that reflected and/or rotated versions of these moves are allowed as well.

### Grid Diagrams

To make it easy for computers to work with knot diagrams, we want to combinatorially encode the information stored in a diagram $\mathcal{D}$. To do so, we introduce grid diagrams.

**Definition 2.1.2.** A *grid diagram* $\mathbb{G}$ is an $n \times n$ grid of squares on the plane $\mathbb{R}^2$, where $n$ of these squares are marked with an $X$ and $n$ are marked with an $O$. Furthermore, these markings must satisfy the following three properties:

- There must be exactly one square marked with an $X$ and one square marked with an $O$ in every column of squares.
- There must be exactly one square marked with an $X$ and one square marked with an $O$ in every row of squares.
- No square may be marked with both an $X$ and an $O$.

Any grid diagram that satisfies these three properties may be compactly described by a pair of permutations $(\mathbb{X}, \mathbb{O}) \in (S_n)^2$. These permutations store the heights of

the $X$- and $O$-markings; that is, the square in column $i$, row $\mathbb{X}_i$[1] is marked with an $X$ (where columns are counted from the left, and rows are counted from the bottom). Since it is so convenient to describe a grid diagram $\mathbb{G}$ with these permutations, we often say that $\mathbb{G} = (\mathbb{X}, \mathbb{O})$.

We can build a knot diagram from a grid diagram as follows. First, in each row, draw an oriented horizontal line from the $O$ to the $X$.[2] Next, for each column, draw an oriented vertical line from the $X$ to the $O$. Finally, at each crossing, let the vertical line pass over the horizontal line. Thus, a grid diagram $\mathbb{G}$ specifies a knot diagram $\mathcal{D}$, which in turn specifies a knot $K$; thus, we may speak of $K(\mathbb{G})$, the knot described by grid diagram $\mathbb{G}$. Figure 2.2 illustrates this process for a $5 \times 5$ grid diagram describing the left-handed trefoil.

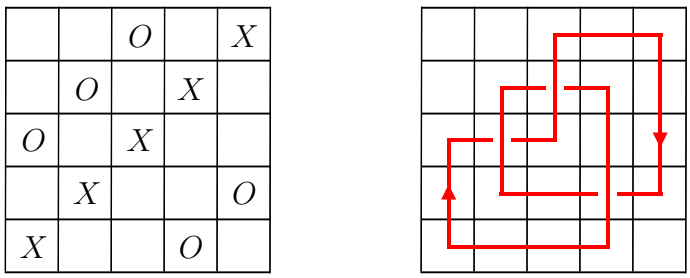


Figure 2.2: Left: The grid diagram described by permutations $\mathbb{X} = (0, 1, 2, 3, 4), \mathbb{O} = (2, 3, 4, 0, 1)$. Right: The oriented knot diagram generated by the grid diagram at left.

To determine when different grid diagrams represent the same knot, we introduce the grid moves, in analogy to the Reidemeister moves. The types of allowed moves are row and column commutations (illustrated in Figure 2.3), and $X$-(de)stabilizations (illustrated Figure 2.4). Row commutations swap two adjacent rows, and are allowed only when when the horiztonal strands in the two rows span either disjoin or nested intervals. Column commutations are defined similarly. Stabilization moves transform

[1] By convention, for any permutation $P \in S_n$, we let $P_i$ denote $P(i)$. Furthermore, we take indices modulo $n$, so that $P_m$ denotes $P(m \mod n)$.

[2] Note that we use the phrases '$X$-markings' and '$X$s' interchangeably.

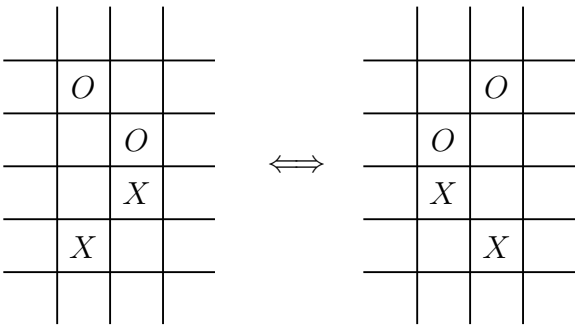


Figure 2.3: Left: A local picture of a grid diagram, showing $X$- and $O$-markings in the squares in columns $i$ and $i+1$. Right: A local picture of the same grid diagram after a column commutation of columns $i$ and $i+1$ has been performed. Note that this column commutation is allowed, because the vertical strands these columns span nested intervals.

a single $X$-marking into a three markings, increasing the grid size by one. There are four different stabilization moves, distinguished by the way in which the three new markings are created. Finally, destabilization moves are the inverse of stabilization moves, transforming three markings into one and decreasing the grid size by one. Two grid diagrams $\mathbb{G}_1$ and $\mathbb{G}_2$ represent topologically isotopic knots if and only $\mathbb{G}_1$ may be transformed into $\mathbb{G}_2$ via a sequence of grid moves.

### Topological Invariants

To determine when two knot diagrams represent the same knot, it suffices to find a sequence of Reidemeister moves connecting the two diagrams. On the other hand, to determine when two knot diagrams represent *distinct* knots, we must introduce the concept of knot invariants. Knot invariants are typically some sort of computable property of a grid diagram – such as a number or a polynomial – that do not change under the Reidemeister moves. Thus, these invariants are properties of not only the knot diagram, but of the underlying topological knot type itself.

Therefore, these invariants can be used to test whether the knots represented by two diagrams are distinct; if two diagrams $\mathcal{D}_1$ and $\mathcal{D}_2$ are found to have different

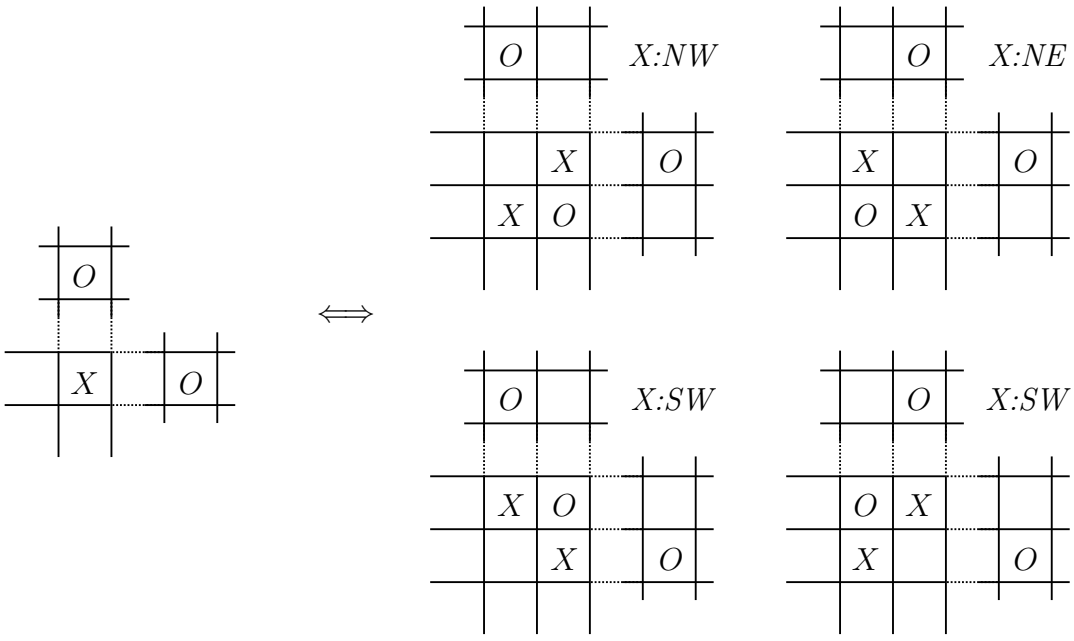


Figure 2.4: **Stabilization and destabilization moves.** Left: A local picture of a grid diagram, showing the $X$- and $O$-markings in column $c$ and row $r$. Right: Four local pictures of the same grid diagram, after performing the four different stabilization moves at the $X$-marking at column $c$, row $r$. These pictures show the $X$- and $O$-markings in columns $c, c+1$ and rows $r, r+1$.

values for a certain invariant, then $K(\mathcal{D}_1) \neq K(\mathcal{D}_2)$.

One well-known knot invariant is the (symmetrized) Alexander polynomial, which we used in this project to rule out possible isotopies between grid diagrams. To define the Alexander polynomial, we must first introduce the notion of skein relations. A triple of oriented knot diagrams $(L_-, L_0, L_+)$ satisfies a skein relation if the three diagrams only differ locally, as in Figure 2.5. At that location, $L_-$ has a negatively-oriented crossing, $L_+$ has a positively-oriented crossing, and $L_0$ has no crossing at all.

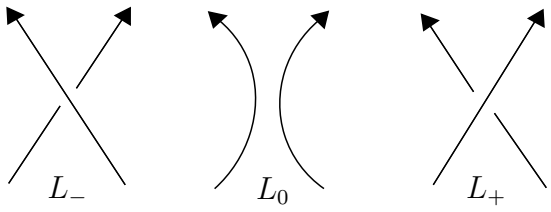


Figure 2.5: Three diagrams corresponding to the local pieces of the knot diagrams of $L_-$, $L_0$, and $L_+$, centered at the location about which all three diagrams differ. Note that these diagrams may specify multiple-component links or single-component knots.

We now define the Alexander polynomial of a link $L$ (denoted $\Delta_L(t)$) by the following three rules. First, the Alexander polynomial of any diagram of the unknot is defined to be one. Second, the Alexander polynomial of any diagram of the unlink (two disconnected unknots) is defined to be zero. Third, for any skein triple $(L_-, L_0, L_+)$, we have:

$$\Delta_{L_-}(t) - \Delta_{L_+}(t) = (t^{-\frac{1}{2}} - t^{\frac{1}{2}})\Delta_{L_0}(t)$$

Iteratively applying these rules allows us to compute the uniquely defined symmetrized Alexander polynomial of any link.

Section 2.2

## Legendrian Knot Theory

Knots in a purely topological setting are interesting in their own right. However, the field becomes far richer if we impose additional structure on our knots. To do so, we turn to the notion of contact structure from the field of contact topology. A contact structure on a 3-manifold $M$ is a 2-plane field $\xi \subset TM$. This thesis does not discuss the basics of contact topology in a general setting. From here on, we only consider the standard contact structure on $\mathbb{R}^3$,

$$\xi_{\text{std}} = \ker(\alpha_{\text{std}}), \quad \alpha_{\text{std}} = dz - y\,dx.$$

We may visualize this contact structure as a plane at each point in $\mathbb{R}^3$, where the plane at each point is perpendicular to the vector $\langle -y, 0, 1\rangle$. Using this contact structure, we may now impose geometric constraints on the shape of a knot. This leads us to the definition of Legendrian knots.

**Definition 2.2.1.** A *Legendrian knot* $\Lambda$ in $\mathbb{R}^3$ is a knot that is everywhere tangent to the standard contact structure on $\mathbb{R}^3$.

We say that two Legendrian knots $\Lambda_1$ and $\Lambda_2$ are *Legendrian* isotopic if they are isotopic through a family of Legendrian knots, and define Legendrian knot types as the equivalence classes under this relation. The universe of Legendrian knot types is much less well understood than that of topological knot types.

In this section, we will introduce Legendrian knot theory, selecting topics that are relevant to this thesis. We will begin by defining front projections and the Legendrian Reidemeister moves. Next, we will return to grid diagrams, and show that they can be used to specify Legendrian knots. Then, we will define the classical Legendrian invari-

ants, the Thurston-Benneqin and rotation numbers, and discuss how these invariants can be used to organize the universe of Legendrian knot types. We will conclude with a discussion of stabilizations and symmetry transformations of Legendrian knots.

### Front Projections

A *front projection* $\mathcal{P}$ is a projection of a Legendrian knot in $\mathbb{R}^3$ onto the $xz$-plane, such that the $y$-axis points into the plane of the page. From every front projection $\mathcal{P}$, we may uniquely recover a Legendrian knot $\Lambda(\mathcal{P})$ as follows: Parameterize $\mathcal{P}$ as $(x(t), z(t))$ for $t \in [0, 1]$. Then, let $\Lambda(\mathcal{P})$ be the knot in $\mathbb{R}^3$ parameterized by $(x(t), \frac{dz}{dx}(t), z(t))$. In other words, the slope of the strand of the front projection encodes the $y$-coordinate of $\Lambda(\mathcal{P})$. Note that this parametrization guarantees that $\Lambda(\mathcal{P})$ is everywhere tangent to $\xi_{\text{std}}$, so it is indeed Legendrian. In this way, the slopes of the strands encode crossing information: strands with a more negative slope pass over strands with a less negative one. Finally, since $y = \frac{dz}{dx}$, front projections cannot have vertical tangencies, and must have cusps instead.

To determine when two front projections represent the same Legendrian knot, we introduce the Legendrian Reidemeister moves, illustrated in Figure 2.6. Note that the Legendrian Reidemeister moves are more restrictive than the topological Reidemeister moves. This follows from the requirement that Legendrian knots be isotopic through a family of Legendrian knots, which limits on the ways in which we can manipulate the knots in $\mathbb{R}^3$. Two front projections $\mathcal{P}_1$ and $\mathcal{P}_2$ represent Legendrian isotopic grids if and only if $\mathcal{P}_1$ may be transformed into $\mathcal{P}_2$ via a sequence of Legendrian Reidemeister moves and cusp-preserving planar isotopies.

### Grid Diagrams, Revisited

Grid diagrams can encode the information stored in a front projection. We may obtain a front projection from a grid diagram as follows. First, for each row, draw

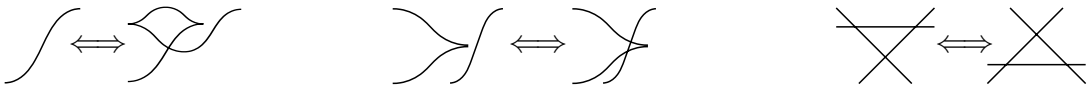

Figure 2.6: Left: A Type I Legendrian Reidemeister mvoe; it also has a variant that may be obtained by rotating the diagram by 180 degrees. Center: A Type II Legendrian Reidemeister move; it has three additional variants, obtained by horizontal and vertical reflections. Right: A Type III Legendrian Reidemeister move.

an oriented horizontal line from the $O$ to the $X$. Next, for each column, draw an oriented vertical line from the $X$ to the $O$. Next, rotate the entire grid diagram 45 degrees clockwise. Finally, smooth corners into horizontal tangencies where possible, and into cusps otherwise. In this way, a grid diagram $\mathbb{G}$ specifies a front projection $\mathcal{P}$, which in turn specifies a Legendrian knot $\Lambda$. Therefore, we may speak of $\Lambda(\mathbb{G})$, the Legendrian knot described by grid diagram $\mathbb{G}$. For an example of this process, see Figure 2.7. Note that during this process, crossings that were positive in the knot diagram $\mathcal{D}(\mathbb{G})$ became negative in the front projection $\mathcal{P}(\mathbb{G})$. Therefore, the *topological* knot described by the front projection $\mathcal{P}(\mathbb{G})$ is equal to the *mirror* of the topological knot described by the grid diagram $\mathcal{D}(\mathbb{G})$.

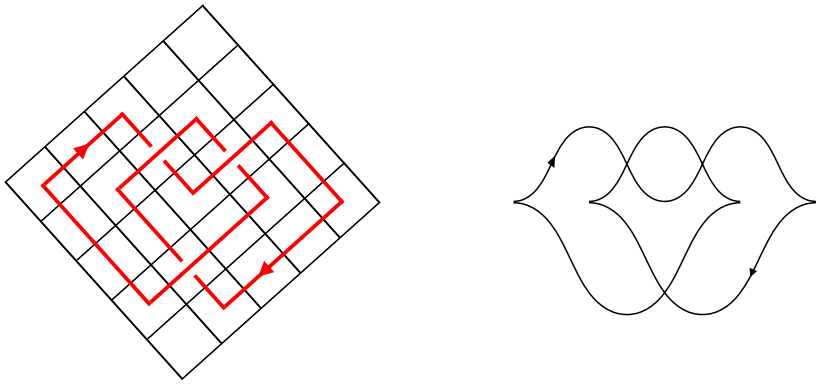

Figure 2.7: Left: The knot diagram generated by a grid diagram $\mathbb{G}$. Right: The front projection generated by the same grid diagram. Note that in front projections, negatively-sloped strands pass over positively-sloped strands. Therefore each of the crossings is flipped when moving from the knot diagram to the front projection.

The following theorem identifies the set of moves that may be performed on a grid diagram $\mathbb{G}$ without changing $\Lambda(\mathbb{G})$. Just as the Legendrian Reidemeister moves are a

subset of the topological Reidemeister moves, the allowed moves here are a subset of the topological-type-preserving grid moves identified in Section 2.1. Note that cyclic column permutations are defined as moving the leftmost column to the far right of the grid diagram (or vice versa), and cyclic row permutations are defined identically.

**Theorem 2.2.2.** *Two grid diagrams $\mathbb{G}_1$ and $\mathbb{G}_2$ represent Legendrian isotopic knots if and only if $\mathbb{G}_1$ may be transformed into $\mathbb{G}_2$ via a sequence of row and column commutations, X:NW, X:SE, O:NW, O:SE (de)stabilizations, and cyclic row and column permutations.*

*Proof.* This theorem summarizes results from [OSS15, Section 12.2]. □

### Legendrian Knot Invariants

To determine when two front projections represent different Legendrian knots, we introduce Legendrian knot invariants. The two 'classical' Legendrian invariants are the Thurston-Benneqin number $tb$ and the rotation number $r$.

**Definition 2.2.3.** Given an oriented front projection $\mathcal{P}$, the Thurston-Benneqin number $tb$ is defined as

$$tb(\mathcal{P}) \equiv wr(\mathcal{P}) - \frac{1}{2}\#(\text{cusps}),$$

where the writhe $wr$ is the number of positive crossings minus the number of negative crossings. See Figure 2.5 for the definitions of positive and negative crossings. The rotation number is defined as

$$r(\mathcal{P}) \equiv \frac{1}{2}\big(\#(\text{downward cusps}) - \#(\text{upward cusps})\big)$$

It it easy to show that these numbers are invariant under the Legendrian Redemeister moves. Therefore, given a Legendrian knot $\Lambda$, we may speak of $tb(\Lambda)$ and $r(\Lambda)$ as Legendrian knot invariants.

Using these invariants, it is easy to show that the same topological knot may have multiple Legendrian representatives. For example, the two front projections in Figure 2.8 both topologically correspond to the unknot. However, they have different values of $tb$ and $r$, and therefore represent distinct Legendrian knots.

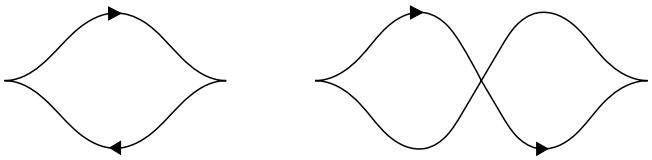

Figure 2.8: Left: A front projection of an unknot with $tb = -1$ and $r = 0$. Right: A front projection of an unknot with $tb = -2$ and $r = -1$.

Distinct Legendrian representatives of the same topological knot are often related to one another by positive or negative Legendrian stabilizations, as defined in [NOT08]. Given a Legendrian knot $\Lambda$, we obtain its positive stabilization $S_+(\Lambda)$ by performing an *X:NE* or *O:SW* stabilization anywhere on any grid diagram $\mathbb{G}$ that represents $\Lambda$, and considering the knot represented by the resulting grid diagram [OSS15, Propsition 12.2.7]. The negative stabilization $S_-(\Lambda)$ is defined similarly, except with *X:SW* or *O:NE* stabilizations instead. These positive and negative stabilizations correspond to local transformations of a front projection, as seen in Figure 2.9.

Note that these stabilizations are valid grid moves in a topological setting, meaning that they always preserve topological type. However, they never preserve Legendrian type, because they affect the classical Legendrian invariants. In general,

$$tb(S_\pm(\Lambda)) = tb(\Lambda) - 1, \quad r(S_\pm(\Lambda)) = r(\Lambda) \pm 1.$$

These stabilizations allow us to organize the Legendrian representatives of a topological knot type $K$ into a single diagram, known as a mountain range. For an example of a simple mountain range, see Figure 2.10; this diagram, reproduced from [CN13], illustrates the mountain range of the right-handed trefoil. Although conventions dif-

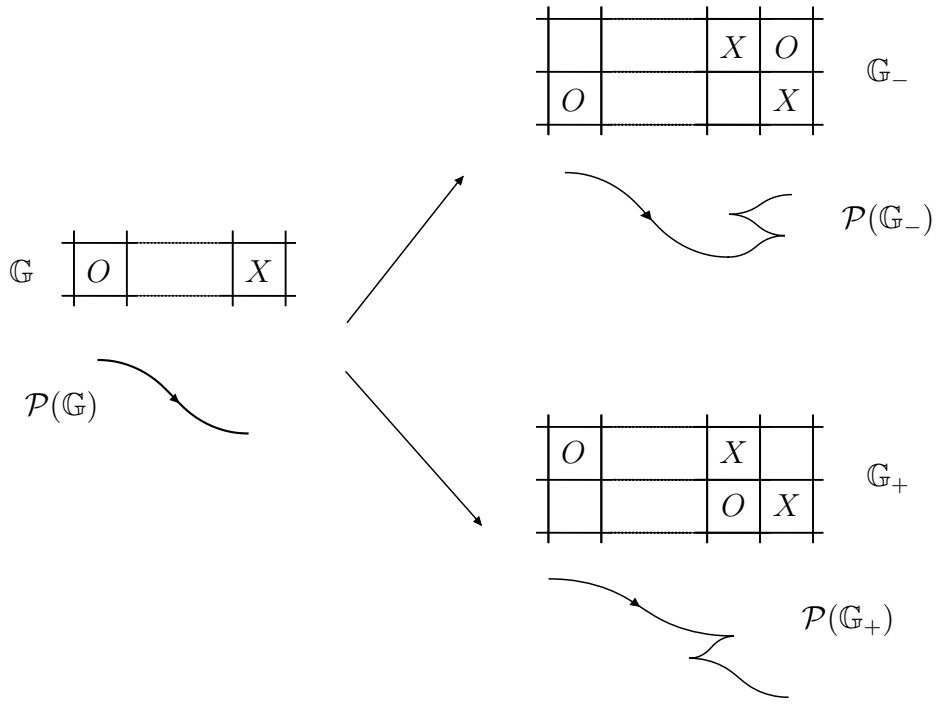


Figure 2.9: **Positive and negative stabilizations of a Legendrian knot.** Left: A local picture of a grid $\mathbb{G}$, showing the $X$- and $O$-markings in a single row. Immediately below it, the the corresponding local piece of the front projection $\mathcal{P}(\mathbb{G})$. Top right: A local picture of a grid $\mathbb{G}_-$, obtained by performing an *X:SW* stabilization on $\mathbb{G}$ at the pictured $X$-marking. Immediately below it, the corresponding local piece of the front projection $\mathcal{P}(\mathbb{G}_-)$. Observe that $\Lambda(\mathbb{G}_-) = S_-(\Lambda(\mathbb{G}))$. Bottom right: A local picture of a grid $\mathbb{G}_+$, obtained by performing an *X:NE* stabilization on $\mathbb{G}$ at the pictured $X$-marking. Immediately below it, the corresponding local piece of the front projection $\mathcal{P}(\mathbb{G}_+)$. Observe that $\Lambda(\mathbb{G}_+) = S_+(\Lambda(\mathbb{G}))$.

fer from author to author, these diagrams are typically interpreted as follows. Each individual dot on the diagram corresponds to a unique Legendrian representative of $K$. The $x$-coordinate of a dot corresponds to the rotation number of that Legendrian representative, and the $y$-coordinate corresponds to the Thurston-Benneqin number. An arrow from representative $\Lambda_1$ to representative $\Lambda_2$ indicates that $\Lambda_2$ is the positive or negative stabilization of $\Lambda_1$. Note that there may be multiple Legendrian representatives with identical $tb$ and $r$, and the stabilizations of these knots may be the same or different.

Observe that these mountain ranges contain a number of 'peaks', points which

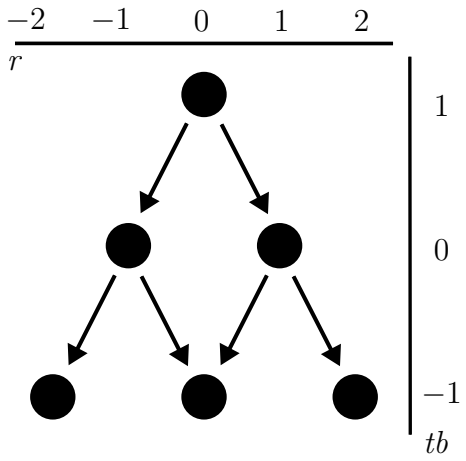


Figure 2.10: The topmost portion of a mountain range for the right-handed trefoil, labeled $3_1$ in the Rolfsen Knot Table. This mountain range tells us that $K$ has a single destabilizable Legendrian representative, with $tb = -7$ and $r = 0$. Positive and negative stabilizations of this knot yield additional Legendrian representatives. Notice that there is either one or zero Legendrian representative of $K$ for each combination of $tb$ and $r$; this property is not true in general.

have no arrows pointing to them. These peaks may occur at any $tb, r$, and can even be in the interior of the range. By definition, every Legendrian representative of some topological type $K$ can be found by performing some sequence of positive or negative stabilizations to one of these peaks. Therefore, we may reconstruct the entire mountain range starting only from the peaks. We formalize the notion of these peaks with the following definition.

**Definition 2.2.4.** A Legendrian knot $\Lambda$ is *non-destabilizable* if it cannot be transversely destabilized, which implies that it is not the positive or negative stabilization of another Legendrian knot. The *grid size* $n$ of a Legendrian knot $\Lambda$ is the smallest such $n$ such that $\Lambda$ may be represented by a size-$n$ grid diagram, and is possibly distinct from the arc index of the underyling topological type.

Any Legendrian knot can be constructed by performing a sequence of positive and/or negative stabilizations on some non-destabilizable Legendrian knot. Furthermore, given two Legendrian representatives $\Lambda_1$ and $\Lambda_2$ of the same topological knot, it is always possible to perform some sequence of positive and/or negative stabilizations

on $\Lambda_1$, then peform another sequence on $\Lambda_2$, and obtain Legendrian-isotopic knots.

### Legendrian Symmetry Types

We now consider the behavior of Legendrian knots under a pair of important symmetry transformations.

First, consider transforming a Legendrian knot $\Lambda$ into its orientation reversal $-\Lambda$. On a grid diagram, this transformation is carried out by replacing $X$s with $O$s, and vice versa. Since this transformation preserves the crossing orientations and the number of cusps, we have $tb(\Lambda) = tb(-\Lambda)$. However, since the transformation switches upward and downward cusps, we know that $r(\Lambda) = -r(-\Lambda)$. Therefore, it is possible for a Legendrian knot to equal its orientation reversal only if $r(\Lambda) = 0$. Furthermore, notice that it is possible that taking the orientation reversal of a knot changes its topological type. The number of these 'non-invertible' knots grows rapidly with increasing grid size, and constitute about half of the knots considered in this project.[3] The important takeaway is that topological properties as well as the classical Legendrian invariants can obstruct isotopies between $\Lambda$ and $-\Lambda$.

Next, consider transforming a Legendrian knot into its 'Legendrian mirror' $\mu(\Lambda)$, obtained by rotating $\Lambda$ about the $y$-axis by 180 degrees. On a grid diagram, this transformation is simply carried out by rotating the entire diagram 180 degrees. Just like orientation reversals, this transformation preserves crossing orientations and the number of cusps, but flips upward and downward cusps. Therefore, $tb(\Lambda) = tb(\mu(\Lambda))$ and $r(\Lambda) = -r(\mu(\Lambda))$, and so it is possible for a Legendrian knot to equal its Legendrian mirror only if $r(\Lambda) = 0$. Just like with orientation reversals, it is possible for the classical Legendrian invariants to obstruct isotopies between $\Lambda$ and $\mu(\Lambda)$. For-

[3] The behavior of an oriented topological knot type under the operations of orientation reversal and mirroring specifies its *symmetry type*. There are five possibly knot symmetry types: invertible, fully chiral, fully amphicheiral, positive amphicheiral, and negative amphicheiral. For more on this, see [sym23].

tunately, we never have to worry about topological properties, since rotating a knot always preserves its topological type.

Finally, consider combining an orientation reversal and a Legendrian mirror to take $-\mu(\Lambda)$ (which is clearly equal to $\mu(-\Lambda)$). Notice that under the composition of these operations, the rotation number does not change, so the classical invariants will never obstruct isotopies between $\Lambda$ and $-\mu(\Lambda)$ for any Legendrian knot $\Lambda$. However, topological properties can still obstruct these isotopies; it is impossible for a Legendrian representative of a non-invertible knot to equal the orientation reversal of its own Legendrian mirror.

We conclude this section by considering how these symmetry transformations relate to positive and negative stabilizations. Notice that taking an orientation reversal of a grid transforms an *X:NE* stabilization pattern to an *O:NE* stabilization pattern, and similarly changes the other stabilization patterns. Therefore, positively stabilizing a grid and then taking the orientation reversal should be equivalent to taking the orientation reversal and then *negatively* stabilizing. Similarly, taking the Legendrian mirror of a grid transforms *X:NE* stabilizations into *X:SW* stabilizations, and taking the orientation reversal of the Legendrian mirror of a grid transforms *X:NE* stabilizations into *O:SW* stabilizations patterns. Following this train of logic, we arrive at the following theorem.

**Theorem 2.2.5.** *The following equalities hold for any Legendrian knot $\Lambda$:*

$$S_+(-\Lambda) = -S_-(\Lambda) \qquad S_-(-\Lambda) = -S_+(\Lambda)$$

$$S_+(\mu(\Lambda)) = \mu(S_-(\Lambda)) \qquad S_-(\mu(\Lambda)) = \mu(S_+(\Lambda))$$

$$S_+(-\mu(\Lambda)) = -\mu(S_+(\Lambda)) \qquad S_-(-\mu(\Lambda)) = -\mu(S_-(\Lambda))$$

*Proof.* This theorem follows from studying the ways in which positive and negative stabilizations, orientation reversals, and Legendrian mirrors are realized on a grid diagram. □

Chapter 3

# Generating the Set of Grids

In this section, we will describe the algorithms and computational techniques used to generate a small set of grids containing every non-destabilizable Legendrian knot of arc index 10. This chapter will be organized into three sections. The first section will explain how we generated a large set of grids guaranteed to contain every non-destabilizable Legendrian knot of arc index 10. The second will explain how we shrunk that set of grids. The third will explain how we organized that set, to improve the memory and runtime efficiency of our algorithms.

Section 3.1

# Generating Grids

As discussed in the introduction to this thesis, every Legendrian knot of arc index 10 can be described by a size-10 grid diagram, which can in turn be described by two permutations in $S_{10}$. Therefore, there is an upper bound of $|S_{10}|^2 = (10!)^2 \approx 10^{14}$ possible distinct Legendrian knots of arc index 10, corresponding to every possible pair of permutations. However, with a more sophisticated generation algorithm, we can produce a candidate set much smaller than this upper bound. To achieve this goal, we kept four principles in mind while designing a generation algorithm:

(a) We only want to generate valid grid diagrams, that is grids that do not have $X$- and $O$-markings in the same grid square.

(b) We only want to generate grid diagrams corresponding to knots, and not links.

(c) We want to avoid generating two grid diagrams which are equivalent to one another via cyclic row and column permutations, since these correspond to the same Legendrian knot.

(d) We want to avoid generating grids which are immediately destabilizable.

Eventually, we settled on a three-step generation algorithm that kept these principles in mind. First, we generated a list of $\mathbb{X}$-permutations, none of which were equivalent to one another via a cyclic permutation. Second, for each $\mathbb{X}$-permutation, we found all $\mathbb{O}$-permutations corresponding to valid diagrams that were not immediately destabilizable. Finally, for each grid $\mathbb{G} = (\mathbb{X}, \mathbb{O})$, we verified that it represented a knot and not a multiple-component link. By implementing this algorithm, we obtained a set of approximately 89 million grids, significantly fewer than the upper bound of $10^{14}$.

### Listing $X$-Permutations

The first way we kept our set small was by finding a small subset of permutation $\Sigma \subset S_{10}$, such that we only needed to consider grids of the form $(\mathbb{X}, \mathbb{O})$ with $\mathbb{X} \in \Sigma$. To find this small set $\Sigma$, we implemented the following algorithm. [1]

**Algorithm 3.1.1.** Generate X-Permutations

On any input:

1. Let $\Sigma$ be the set of all permutations $\mathbb{X} \in S_{10}$ such that $\mathbb{X}_0 = 0$.
2. For each $\mathbb{X} \in \Sigma$:
   a. Consider the grid $\mathbb{G}_{\mathbb{X}}$ described by $\mathbb{X}$ created by marking the square $(i, \mathbb{X}_i)$ with an $\mathbb{X}$ for each $i \in [10]$.
   b. For each column $col \in [n]$:
      i. Define $row = \mathbb{X}_{col}$, then construct the following permutation:

$$\mathbb{X}'_i = (\mathbb{X}_{i+col} - row) \mod 10$$

         Remember that were are taking indices mod 10. Note that this permutation describes a grid $\mathbb{G}_{\mathbb{X}'}$, obtained by cyclically permuting the rows and columns of $\mathbb{G}_{\mathbb{X}}$ so that the $X$-marking at square $(col, \mathbb{X}_{col})$ is moved to the bottom-left corner.
      ii. If $\mathbb{X}' \neq \mathbb{X}$ and $\mathbb{X}' \in \Sigma$, remove $\mathbb{X}'$ from $\Sigma$.
3. Return $\Sigma$.

In plainer English, this algorithm generates a small subset of permutations $\Sigma \subset S_{10}$, and then further shrink it by removing permutations corresponding to grids that

[1] Throughout this algorithm and the rest of this thesis, $[n]$ shall denote the set $\{0, 1, ..., n-1\}$.

are related by cyclic row and column permutations. We now show that we can build any Legendrian knot of arc index 10 out of an $\mathbb{X}$-permutation in this set.

**Theorem 3.1.2.** *Let $\Sigma \subset S_{10}$ be the set of permutations returned by Algorithm 3.1.1. Every Legendrian knot $\Lambda$ of Legendrian grid size 10 can be represented by a grid diagram $\mathbb{G} = (\mathbb{X}, \mathbb{O})$, where $\mathbb{X} \in \Sigma$ and $\mathbb{O} \in S_{10}$.*

To prove this theorem, we must first introduce some notation, and then prove a pair of lemmas. Let $\Lambda(\mathbb{X})$ denote the set of Legendrian knots described by planar grid diagrams of the form $\mathbb{G} = (\mathbb{X}, \mathbb{O})$. Then, given a set of permutations $\Sigma$, let $\Lambda(\Sigma) = \bigcup_{\mathbb{X} \in \Sigma} \Lambda(\mathbb{X})$. Next, consider the following lemmas:

**Lemma 3.1.3.** *Every Legendrian knot of arc index 10 can be represented by a planar grid diagram with an $X$ in the bottom left corner.*

*Proof.* Let $\Lambda$ be any arc-index 10 Legendrian knot. Choose an arbitrary size-10 planar grid diagram representing $\Lambda$, denoted $\mathbb{G} = (\mathbb{X}, \mathbb{O})$. Next, move the bottom row of $\mathbb{G}$ to the top $\mathbb{X}_0$ times, so that the resulting grid $\mathbb{G}'$ has an $X$ in the bottom left corner. By Theorem 2.2.2, $\Lambda(\mathbb{G}) = \Lambda(\mathbb{G}') = \Lambda$. Thus, $\Lambda$ is represented by the planar grid diagram $\mathbb{G}'$ which has an $X$ in the bottom left corner. □

**Lemma 3.1.4.** *Let $\mathbb{X}$ and $\mathbb{X}'$ be permutations in $S_{10}$, and let $\mathbb{G}_{\mathbb{X}}$ ($\mathbb{G}_{\mathbb{X}'}$) be the grid diagram obtained by marking the grid square in column $i$, row $\mathbb{X}_i$ ($\mathbb{X}'_i$) with an $X$. Suppose $\mathbb{G}_{\mathbb{X}}$ and $\mathbb{G}_{\mathbb{X}'}$ are related by a sequence of cyclic row and column permutations. Then $\Lambda(\mathbb{X}) = \Lambda(\mathbb{X}')$.*

*Proof.* Let $\Lambda$ be an arbitrary Legendrian knot in $\Lambda(\mathbb{X})$, and choose a planar grid diagram of the form $\mathbb{G} = (\mathbb{X}, \mathbb{O})$ that represents $\Lambda$. Now, transform $\mathbb{G}$ by performing the same sequence of cyclic row and column permutations that relate $\mathbb{G}_{\mathbb{X}}$ and $\mathbb{G}_{\mathbb{X}'}$. This will transform $\mathbb{G}$ into a grid of the form $\mathbb{G}' = (\mathbb{X}', \mathbb{O}')$, but by Theorem 2.2.2 does

not chance the Legendrian knot type. Consequently, $\Lambda \in \Lambda(\mathbb{X}')$, so $\Lambda(\mathbb{X}) \subset \Lambda(\mathbb{X}')$. After running the same argument in reverse, we conclude that $\Lambda(\mathbb{X}) = \Lambda(\mathbb{X}')$. □

We are now prepared to prove Theorem 3.1.2.

*Proof.* Consider the set $\Sigma$ in the algorithm described above. Step 1 of the algorithm initializes $\Sigma$ as the set of permutations $\mathbb{X} \in S_{10}$ with $\mathbb{X}_0 = 0$, and therefore $\Lambda(\Sigma)$ at this step is the set of all Legendrian knots described by a grid of size 10 with an $X$ in the bottom left corner. By Lemma 3.1.3, we know that $\Lambda(\Sigma)$ at this point contains every Legendrian knot of grid size 10.

Next, consider the reductions of $\Sigma$ during Step 2. Every time a permutation $\mathbb{X}'$ is removed from $\Sigma$ in Step 2bi, we know that $\mathbb{X}'$ is related to another permutation $\mathbb{X} \in \Sigma$ by a sequence of row and column permutations, and therefore by Lemma 3.1.4, $\Lambda(X) = \Lambda(\mathbb{X}')$. But this means that all of the knots in $\Lambda(\Sigma)$ that would be lost by removing $\mathbb{X}'$ from $\Sigma$ are still contained in $\Lambda(\Sigma)$ through $\Lambda(\mathbb{X})$. Thus, none of the reductions of $\Sigma$ in Step 2 affect $\Lambda(\Sigma)$. □

This algorithm yields a set of 36,336 unique $\mathbb{X}$-permutations, a factor of 100 improvement over the 36 million permutations in $S_{10}$. With this small set, we may build any Legendrian knot of grid size 10.

### Finding $O$-Permutations

In this section, we use the set $\Sigma$ generated in Algorithm 3.1.1 to build a set $S_{\text{NA}}$ which contains at least one grid diagram for each Legendrian knot of arc index 10. (NA stands for no-adjacency; the meaning will become clearer later in the section.)

Before describing an algorithm to accomplish this task, we must introduce a the exact cover problem. Let $\mathcal{U}$ be a set, and let $\mathcal{C}$ be a collection of subsets of $\mathcal{U}$, that is $\mathcal{C} = \{s_1, s_2, ...\}$ such that each $s_i \subset \mathcal{U}$. The exact cover problem asks if there is a subcollection $\mathcal{S} \subset \mathcal{C}$ that exactly covers $\mathcal{U}$. In other words, can we find a set

$\mathcal{S} = \{s_{k_1}, s_{k_2}, ...\}$ such that (1) $s_{k_i} \cap s_{k_j} = \emptyset$ whenever $i \neq j$ and (2) $\bigcup_{s_{k_i} \in \mathcal{S}} = \mathcal{U}$. Knuth's Algorithm X, first described by Donald Knuth in [Knu00] and implemented as Python code in [Ass23], finds *all* solutions this problem; that is, it returns ever subcollection $\mathcal{S}$ that exactly covers $\mathcal{U}$. This algorithm, used elsewhere to solve Sudoku and other puzzles, is the key to generating $S_{\text{NA}}$.

**Algorithm 3.1.5.** Generate O-Permutations

On input $\mathbb{X} \in S_{10}$:

1. Let $S_{\text{NA},\mathbb{X}} = \emptyset$.
2. Create an empty $10 \times 10$ grid, and label the grid spaces in row-major order from 0 to 99.[a]
3. For each $i \in [10]$:
   a. Place a marker in the grid space at column $i$, row $X_i$.
   b. Place four markers in the grid spaces directly above, directly below, directly to the left, and directly to the right of column $i$, row $X_i$. Wrap around the sides of the grid as necessary; for example, if you are asked to place a marker directly to the right of the rightmost column, place it in the leftmost column. This process is illustrated in Figure 3.1.
4. Let $\mathcal{U} = \{r_0, r_1, ...r_9, c_0, c_1, ..., c_9\}$.
5. For each unmarked square numbered $n$, let $i$ be its row and $j$ be its column. (Note that $i = \lfloor n/10 \rfloor$ and $j = n \bmod 10$.) Then construct the set $s_i = \{r_i, c_j\} \subset \mathcal{U}$. Let $\mathcal{C}$ denote the collection of sets $s_i$ constructed in this manner.

6. Invoke Knuth's Algorithm X to find all solutions to the exact cover problem over universe $\mathcal{U}$ and collection $\mathcal{C}$. Each solution is a subcollection $\mathcal{S} \subset \mathcal{C}$ with $|\mathcal{S}| = 10$. For each of these subcollections:

   a. Denote the elements of $\mathcal{S}$ as $s_{n_0}, ...s_{n_9}$, and recall that each $s_{n_k} = \{r_i, c_j\}$ for some $i, j$. Construct a permutation $\mathbb{O} \in S_{10}$ as follows. For each $k \in [10]$, let $i = \lfloor n_k/10 \rfloor$, and $j = n_k \mod 10$, so that $i, j$ are the row and column of the square labeled $n_k$. Then, let $\mathbb{O}_j = i$.

   b. Add $(\mathbb{X}, \mathbb{O})$ to $S_{\text{NA},\mathbb{X}}$.

7. Return $S_{\text{NA},\mathbb{X}}$.

[a] In row-major order, the top-left square is numbered zero, and the remaining squares are labeled in increasing order moving left to right along a row, and moving to the leftmost square of the next column at the end of each row. See Figure 3.1 for an example of a grid with squares numbered in row-major order.

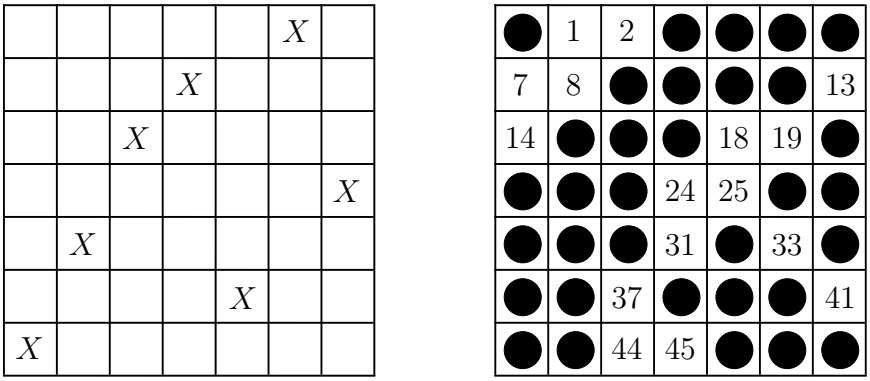


Figure 3.1: **An illustration of the blocked grid created in Step 3 of Algorithm 3.1.5, for a small $7 \times 7$ example grid.** Left: A grid diagram with $X$-markings specified by the permutation $\mathbb{X} = (0, 2, 4, 5, 1, 6, 3)$. Right: The resulting blocked grid, where the unblocked squares are labeled in row-major order (as in Step 2).

To generate $S_{\text{NA}}$, we ran this subroutine on every input $X \in \Sigma$ (where $\Sigma$ is the output of Algorithm 3.1.1), and then took the union of all of the outputs. To prove the correctness of this approach, we must show that every non-destabilizable

Legendrian knot of grid size 10 is represented by some $\mathbb{G} \in S_{\text{NA}}$. To do so, we begin by introducing some terminology and proving a pair of lemmas.

**Definition 3.1.6.** A planar grid diagram $\mathbb{G} = (\mathbb{X}, \mathbb{O}) \in (S_{10})^2$ is *adjacency-free* if no $X$ is placed horizontally or vertically adjacent to an $O$ (including adjacencies that wrap around the edges of the grid diagram). This corresponds to the following combinatorial conditions on $\mathbb{X}$ and $\mathbb{O}$:

$$\text{For all } i \in [10],\ |\mathbb{X}_i - \mathbb{O}_i| \mod 10 \neq 1$$

$$\text{For all } i \in [10],\ \mathbb{X}_i \neq \mathbb{O}_{i+1} \text{ and } \mathbb{O}_i \neq \mathbb{X}_{i+1}$$

Recall that we are taking indices mod 10.

**Definition 3.1.7.** Let $\Lambda_{\text{NA}}(\mathbb{X})$ be the set of Legendrian knots that may be represented by a size-10 adjacency-free grid diagram of the form $(\mathbb{X}, \mathbb{O})$. Then, given a set of permutations $\Sigma \subset S_{10}$, Let $\Lambda_{\text{NA}}(\Sigma) = \bigcup_{\mathbb{X} \in \Sigma} \Lambda_{\text{NA}}(\mathbb{X})$.

**Lemma 3.1.8.** *When run on input $\mathbb{X} \in S_{10}$, Algorithm 3.1.5 returns the set of all adjacency-free grids described by permutations $(\mathbb{X}, \mathbb{O})$ where $\mathbb{O} \in S_{10}$.*

*Proof.* First, we must prove that the $\mathbb{O}$ defined in Step 6a is a permutation. Consider a subcollection $\mathcal{S}$ returned by Knuth's Algorithm X in step 6. Because $\mathcal{S}$ is an exact cover of $\mathcal{U}$, and each set in $s_n \in \mathcal{S}$ is a 2-element subset of $\mathcal{U}$, we know that $|\mathcal{S}| = 10$. Next, recall that each set $s_n = \{r_i, c_j\} \in \mathcal{S}$ corresponds to the $n$-th square in a size-10 grid (when labeled in row-major order), where $r_i$ and $c_j$ are the row and column of square $n$. Now, since $\mathcal{S}$ is an exact cover, each $r_i$ and each $c_j$ may only appear in one $s_n$. Therefore, for two sets $s_n, s_{n'} \in \mathcal{S}$, we know that squares $n$ and $n'$ are in neither the same row nor the same column. Since this property holds for every pair of squares represented by sets in $\mathcal{S}$, we conclude that $\mathcal{S}$ must contain a set representing exactly

one square in each row and one square in each column. This is the exact condition needed to convert a set of squares into a permutation in $S_{10}$.

Having established that the grids generated in Step 6b are indeed valid grid diagrams, we now consider their properties. Let $S_{\mathrm{NA},\mathbb{X}}$ denote the set of grids returned by this algorithm when run on input $\mathbb{X}$, and let $B$ denote the set of adjacency-free grids described by permutations $(\mathbb{X}, \mathbb{O})$ with $\mathbb{O} \in S_{10}$.

- ($\subset$) Suppose $\mathbb{G} = (\mathbb{X}, \mathbb{O}) \in S_{\mathrm{NA},\mathbb{X}}$, and suppose for purposes of contradiction that $\mathbb{G}$ is not adjacency-free. Then there must be some column such that an $O$ is placed one square vertically above or below an $X$, or there must be some row such that an $O$ is placed on square horizontally above or below an $X$. Now, the placements of $O$-markings in $\mathbb{G}$ correspond to the set of squares $\mathcal{S}$ chosen by Knuth's Algorithm X in Step 6. However, in Step 3, every square horizontally or vertically adjacent to an $X$ was blocked. Therefore, it is impossible for Step 6 to have selected a square that is adjacent to an $X$. This is a contradiction, so we conclude that $\mathbb{G}$ must be adjacency free. Therefore, $S_{\mathrm{NA},\mathbb{X}} \subset B$.

- ($\supset$) Suppose $\mathbb{G} = (\mathbb{X}, \mathbb{O}) \in B$, and consider the set of squares $n_0, ..., n_9$ corresponding to the positions of $O$s in $\mathbb{G}$. Since $\mathbb{G}$ is adjacency-free, none of these squares were blocked during Step 3, and thus each of these squares were converted into a set $s_i$ in Step 5. Next, since $\mathbb{O}$ is a permutation, the set $\mathcal{S} = \{s_{n_0}, ..., s_{n_9}\}$ will exactly cover $\mathcal{U}$. Therefore, $\mathcal{S}$ will be found by Knuth's Algorithm X in step 6, and the algorithm will return $(\mathbb{X}, \mathbb{O})$. This shows that for any $\mathbb{G} \in B$, $\mathbb{G} \in S_{\mathrm{NA},\mathbb{X}}$, and therefore $B \subset S_{\mathrm{NA},\mathbb{X}}$.

In conclusion, the set of grids returned by the above algorithm is exactly equal to the set of adjacency-free grids described by permutations $(\mathbb{X}, \mathbb{O})$ for some $\mathbb{O} \in S_{10}$. □

**Lemma 3.1.9.** *Suppose $\mathbb{G}$ is a $10 \times 10$ planar grid diagram that is not adjacency-free. Then $\Lambda(\mathbb{G})$ is not a non-destabilizable Legendrian knot of grid size 10.*

*Proof.* By assumption, this grid contains an $X$-marked square and an $O$-marked square which are vertically or horizontally adjacent. (If the adjacency wraps around the edge of the grid, simply perform a cyclic row and column permutation to move the adjacency away from the edge.) By Theorem 2.2.2, we may commute the row or column containing these adjacent marked squares until three markings are brought into an L-shape, at which point the grid may be destabilized. The resulting grid $\mathbb{G}'$ will be a $9 \times 9$ grid diagram such that either $\Lambda(\mathbb{G}') = \Lambda(\mathbb{G})$ or $S_{\pm}(\Lambda(\mathbb{G}')) = \Lambda(\mathbb{G})$ (depending on what type of destabilization was performed). In the first case, we see that $\Lambda(\mathbb{G})$ may be represented by a grid of size less than 10, the grid size of $\Lambda$ is less than 10. In the second case Definition 2.2.4 tells us that $\Lambda(\mathbb{G})$ is not non-destabilizable. □

We are now prepared to prove that the set $S_{\mathrm{NA}}$ does indeed contain a grid representing every non-destabilizable Legendrian knot of grid size 10.

**Theorem 3.1.10.** *Let $S_{\mathrm{NA}} = \bigcup_{\mathbb{X}\in\Sigma} S_{\mathrm{NA},\mathbb{X}}$, where $S_{\mathrm{NA},\mathbb{X}}$ is the set of grids returned by Algorithm 3.1.5 when run on input $\mathbb{X} \in S_{10}$, and $\Sigma$ is the set of permutations returned by Algorithm 3.1.1. Every non-destabilizable Legendrian knot of grid size 10 is represented by some planar grid diagram $\mathbb{G} \in S_{\mathrm{NA}}$.*

*Proof.* By Lemma 3.1.8, we know that $S_{\mathrm{NA}}$ (as defined in Theorem 3.1.10) is a set of all adjacency-free grid diagrams of the form $(\mathbb{X}, \mathbb{O})$, where every $\mathbb{X}$ is in the set $\Sigma$ returned by Algorithm 3.1.1. Furthermore, Theorem 3.1.2 tells us that every Legendrian knot (and therefore every non-destabilizable Legendrian knot) of grid size 10 may be represented by a grid of the form $(\mathbb{X}, \mathbb{O})$ where $\mathbb{X} \in \Sigma$. Now, by Lemma 3.1.9, zero knots in $\Lambda(\Sigma) - \Lambda_{\mathrm{NA}}(\Sigma)$ are non-destabilizable Legendrian knots of grid size 10. Therefore, every non-destabilizable Legendrian knot of grid size 10 must be in $\Lambda_{\mathrm{NA}}(\Sigma)$, and therefore can be represented by a grid diagram $\mathbb{G} = (\mathbb{X}, \mathbb{O}) \in S_{\mathrm{NA}}$. □

Running Algorithm 3.1.5 on every $\mathbb{X} \in \Sigma$ (where $\Sigma$ is the output Algorithm 3.1.1), we obtain about 200 million grids, far less than the upper bound of $10^{14}$ established earlier in the chapter.

### Removing Links

In the previous section, we generated a set of about 200 million adjacency-free grids. However, not all of these grids represent knots; some of them represent links with two or more components. We filtered out these links by tracing the grid diagram from $X$ to $O$ along vertical lines, and $O$ to $X$ along horizontal lines. If we found a closed loop before encountering all 20 markings, then we knew that it represented a multiple-component link, so we removed it from the set. After discarding grids in this way, we were left with a set of 89,763,984 grids, which we knew contained at least one grid diagram representing each non-destablizable Legendrian knot of grid size 10.

Section 3.2

# Reducing the Set of Grids

In the previous section, we outlined a process to generate a set of grid diagrams $\mathcal{S}$, and proved that every Legendrian knot of Legendrian grid size 10 is represented by some diagram $\mathbb{G} \in \mathcal{S}$. This algorithm returned a set of 89 million distinct grids. However, there are two reasons that this set was far from the *smallest* possible set containing all Legendrian knots of Legendrian grid size 10.

(a) Some grids $\mathbb{G} \in \mathcal{S}$ are destabilizable or have grid size less than 10.

(b) The set $\mathcal{S}$ might contain duplicate grids; that is that two different grids $\mathbb{G}_1, \mathbb{G}_2 \in \mathcal{S}$ may represent the same Legendrian knot.

Therefore, to find a minimal set of grids, we attempted to remove both destabilizable and duplicate grids. To do so, we adopted the following approach. For every grid $\mathbb{G} \in \mathcal{S}$, we generated a 'bubble' of equivalent grids using a combination of commutation moves, stabilization moves, and cyclic row and column permutation moves. We then used these bubbles to hunt for destabilizable grids and Legendrian isotopies between grids.

### Generating Bubbles

Given a grid diagram $\mathbb{G}$, an easy way to generate several equivalent grid diagrams is to make a number of cyclic row and column permutations. The following does exactly that, returning all grids that can be reached from $\mathbb{G}$ via cyclic row and columns, and have an $X$ in the bottom left corner. For a single input, this algorithm typically outputs ten grids.

**Algorithm 3.2.1.** Generate Cyclic-Permutation-Equivalent Grids

On input $\mathbb{G} = (\mathbb{X}, \mathbb{O}) \in (S_{10}^2)$:

1. Let $\mathcal{S} = \emptyset$.
2. For $col \in [10]$:
   a. Let $row = \mathbb{X}_{col}$. Then define two new permutations $\mathbb{X}', \mathbb{O}' \in (S_{10})^2$ as follows, recalling that indices are taken mod 10:
      - $\mathbb{X}'_i = (\mathbb{X}_{i+row} - col) \mod 10$
      - $\mathbb{O}'_i = (\mathbb{O}_{i+row} - col) \mod 10$
   b. Add the grid $\mathbb{G}' = (\mathbb{X}', \mathbb{O}')$ to $\mathcal{S}$.
3. Return $\mathcal{S}$.

**Lemma 3.2.2.** *Let $\mathcal{S}$ be the set of grids returned by Algorithm 3.2.1 when run on input $\mathbb{G}$. Then $\Lambda(\mathbb{G}') = \Lambda(\mathbb{G})$ for every grid $\mathbb{G}' \in \mathcal{S}$.*

*Proof.* Consider an arbitrary grid $\mathbb{G}' \in \mathcal{S}$, added to $\mathcal{S}$ in Step 2c while the column counter had value *col*. It is related to $\mathbb{G}$ by the following sequence of cyclic row and column permutations. Starting with $\mathbb{G}$, first permute the bottommost row to the top $\mathbb{X}_{col}$ times. Then permute the leftmost column to the right *col* times. Figure 3.2 illustrates this sequence. Examining this figure, it is evident that the pair of permutations $(\mathbb{X}', \mathbb{O}')$ generated in Step 2a describes the rightmost grid. Since $\mathbb{G}' = (\mathbb{X}', \mathbb{O}')$ and $\mathbb{G}$ are related by a sequence of cyclic row and column permutations, Theorem 2.2.2 tells us that $\Lambda(\mathbb{G}) = \Lambda(\mathbb{G}')$. □

Next, we wrote an algorithm to perform row and column commutations. Given a grid diagram $\mathbb{G}$, it checks if each commuting each pair of adjacent rows or columns is a legal grid move, and if so, returns the resulting grid. There are 18 possible row and/or column commutations for every grid. However, since not all of these commutations

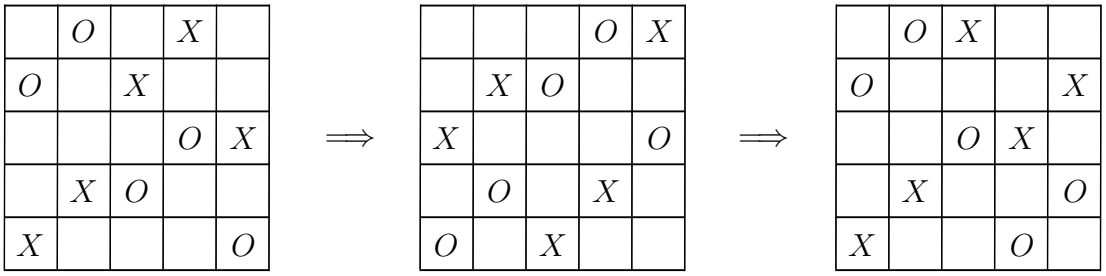


Figure 3.2: **An example of how Algorithm 3.2.1 generates grids diagrams, using a small $5\times5$ grid diagram.** Left: A grid diagram $\mathbb{G}$ specified by permutations $\mathbb{X} = (0, 1, 3, 4, 2)$, $\mathbb{O} = (3, 4, 1, 2, 0)$. Center: The result of moving the bottommost row of $\mathbb{G}$ to the top 3 times. Right: The result of diagram is the result of moving the rightmost column of the center diagram to the left 2 times. This would be the grid added to $\mathcal{S}$ at Step 2b of Algorithm 3.2.1 with $col = 2$ ($\mathbb{X}_{col} = 3$). Observe that there is an $\mathbb{X}$ in the bottom left corner of this diagram.

are legal, this algorithm typically outputs around 5 grids.

**Algorithm 3.2.3.** Generate Commutation-Equivalent Grids

On input $\mathbb{G} = (\mathbb{X}, \mathbb{O}) \in (S^2_{10})$:

1. Let $\mathcal{S} = \emptyset$.

2. For $col \in [9]$:

   a. Check if the commutation of columns $col$ and $col + 1$ would be a legal Legendrian grid move. To do so, let $a = \min(\mathbb{X}_{col}, \mathbb{O}_{col})$, $b = \max(\mathbb{X}_{col}, \mathbb{O}_{col})$, $c = \min(\mathbb{X}_{col+1}, \mathbb{O}_{col+1})$, and $d = \max(\mathbb{X}_{col+1}, \mathbb{O}_{col+1})$. Then, check if any one of the following four conditions is true.

$$a > d$$

$$b < c$$

$$a > c \text{ and } b < d$$

$$a < c \text{ and } b > d$$

b. If so, define two new permutations $\mathbb{X}', \mathbb{O}' \in (S_{10})^2$ as follows:

$$\mathbb{X}'_i = \begin{cases} \mathbb{X}_{col} & i = col+1 \\ \mathbb{X}_{col+1} & i = col \\ \mathbb{X}_i & \text{otherwise} \end{cases} \qquad \mathbb{O}'_i = \begin{cases} \mathbb{O}_{col} & i = col+1 \\ \mathbb{O}_{col+1} & i = col \\ \mathbb{O}_i & \text{otherwise} \end{cases}$$

c. Add the grid $\mathbb{G}' = (\mathbb{X}', \mathbb{O}')$ to $\mathcal{S}$.

3. For $row \in [9]$:

a. Check if the commutation of rows $row$ and $row+1$ is a legal Legendrian Reidemeister move, following a similar procedure as the one used in Step 2.

b. If so, define two new permutations $\mathbb{X}', \mathbb{O}' \in (S_{10})^2$ as follows:

$$\mathbb{X}'_i = \begin{cases} row & \mathbb{X}_i = row+1 \\ row+1 & \mathbb{X}_i = row \\ \mathbb{X}_i & \text{otherwise} \end{cases} \qquad \mathbb{O}'_i = \begin{cases} row & \mathbb{O}_i = row+1 \\ row+1 & O_i = row \\ \mathbb{O}_i & \text{otherwise} \end{cases}$$

c. Add the grid $\mathbb{G}' = (\mathbb{X}', \mathbb{O}')$ to $\mathcal{S}$.

4. Return $\mathcal{S}$.

**Lemma 3.2.4.** *Let $\mathcal{S}$ be the set of grids returned by Algorithm 3.2.3 when run on input $\mathbb{G}$. Then $\Lambda(\mathbb{G}') = \Lambda(\mathbb{G})$ for every grid $\mathbb{G}' \in \mathcal{S}$.*

*Proof.* Consider an arbitrary grid $\mathbb{G}' \in \mathcal{S}$, and suppose that $\mathbb{G}'$ was added to $\mathcal{S}$ in Step 2c while the column counter had value $i$. I claim that $\mathbb{G}'$ is related to $\mathbb{G}$ by the commutation of columns $i$ and $i$, and that the commutation of those two columns is a legal Legendrian grid move. Now, since Algorithm 3.2.3 reaches Step 2c, then one

of the conditions in Step 2a must be satisfied. Working case-by-case:

- If $a > d$, then the $X$ and $O$ in column $i$ are both 'above' the $X$ and $O$ in column $i+1$. Thus, the vertical lines in columns $i$ and $i+1$ are disjoint, so the column commutation is allowed.
- If $b < c$, then the $X$ and $O$ in column $i$ are both 'below' the $X$ and $O$ in column $i+1$. Thus, the vertical lines in columns $i$ and $i+1$ are disjoint, so the column commutation is allowed.
- If $a > c$ and $b < d$, then the $X$ and $O$ in column $i$ are located 'between' the $X$ and $O$ in column $i+1$. Thus, the vertical lines in columns $i$ and $i+1$ are nested, so the column commutation is allowed.
- If $a < c$ and $b > d$, then the $X$ and $O$ in column $i+1$ are located 'between' the $X$ and $O$ in column $i$. Thus, the vertical lines in columns $i$ and $i+1$ are nested, so the column commutation is allowed.

Thus, Algorithm 3.2.3 will only reach Step 2c if the commutation of columns $i$ and $i+1$ is a legal Legendrian grid move. See Figure 3.3 for an example of a grid diagram with such a legal column commutation. Examining this figure, we see that the permutations $(\mathbb{X}', \mathbb{O}')$ generated in Step 2b indeed describe the grid generated by commuting columns $i$ and $i+1$. Thus, since $\mathbb{G}' = (\mathbb{X}', \mathbb{O}')$ is related to $\mathbb{G}$ by a Legendrian grid move, $\Lambda(\mathbb{G}') = \Lambda(\mathbb{G})$. Finally, if $\mathbb{G}'$ was added to $\mathcal{S}$ in Step 3c instead of 2c, an identical argument shows that $\mathbb{G}'$ and $\mathbb{G}$ are related by a row commutation, and we again have $\Lambda(\mathbb{G}') = \Lambda(\mathbb{G})$. □

Next, we used our cyclic-permutation and commutation algorithms to implement the following bubble-generation algorithm. This algorithm works by alternating calls to Algorithm 3.2.3 and Algorithm 3.2.1 to build a growing set of grid diagrams, all of which represent the same Legendrian knot.

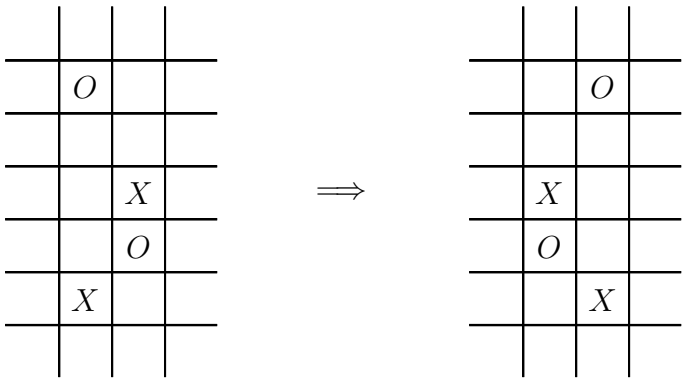


Figure 3.3: Left: A local picture of a grid, showing its $X$- and $O$-markings in columns $i$ and $i+1$. Right: The result of commuting columns $i$ and $i+1$. For this grid, the commutation of column $i$ and column $i+1$ is possible because the markings in column $i$ are nested between the markings in column $i+1$. Combinatorially, this corresponds to the condition that $\min\{\mathbb{X}_i, \mathbb{O}_i\} < \min\{\mathbb{X}_{i+1}, \mathbb{O}_{i+1}\}$ and $\max\{\mathbb{X}_i, \mathbb{O}_i\} > \max\{\mathbb{X}_{i+1}, \mathbb{O}_{i+1}\}$.

**Algorithm 3.2.5.** Generate Bubble

On inputs $(S, \mathit{depth})$, where $S$ is a set of grids such that each grid represents the same Legendrian knot $\Lambda$ and *depth* is an integer depth parameter:

1. Initialize $\mathcal{S} = S$ and $\mathit{lastLayer} = S$.
2. Repeat the following procedure *depth* times:
   a. Initialize $T_1 = \emptyset$. Then, for each grid $\mathbb{G} \in \mathit{lastLayer}$, find all commutation-equivalent grids by running Algorithm 3.2.3 on input $\mathbb{G}$, and add those grids to $T_1$. Note that by Lemma 3.2.4, every grid $\mathbb{G}' \in T_1$ satisfies $\Lambda(\mathbb{G}') = \Lambda(\mathbb{G}) = \Lambda$.
   b. Initialize $T_2 = \emptyset$. Then, for each grid $\mathbb{G} \in T_1$, find all cyclic-permutation-equivalent grids by running Algorithm 3.2.1 on input $\mathbb{G}$, and add those grids to $T_2$. Note that by Lemma 3.2.2, every grid $\mathbb{G}' \in T_2$ satisfies $\Lambda(\mathbb{G}') = \Lambda(\mathbb{G}) = \Lambda$.

c. Let $lastLayer = T_2 - \mathcal{S}$. Then, add all grids in $lastLayer$ to $\mathcal{S}$.

3. Return $\mathcal{S}$.

**Theorem 3.2.6.** *Let $S$ be a set of grids such that each grid represents the same Legendrian knot $\Lambda$, and let $\mathcal{S}$ be the set of grids returned by Algorithm 3.2.5 when run on inputs $(S, depth)$. Then $\Lambda(\mathbb{G}') = \Lambda$ for every grid $\mathbb{G}' \in \mathcal{S}$.*

*Proof.* We prove this by induction on the depth parameter *depth*. Fix an input set $S$, and let $\mathcal{S}_d$ denote the output of the algorithm when $depth = d$. For $d = 0$, $\mathcal{S}_0 = S$ and the theorem is trivially true. When $depth = 1$, at the end of Step 2 every grid $\mathbb{G}' \in lastLayer \subset T_2$ satisfies $\Lambda(\mathbb{G}') = \Lambda$; and since $\mathcal{S}_1 = S \cup T_2$, the theorem holds again. Finally, notice that $\mathcal{S}_{d+1}$ equals the output of the algorithm when run on inputs $(\mathcal{S}_d, 1)$. Therefore, by the principle of induction, the theorem holds for all possible values of *depth*. □

By our order-of-magnitude estimates, we can estimate that $|T_1| \approx 5|lastLayer|$ and $|T_2| \approx 10|T_1|$. Therefore, in each iteration of the loop in Step 2, it is possible that *lastLayer* grows by a factor of 50. In practice, this number is diminished by the presence of identical grids, meaning *lastLayer* typically grows by a factor between 1 and 2. Nonetheless, even with 'small' depth parameters on the order of 20 or so, it is possible to generate hundreds of thousands of grid diagrams that all represent the same Legendrian knot.

### Eliminating Duplicate Grids

Using the bubble-generation algorithm described in the previous section, we now turn to the problem of shrinking our set of grids $\mathcal{S}$, while preserving the property that $\mathcal{S}$ contains at least one grid diagram for each non-destabilizable Legendrian knot of grid size 10. To do so, we implemented the following algorithm, which searches both for destabilizable grids and for isotopies between different grids.

**Algorithm 3.2.7.** Cut Grids

On input $(S, \textit{depth})$, where $S$ is a set of grids and *depth* is an integer depth parameter:

1. Let $\mathcal{S} = S$, and let $n = |\mathcal{S}|$.
2. Number the grids in $\mathcal{S}$ as $\mathbb{G}_0, ..., \mathbb{G}_{n-1}$. Then, for each grid $\mathbb{G}_i$, construct a set of grids $T_i$ by running Algorithm 3.2.5 on inputs $(\mathbb{G}_i, \textit{depth})$. Note that by Theorem 3.2.6, every grid $\mathbb{G}_i' \in T_i$ satisfies $\Lambda(\mathbb{G}_i') = \Lambda(\mathbb{G}_i)$.
3. For each $i \in [n]$, if $T_i$ contains a grid with an adjacency, remove $\mathbb{G}_i$ from $\mathcal{S}$.
4. For each pair $(i, j) \in [n]^2$ with $j > i$, if the intersection of $T_i$ and $T_j$ is nonempty, remove $\mathbb{G}_j$ from $\mathcal{S}$.
5. Return $\mathcal{S}$.

To prove the correctness of this algorithm, we must show that it never removes from $\mathcal{S}$ all grids that represent a certain non-destabilizable knot type.

**Lemma 3.2.8.** *Let $\mathcal{S}$ be the set of grids returned by Algorithm 3.2.7 when run on inputs $(S, \textit{depth})$. For every grid $\mathbb{G}$ in $S$, if $\Lambda(\mathbb{G})$ is a nondestablizable Legendrian knot of grid size 10, then there exists a grid $\mathbb{G}' \in \mathcal{S}$ such that $\Lambda(\mathbb{G}) = \Lambda(\mathbb{G}')$.*

*Proof.* Consider an arbitrary $\mathbb{G}_j \in S$. If $\mathbb{G}_j$ was never removed from $\mathcal{S}$, then the lemma holds trivially for $\mathbb{G}' = \mathbb{G}_j$. If $\mathbb{G}_j$ was removed from $\mathcal{S}$ in Step 3, then we know by Lemma 3.1.9 that $\Lambda(\mathbb{G}_j)$ is destabilizable. Finally, if $\mathbb{G}_j$ was removed from $\mathcal{S}$ in Step 4, then we know that there exists a grid $\mathbb{G}_{\text{intermediate}}$ in both $T_i$ and $T_j$ for some $i < j$, and therefore $\Lambda(\mathbb{G}_j) = \Lambda(\mathbb{G}_{\text{intermediate}}) = \Lambda(\mathbb{G}_i)$. Thus, we may consider the nonempty set of grids in $S$ that are equivalent to $\mathbb{G}_j$, i.e. the set of grids that represent $\Lambda(\mathbb{G}_j)$. Note that the lowest-numbered grid in this set (which we denote

$\mathbb{G}_{n_0}$) may never be removed by Step 4. Therefore, after Algorithm 3.2.7 completes Step 4, we may consider two cases. First, if $\mathbb{G}_{n_0}$ remains in $\mathcal{S}$, then the lemma is satisfied with $\mathbb{G}' = \mathbb{G}_{n_0}$. On the other hand, if $\mathbb{G}_{n_0}$ does not remain in $\mathcal{S}$, it must have been removed in Step 3, in which case $\Lambda(\mathbb{G}_j)$ is destabilizable and the lemma is still satisfied. □

Lemma 3.2.8 implies that Algorithm 3.2.7 removes only duplicate grids or grids corresponding to destabilizable Legendrian knots. Therefore, we may safely run this algorithm to reduce the size of our set $\mathcal{S}$ of grids, knowing that we never accidentally remove all grids representing destabilizable Legendrian knot type.

### Incorporating Stabilizations

Notice that the algorithm in the previous section does not incorporate stabilization moves. However, it is possible that two grids may be transformed into one another only through a sequence of stabilization and destabilization moves. To handle these cases, we began by implementing the following algorithm. This algorithm accepts a $10 \times 10$ grid diagram $\mathbb{G}$, stabilizes it at every marking, and returns the resulting set of $11 \times 11$ grid diagrams. Theorem 2.2.2 tells us that at every $X$- or $O$-marking, there are two ways to stabilze $\mathbb{G}$ while preserving Legendrian knot type. Therefore, this algorithm typically outputs 40 grids. (It will output fewer than 40 grids only if two different stabilization moves result in identical grids.)

**Algorithm 3.2.9.** Stabilize Grid

On input $\mathbb{G} = (\mathbb{X}, \mathbb{O}) \in (S_{10})^2$:

1. Let $\mathcal{S} = \emptyset$.
2. For $col \in [10]$:

a. Construct permutations $\mathbb{X}', \mathbb{O}'$ by performing an *X:NE* stabilization on the $X$ in column $col$. To do so, let $row = \mathbb{X}_{col}$. Then construct permutations $\mathbb{X}'$ and $\mathbb{O}'$ as follows:

$$\mathbb{X}'_i = \begin{cases} \mathbb{X}_i & i < col, \mathbb{X}_i < row \\ \mathbb{X}_i + 1 & i < col, \mathbb{X}_i > row \\ \mathbb{X}_{i-1} & i > col + 1, \mathbb{X}_i < row \\ \mathbb{X}_{i-1} + 1 & i > col + 1, \mathbb{X}_i > row \\ row + 1 & i = col \\ row & i = col + 1 \end{cases}$$

and

$$\mathbb{O}'_i = \begin{cases} \mathbb{O}_i & i < col, \mathbb{O}_i < row \\ \mathbb{O}_i + 1 & i < col, \mathbb{O}_i \geq row \\ \mathbb{O}_{i-1} & i > col + 1, \mathbb{O}_i < row \\ \mathbb{O}_{i-1} + 1 & i > col + 1, \mathbb{O}_i \geq row \\ row & i = col \end{cases}$$

Finally, let $\mathbb{G}' = (\mathbb{X}', \mathbb{O}')$ and add $\mathbb{G}'$ to $\mathcal{S}$.

b. Repeat step 2a three more times. The first time, perform an *X:SW* stabilization at the $X$ in column $col$. The second time, perform an *O:NE* stabilization at the $O$ in column $col$. The third time, perform an *O:SW* stabilization at the $O$ in column $col$.

3. Return $\mathcal{S}$.

**Lemma 3.2.10.** *Let $\mathcal{S}$ be the set of grids returned by Algorithm 3.2.9 when run on input $\mathbb{G}$. Then $\Lambda(\mathbb{G}') = \Lambda(\mathbb{G})$ for every grid $\mathbb{G}' \in \mathcal{S}$.*

*Proof.* Consider an arbitrary grid $\mathbb{G}' \in \mathcal{S}$, and suppose $\mathbb{G}' = (\mathbb{X}', \mathbb{O}')$ was added to $\mathcal{S}$ in Step 2a when the the column counter had value *col*. Figure 3.4 demonstrates that the permutations $\mathbb{X}', \mathbb{O}'$ describe a grid that is related to $\mathbb{G}$ by an *X:NE* stabilization on the $X$ in column *col*. Since $\mathbb{G}'$ and $\mathbb{G}$ are related by a Legendrian grid move, $\Lambda(\mathbb{G}') = \Lambda(\mathbb{G})$. Finally, if $\mathbb{G}'$ was added to $\mathcal{S}$ in Step 2b, $\mathbb{G}'$ must be related to $\mathbb{G}$ by one of the moves identified in Theorem 2.2.2, again yielding $\Lambda(\mathbb{G}') = \Lambda(\mathbb{G})$. □

| | | | | |
|---|---|---|---|---|
| | | $O$ | | $X$ |
| | $O$ | | $X$ | |
| $O$ | | $X$ | | |
| | $X$ | | | $O$ |
| $X$ | | | $O$ | |

| | | | | | |
|---|---|---|---|---|---|
| | | $O$ | | | $X$ |
| | $O$ | | | $X$ | |
| $O$ | | | $X$ | | |
| | | $X$ | $O$ | | |
| | $X$ | | | | $O$ |
| $X$ | | | | $O$ | |

Figure 3.4: Left: The grid diagram described by permutations $\mathbb{X} = (0, 1, 2, 3, 4)$, $\mathbb{O} = (2, 3, 4, 0, 1)$. Right: The grid diagram obtained by performing an *X:NE* stabilization at the $X$ in column 2 of the diagram at left. It is described by permutations $\mathbb{X}' = (0, 1, 2, 3, 4, 5, 6)$, $\mathbb{O} = (3, 4, 5, 2, 0, 1)$. Note that these permutations are the permutations generated by Algorithm 3.2.9 Step 2a when run on input $(\mathbb{X}, \mathbb{O})$, when the column counter $col = 2$. (This example uses $5 \times 5$ rather than 10 grid diagrams, but the size of the grid is unrelated to the correctness of the algorithm.)

We then used this stabilization algorithm to search for isotopies involving a single stabilization and destabilization move.

**Algorithm 3.2.11.** Cutting Grids, with Stabilization

On inputs $S$, *depth*, where $S$ is a set of grids and $depth \in \mathbb{N}$:

1. Let $\mathcal{S} = S$, and let $n = |\mathcal{S}|$.

2. Number the grids in $\mathcal{S}$ as $G_0, ..., G_{n-1}$. Then, for each grid $\mathbb{G}_i$, construct a set of stabilized grids $T_i$ by running Algorithm 3.2.9 on inputs $(\mathbb{G}_i, depth)$. By Lemma 3.2.10, every grid $\mathbb{G}' \in T_i$ satisfies $\Lambda(\mathbb{G}') = \Lambda(\mathbb{G}_i)$.

3. For $i \in [n]$:

   a. Let $U_i = \emptyset$. Then, for each grid $\mathbb{G}_{\text{stab}} \in T_i$, find all cyclic-permutation-equivalent grids by running Algorithm 3.2.1 on input $\mathbb{G}_{\text{stab}}$, and add those grids to $U_i$. By Lemma 3.2.2, every grid $\mathbb{G}' \in U_i$ satisfies $\Lambda(\mathbb{G}') = \Lambda(\mathbb{G}_i)$.

   b. Generate a bubble of grids $V_i$ by running Algorithm 3.2.5 on inputs $(U_i, \mathit{depth})$. By Theorem 3.2.6, every grid $\mathbb{G}' \in V_i$ satisfies $\Lambda(\mathbb{G}') = \Lambda(\mathbb{G}_i)$.

4. For each pair $(i, j) \in [n]^2$ with $j > i$, if the intersection of $V_i$ and $V_j$ is nonempty, remove $\mathbb{G}_j$ from $\mathcal{S}$.

5. Return $\mathcal{S}$.

Just as before, we must show that this algorithm never removes all grid diagrams representing a single Legendrian knot type.

**Lemma 3.2.12.** *Let $\mathcal{S}$ be the set of grids returned by Algorithm 3.2.11. For every $\mathbb{G} \in S$, if $\Lambda(\mathbb{G})$ is a Legendrian knot of grid size 10, then there exists a grid $\mathbb{G}' \in \mathcal{S}$ such that $\Lambda(\mathbb{G}') = \Lambda(\mathbb{G})$.*

*Proof.* First, I claim that if a grid $\mathbb{G}_j \in S$ is removed from $\mathcal{S}$ in Step 4, then there exists a grid $\mathbb{G}_i \in S$ with $i < j$ and $\Lambda(\mathbb{G}_i) = \Lambda(\mathbb{G}_j)$. To see this, notice that if $\mathbb{G}_j$ is removed from $\mathcal{S}$ in Step 4, then there must exist a grid $\mathbb{G}'$ in both $V_i$ and $V_j$, where $i < j$. Therefore, there must exist a grid $\mathbb{G}_i$ with $i < j$ and $\Lambda(\mathbb{G}_i) = \Lambda(\mathbb{G}') = \Lambda(\mathbb{G}_j)$.

The remainder of the proof proceeds similarly to the proof of Lemma 3.2.8. Consider an arbitrary $\mathbb{G}_j \in S$. If $\mathbb{G}_j$ was never removed from $\mathcal{S}$, then the the lemma holds trivially with $\mathbb{G}' = \mathbb{G}_j$. On the other hand, $\mathbb{G}_j$ was removed from $\mathcal{S}$ in Step 4, consider the nonempty set of grids that are equivalent to $\mathbb{G}_j$, and take the lowest-numbered

grid in this set, $\mathbb{G}_{n_0}$. As shown above, this grid can never be removed from $\mathcal{S}$, so our lemma holds with $\mathbb{G}' = \mathbb{G}_{n_0}$. □

Lemma 3.2.12 implies that Algorithm 3.2.11 only removes duplicate grids. Therefore, we may safely run this algorithm to reduce the size of our set $\mathcal{S}$ of grids, knowing that we never remove all grids representing a Legendrian knot.

Section 3.3

## Organizing the Candidate Set

The 'bubble' algorithms outlined in the previous section are subject to serious limitations. The set of grids we found in Section 3.1 contains nearly 100 million grids; simultaneously storing millions or even thousands of bubbles is memory-intensive, especially when the depth parameters and size of these bubbles are large. Furthermore, using these bubbles to search for Legendrian-isotopic grid pairs requires comparing the bubble of every grid to the bubble of every other grid, which has quadratic runtime in the number of grids.

To overcome these limitations, we took advantage of knot invariants. If we find that grids $\mathbb{G}_1$ and $\mathbb{G}_2$ have different invariants, then we immediately know that $\mathbb{G}_1$ and $\mathbb{G}_2$ represent different Legendrian knots $\Lambda$, and it is no longer necessary to look for isotopies by comparing their bubbles. This suggests the following approach:

(a) Compute a handful of knot invariants for each grid in our set.

(b) Place grids into 'buckets', where each bucket is filled only with grids with matching invariants.

(c) One at a time, apply the bubble algorithm to grids in each bucket.

This approach addresses the memory constraint, because we must now only store the bubbles for grids in a single bucket rather than for all grids in our broader set. This procedure also addresses the runtime constraint, because the runtime is quadratic in the size of each bucket and not in the size of the broader set. Furthermore, the bubble and cutting algorithms of each bucket are completely independent and can thus be run in parallel. This has the advantage of further improving the runtime, at the cost of requiring more memory.

To implement this approach, we bucketed grids according to four properties. First, we bucketed by the classical Legendrian knot invariants $tb$ and $r$. Next, we refined this bucketing with the Alexander polynomial. Last, we bucketed by topological knot type. In this section, we describe the algorithms used to compute these properties.

### Organizing by Classical Invariants

We began by bucketing by the classical Legendrian knot invariants, the Thurston-Benneqin number $tb$ and the rotation number $r$. To compute these invariants, we drew upon definitions and ideas presented in [OSS15].

**Definition 3.3.1.** First, define a partial ordering on $\mathbb{R}^2$ as follows. For two points $a = (a_x, a_y)$ and $b = (b_x, b_y)$, we say that $a > b$ if $a_x > b_x$ and $a_y > b_y$. Now, let $P, Q$ be finite sets of points in $\mathbb{R}^2$. Denote the elements of these sets as $P = \{p_1, ..., p_n\}$ and $Q = \{q_1, ..., q_m\}$. Define a function $\mathcal{J}$ as follows:

$$\mathcal{J}(P,Q) = \frac{1}{2}\sum_{i=0}^{n}\sum_{j=0}^{m}\sum_{p_i>q_j} 1 + \frac{1}{2}\sum_{i=0}^{n}\sum_{j=0}^{m}\sum_{q_j>p_i} 1$$

**Definition 3.3.2.** A grid state of an $n \times n$ toroidal grid diagram $\mathbb{G}$ is a collection of points on that diagram, such that exactly one point lies on each vertical and horizontal circle. Note that for a toroidal grid diagram, the vertical / horizontal circles appear as vertical / horizontal lines. Furthermore, the topmost line is the same as the bottommost, and the leftmost the same as the rightmost.

**Definition 3.3.3.** The grid state $\mathbf{x}^+$ of a grid diagram $\mathbb{G} = (\mathbb{X}, \mathbb{O})$ is the set of points that lie on the upper right-hand corners of the grid squares marked with an $\mathbb{X}$. The grid state $\mathbf{x}^-$ of a grid diagram is the set of points that lie on the lower left-hand corners of the grid squares marked with an $\mathbb{X}$. For an example of these grid states, see Figure 3.5.

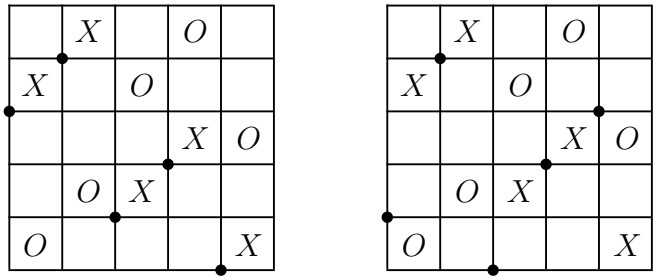


Figure 3.5: Left: The $\mathbf{x}^-$ grid state of the grid diagram described by permutations $\mathbb{X} = (3, 4, 1, 2, 0), \mathbb{O} = (0, 1, 3, 4, 2)$, where the black dots represent the points in $\mathbf{x}^-$. Right: The $\mathbf{x}^+$ grid state of the same grid diagram.

With these definitions in mind, we implemented the following algorithm to compute the classical invariants.

**Algorithm 3.3.4.** Computing $tb, r$

On input $\mathbb{G} = (\mathbb{X}, \mathbb{O}) \in (S_n)^2$:

1. Construct three sets of points as follows:

   - $\mathcal{P}_{\mathbb{O}} = \{(i + 0.5, \mathbb{O}_i + 0.5) : i \in [n]\}$
   - $\mathbf{x}^- = \{(i, \mathbb{X}_i) : i \in [n]\}$
   - $\mathbf{x}^+ = \{((i + 1) \mod n, \mathbb{X}_{i+1}) : i \in [n]\}$

   where we remember to take indices mod $n$. Note that $\mathcal{P}_{\mathbb{O}}$ corresponds to the locations of the $\mathbb{O}$ markings, where these markings are taken to be located at the center of their grid squares.

2. Compute the Maslov gradings of the $\mathbf{x}^+$ and $\mathbf{x}^-$ states as follows:

   - $M_{\mathbb{O}}(\mathbf{x}^+) = \mathcal{J}(\mathbf{x}^+, \mathbf{x}^+) - 2\mathcal{J}(\mathbf{x}^+, \mathcal{P}_{\mathbb{O}}) + \mathcal{J}(\mathcal{P}_{\mathbb{O}}, \mathcal{P}_{\mathbb{O}}) + 1.$
   - $M_{\mathbb{O}}(\mathbf{x}^-) = \mathcal{J}(\mathbf{x}^-, \mathbf{x}^-) - 2\mathcal{J}(\mathbf{x}^-, \mathcal{P}_{\mathbb{O}}) + \mathcal{J}(\mathcal{P}_{\mathbb{O}}, \mathcal{P}_{\mathbb{O}}) + 1.$

3. Let $tb = \frac{1}{2}(M_{\mathbb{O}}(\mathbf{x}^-) + M_{\mathbb{O}}(\mathbf{x}^+)) - 1$, and let $r = \frac{1}{2}(M_{\mathbb{O}}(\mathbf{x}^-) - M_{\mathbb{O}}(\mathbf{x}^+))$.

4. Return $(tb, r)$.

**Theorem 3.3.5.** *Let $(tb, r)$ be the output of Algorithm 3.3.4 when run on input $\mathbb{G}$. Then $tb = tb(\Lambda(\mathbb{G}))$ and $r = r(\Lambda(\mathbb{G}))$.*

*Proof.* This proof follows from definitions presented in [OSS15]. In Step 2, we correctly calculate the Maslov $\mathbb{O}$-gradings of the $\mathbf{x}^+$ and $\mathbf{x}^-$ states, as defined in [OSS15, Equation 4.5]. Next, [OSS15, Theorem 12.3.2] gives us

$$M_{\mathbb{O}}(\mathbf{x}^+) = tb(\Lambda) - r(\Lambda) + 1, \quad M_{\mathbb{O}}(\mathbf{x}^-) = tb(\Lambda) + r(\Lambda) + 1.$$

When we solve this system of equations for $tb$ and $r$, the solutions match the values calculated in Step 3 and returned in Step 4. □

### Organizing by Alexander Polynomial

After bucketing grids by their classical Legendrian invariants, we proceeded to refine our bucketing by sorting by Alexander polynomial. To compute the Alexander polynomial of a knot represented by a grid, we drew upon ideas in [OSS15]. Running this algorithm requires that a computer perform symbolic algebra, which is often a slow task for computers. Therefore, it was only practical to run compute Alexander polynomials once the set of grids was already small.

**Algorithm 3.3.6.** Computing the Alexander Polynomial

On input $\mathbb{G} = (\mathbb{X}, \mathbb{O}) \in (S_{10})^2$:

1. Create an $10 \times 10$ integer array denoted $A$, filled with zeroes. Let $A_{i,j}$ denote the integer in the $i$-th column and the $j$-th row, where the $i = 0$ corresponds to the leftmost column and $j = 0$ corresponds to the bottommmost row. Note that this convention choice matches the permutation description of a grid diagram.

2. Set every entry in the leftmost column of $A$ to zero.

3. For $col \in [9]$:

   a. For $row \in [10]$:

      i. If $\mathbb{X}_{col} < row \leq \mathbb{O}_{col}$, set $A_{col+1,row} = A_{col,row} + 1$

      ii. If $\mathbb{O}_{col} < row \leq \mathbb{X}_{col}$, set $A_{col+1,row+1} = A_{col,row} - 1$

      iii. Otherwise, set $A_{col+1,row} = A_{col,row}$

4. Construct the $10 \times 10$ matrix $\mathbf{M}_{i,j}$, whose entries are polynomials in a formal variable $t$. Set $\mathbf{M}_{i,j} = t^{A_{i,j}}$ (using the same convention for row and column indices as in Step 1).

5. Take the determinant $\det(\mathbf{M}_{i,j})$, which is a polynomial in $t$. Then, divide this polynomial by $\pm(1-t)^9$, multiply it by $\pm t^l$ for some uniquely defined integer $l$, and reduce the result to the form $a_k t^k + a_{k-1} t^{k-1} + ... + a_0$ with $a_0 > 0$. Return the sequence $[a_k, ..., a_0]$.

**Theorem 3.3.7.** *Let $L$ be the output of Algorithm 3.3.6 when run on input $\mathbb{G}$. Then the entries of $L$ are the coefficients of the Alexander polynomial of $\Lambda(\mathbb{G})$.*

*Proof.* This proof follows from definitions presented in [OSS15]. First, recall that [OSS15, Definition 4.5] defines the grid matrix $\mathbf{M}(\mathbb{G})$ as the matrix such that the entry at column $i$, row $j$ is the formal variable $t$, raised to the power of the winding number of $\Lambda(\mathbb{G})$ about the intersection of vertical line $i$ and horizontal line $j$ (which I shall denote $(i, j)$). I now claim that Steps 2, 3, and 4 correctly compute $\mathbf{M}(\mathbb{G})$.

To prove this, it is sufficient to show that after Step 3, each matrix entry $A_{i,j}$ correctly matches the winding number of $\Lambda(\mathbb{G})$ about $(i, j)$. To show this, begin by noticing that the strands of the knot, represented on the grid diagram as vertical and horizontal lines connecting $X$s and $O$s, subdivide the plane of the grid diagram

into regions of constant winding number. Passing from left to right across an upward-oriented strand corresponds to passing from a region of winding number $w$ to a region of winding number $w+1$, and passing from left-to-right across a downward-oriented strand corresponds to passing from a region of winding number $w$ to a region of winding number $w-1$. Furthermore, the region outside all of the knot strands clearly has winding number zero. Figure 3.6 illustrates how a grid diagram of the left-handed trefoil divides the plane into such regions.

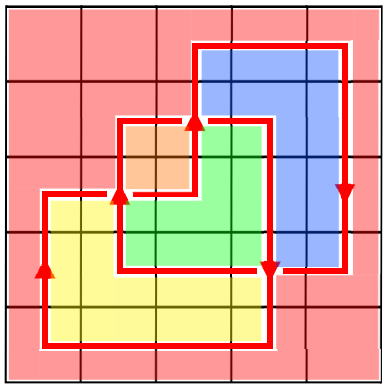

Figure 3.6: This grid diagram divides the plane into five regions of constant winding number. Points inside the red region have winding number $w = 0$; inside the orange, yellow, or blue regions, $w = 1$; and inside the green region, $w = 2$.

We now prove the correctness of the matrix entries by induction on the columns of $A$. Notice that the entries in the leftmost column are all set to zero, because these entries correspond to the intersection points outside the knot strands. Next, notice that $A_{i+1,j} = A_{i,j} + 1$ whenever $\mathbb{X}_i < j \leq \mathbb{O}_i$. Since $X$s connect to $O$s vertically, this means that the regions containing intersection points $(i, j)$ and $(i+1, j)$ are separated by a single upward-oriented strand, and therefore the winding number at $(i + 1, j)$ must be exactly one greater than the winding number at $(i, j)$. By similar arguments, we see that $A_{i+1,j} = A_{i,j} - 1$ whenever $(i, j)$ and $(i + 1, j)$ are separated by a single downward-oriented strand, and $A_{i+1,j} = A_{i,j}$ whenever $(i, j)$ and $(i + 1, j)$ lie in the same region on the plane. Thus, if the entries in column $i$ correctly match the winding

numbers at the intersection points, then the entries in column $i+1$ correctly match as well. But we know that the entries in the zeroth column are correct, and so the entries in every column are correct as well.

Since $A_{i,j}$ correctly stores the winding numbers at the intersection points, $\mathbf{M}_{i,j}$ as defined in Step 4 is indeed the grid matrix $\mathbf{M}(\mathbb{G})$. Now, by [OSS15, Theorem 3.36], the determinant of the grid matrix divided by $(1-t)^9$ is indeed the Alexander polynomial of $\Lambda(\mathbb{G})$, up to a factor of $\pm t^l$ for some integer $l$. However, this factor does not affect the polynomial coefficients, so the sequence returned in Step 5 is indeed the coefficients of the Alexander polynomial of $\Lambda(\mathbb{G})$. □

### Organizing by Topological Type

After bucketing grids by their Alexander polynomial, we further refined the bucketing by topologically identifying the grids using SnapPy [CDGW] and SageMath [The23]. Since the topological type of a knot determines its Alexander polynomial, bucketing by topological type in theory renders obsolete our efforts to bucket by Alexander polynomial. However, interfacing the SnapPy and SageMath databases with our set of grids was a serious challenge. Therefore, this bucketing was performed chronologically last, and our Alexander polynomial bucketing was a temporary substitute.

To interface SnapPy and SageMath with our set of grids, it was necessary to write a script to determine the PD ('planar diagram') code of a grid diagram. Given a knot diagram, the PD code is determined algorithmically. Starting from an arbitrary point, traverse the knot and label the knot strands in increasing order, incrementing the strand counter every time you pass through a crossing. Then, for each crossing, write down a symbol $X_{ijkl}$, where the indices denote the strands at each crossing, ordered counterclockwise starting from the incoming lower edge. A full PD code is written as $X_{abcd}X_{efgh}...$, with one symbol for each crossing. Figure 3.7 demonstrates how to construct a PD code for a left-handed trefoil.

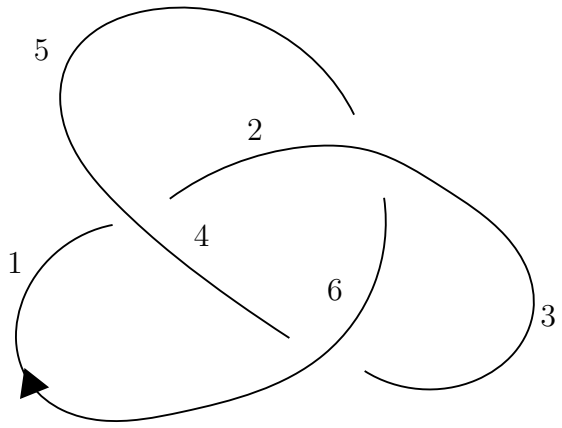


Figure 3.7: A knot diagram of the left-handed trefoil with strands labeled in increasing order starting at the arrow. The PD code of this knot diagram is $X_{1425}X_{5263}X_{3641}$.

These PD codes allowed us to interface our set of grids with various existing knot databases and programs. We began by feeding our PD codes into SnapPy, a program designed to study 3-manifolds [CDGW]. By considering knot exteriors, SnapPy was able to identify the topological type of all but about 200 grids in our set. However, this identification did not distinguish knots from their orientation reversals or their mirror images. Using different functionalities, we were able to SnapPy was also able to determine the symmetry type of all but 6 knots in our set (that is, whether or not a knots is reversible). For these remaining knots, we found their symmetry types manually.

Most recently, we fed our PD codes into SageMath, a mathematics software system that contains the KnotInfo database [The23], a large repository of known knot information. SageMath was able to identify the topological type of every grid in our set (except for a single 17-crossing knot), and was able to distinguish knots from their mirror images.

We would like to thank Chuck Livingston [Liv] for advice on topological identification and computation of symmetry types using SnapPy.

Chapter 4

# Computing Mountain Ranges

This chapter details the computation of mountain ranges and symmetry relations for each grid in the set.

Section 4.1

# Symmetries

Prior to computing the next level of a mountain range, we found that it was helpful to identify the properties of knots in our set under the transformations of orientation reversal and Legendrian mirroring. For example, given a grid $\mathbb{G}_i$ in our set, we might want to know if there exists a grid $\mathbb{G}_j$ in our set such that $\Lambda(\mathbb{G}_i) = -\mu(\Lambda(\mathbb{G}_j))$. To compute these symmetry relations, we implemented the following algorithm.

**Algorithm 4.1.1.** Check Symmetry Relations

On inputs ($\mathbb{G}$, $\mathbb{G}'$, *depth*), where $\mathbb{G} = (\mathbb{X}, \mathbb{O})$ and $\mathbb{G}' = (\mathbb{X}', \mathbb{O}')$ are $n \times n$ grid diagrams representing Legendrian knots which are topologically isotopic and have equal values of the classical invariants $tb$ and $r$, and *depth* is an integer depth parameter:

1. Construct the grid diagram $\mathbb{G}'_{\text{MR}} = (\mathbb{X}'_{\text{MR}}, \mathbb{O}'_{\text{MR}})$ as follows:

$$(\mathbb{X}'_{\text{MR}})_i = n - \mathbb{O}'_{n-i} \quad (\mathbb{O}'_{\text{MR}})_i = n - \mathbb{X}'_{n-i}$$

2. If $r(\Lambda(\mathbb{G})) = 0$, construct the grid diagram $\mathbb{G}'_{\text{M}} = (\mathbb{X}'_{\text{M}}, \mathbb{O}'_{\text{M}})$ as follows:

$$(\mathbb{X}'_{\text{M}})_i = n - \mathbb{X}'_{n-i} \quad (\mathbb{O}'_{\text{M}})_i = n - \mathbb{O}'_{n-i}$$

   Then, construct the grid diagram $\mathbb{G}'_{\text{R}} = (\mathbb{O}', \mathbb{X}')$.

3. Compute a set of equivalent grids $\mathcal{S}_{\text{MR}}$ by running Algorithm 3.2.5 on input $(\{\mathbb{G}'_{\text{MR}}\}, \textit{depth})$. If $r(\Lambda(\mathbb{G})) = 0$, generate $\mathcal{S}_{\text{M}}$ and $\mathcal{S}_{\text{R}}$ similarly. Finally, compute a set of grids $\mathcal{S}$ that are equivalent to $\mathbb{G}$ by running Algorithm 3.2.5

on input $(\{\mathbb{G}\}, \textit{depth})$.

4. Let $B_{\text{MR}}$ be a boolean variable (taking value 0 or 1) indicating whether or not there is at least one grid in the intersection of $\mathcal{S}$ and $\mathcal{S}_{\text{MR}}$. Define $B_{\text{M}}, B_{\text{R}}$ similarly. (If $r(\Lambda(\mathbb{G})) \neq 0$, let $B_{\text{M}}, B_{\text{R}}$ both be false.) Then return the triple $(B_{\text{MR}}, B_{\text{M}}, B_{\text{R}})$.

**Theorem 4.1.2.** *Let $(B_{\text{MR}}, B_{\text{M}}, B_{\text{R}})$ be the outputs of Algorithm 4.1.1 when run on inputs $\mathbb{G}, \mathbb{G}'$. If $B_{\text{MR}} = 1$, then $\Lambda(\mathbb{G}) = -\mu(\Lambda(\mathbb{G}'))$. Similarly, if $B_{\text{M}} = 1$, then $\Lambda(\mathbb{G}) = \mu(\Lambda(\mathbb{G}'))$; and if $B_{\text{R}} = 1$, then $\Lambda(\mathbb{G}) = -\Lambda(\mathbb{G}')$.*

*Proof.* First, notice that the grid $\mathbb{G}'_{\text{MR}}$ describes a grid that is the 180 degree rotation of $\mathbb{G}$ with $X$s and $O$s flipped, so $\Lambda(\mathbb{G}'_{\text{MR}}) = -\mu(\Lambda(\mathbb{G}'))$. Consequently, by Theorem 3.2.6, every grid in $\mathcal{S}_{\text{MR}}$ represents $-\mu(\Lambda(\mathbb{G}'))$. Similarly, every grid in $\mathcal{S}$ represents $\Lambda(\mathbb{G})$. Now, if $B_{\text{MR}} = 1$, there is an intersection between $\mathcal{S}$ and $\mathcal{S}_{\text{MR}}$, and so there is a grid $\mathbb{G}_{\text{intermediate}}$ in both sets. As a result, $\Lambda(\mathbb{G}) = \Lambda(\mathbb{G}_{\text{intermediate}}) = -\mu(\Lambda(\mathbb{G}'))$, and the theorem holds. Similar reasoning holds for $B_{\text{M}}$ and $B_{\text{R}}$. □

Using certain runtime optimizations (such as not recomputing the sets $\mathcal{S}_{\text{MR}}$ every time the algorithm is run), we were able to efficiently use this algorithm to identify the symmetry relations of many grids in our set. With that said, there is an important caveat: if this algorithm fails to find a symmetry relation between two grids, it does not imply that no such symmetry relation exists.

Section 4.2

# Mountain Ranges

In this section, we turn to the question of finding grids that represent Legendrian knots in subsequent layers of the mountain range of some topological knot type $K$. Naively, we might approach this problem by taking each grid in the lowest layer so far calculated, positively and negatively stabilizing it, and proceeding with reductions from there. The goal of the following algorithm is to improve on the naive method. This algorithm blends our knowledge of Legendrian isotopies, symmetry relations, and stabilizations, to produce an efficient algorithm that generates an already-reduced set of grids in the next layer of the mountain range.

**Algorithm 4.2.1.** Generate Next Layer

On inputs ($\mathcal{S}$,*depth*), where $\mathcal{S}$ is the set of grids in the bottommost layer of the mountain range of some topological knot $K$:

1. Denote the elements of $\mathcal{S}$ as $\mathbb{G}_0, ..., \mathbb{G}_{n-1}$, and define $\Lambda_i = \Lambda(\mathbb{G}_i)$. Suppose that $\mathcal{S}$ is subdivided into buckets $B_1, ...B_m$, where the grids in each bucket all share the same value of $tb$ and $r$. Within each bucket, use Algorithm 4.1.1 to compute the symmetry relations between grids in the same bucket.

2. Use a variant of Algorithm 3.2.9 to positively and negatively stabilize each grid in each bucket. Denote the sets of grids returned by this algorithm as $S_0, ..., S_{2n-1}$, where $S_i$ and $S_{i+n}$ are the are sets full of positive and negative stabilizations of $\mathbb{G}_i$, respectively.

3. Expand each set $S_i$ by running Algorithm 3.2.5 on inputs ($S_i$,*depth*). Then, for each set $S_i$, construct the set $S_i^{\mathrm{R}}$ by taking the orientation reversal of

each element in $S_i$. In a similar manner, construct sets $S_i^{\mathrm{M}}$ and $S_i^{\mathrm{MR}}$ by taking Legendrian mirrors and orientation reversals of Legendrian mirrors.

4. Create an undirected graph $G = (V, E)$, where the $8n$ vertices are the knots $S_\pm(\Lambda_i)$, $-S_\pm(\Lambda_i)$, $\mu(S_\pm(\Lambda_i))$, and $-\mu(S_\pm(\Lambda_i))$.

5. Add edges to the graph as follows. For each pair $(i, j) \in [2n]^2$:

   a. If $S_i \cap S_j$ is nonempty, we add edges to our graph as follows:

      i. Find the knot $\Lambda_{i'}$ such that grids in $S_i$ represent $S_\pm(\Lambda_{i'})$. Then, using the stabilization data computer in Step 1, construct the following set:

      $$L = \{S_\pm(\Lambda_{i'}), -S_\mp(-\Lambda_{i'}), \mu(S_\mp(\mu(\Lambda_{i'}))), -\mu(S_\pm(-\mu(\Lambda_{i'})))\}$$

      Notice that by Theorem 2.2.5, every knot in $L$ is isotopic.

      ii. Construct a set $R$ similarly, using the knot $\Lambda_{j'}$ such that grids in $S_j$ represent $S_\pm(\Lambda_{j'})$. Note that Theorem 2.2.5 implies that every knot in $R$ is isotopic.

      iii. For every pair of stabilized Legendrian knots in $L \times R$, denoted $(S\Lambda_l, S\Lambda_r)$, add to our graph $G$ an edge connecting $S\Lambda_l$ to $S\Lambda_r$. Then, add three more edges, connecting $-S\Lambda_l$ to $-S\Lambda_r$, $\mu(S\Lambda_l)$ to $\mu(S\Lambda_r)$, and $-\mu(S\Lambda_l)$ to $-\mu(S\Lambda_r)$.

   b. If $S_i \cap S_j^{\mathrm{M}}$ is nonempty, we perform a similar procedure, eventually adding edges to our graph. Do the same for $S_j^{\mathrm{R}}$ and $S_j^{\mathrm{MR}}$.

6. Initialize an empty set of stabilized grid diagrams $\mathcal{S}'$. Then each connected component in the graph $G$ that contains a knot of the form $S_\pm(\Lambda_i)$, add a

single grid representative of $S_\pm(\Lambda_i)$ to $\mathcal{S}'$.

7. Return $\mathcal{S}'$.

This algorithm seems intimidating, so let's break it down piece by piece. Step 1 collects symmetry data. Steps 2 and 3 compute sets of grids that are equivalent to stabilizations of grids in $\mathcal{S}$, and then generates additional sets by performing symmetry transformations on those grids. Step 4 initializes a graph $G$ that stores known isotopies between stabilized grids and the symmetry-transformed versions of those stabilized grids. Step 5 searches for isotopies by comparing sets of grids; this step also exploits symmetry data and properties of stabilizations to maximize the number of recorded isotopy equivalence relations per isotopy discovered. At the end of this step, the graph $G$ represents a web of isotopy equivalences, where all knots in a single connected component of $G$ are Legendrian isotopic. At this point, Step 6 finds each of these connected components, and chooses a single grid that represents every knot in that component. It is this small set of knots that is returned in Step 7. We now prove that this algorithm does in fact return a set of grids where at least one grid represents each knot in the next layer of the mountain range.

**Theorem 4.2.2.** *Let $\mathcal{S}'$ be the output of Algorithm 4.2.1 when run on inputs ($\mathcal{S}$,depth). For each distinct Legendrian knot of the form $S_\pm(\Lambda(\mathbb{G}))$ where $\mathbb{G} \in \mathcal{S}$, there is at least one grid $\mathbb{G}' \in \mathcal{S}'$ such that $\Lambda(\mathbb{G}') = S_\pm(\Lambda(\mathbb{G}))$.*

*Proof.* To prove this, we begin by showing that the knots in each connected component of $G$ are Legendrian isotopic. To show this, it is sufficient to prove that there is an edge between knots only if those knots are Legendrian isotopic. First, notice that all knots in the sets $L$ as constructed in Step 5ai are Legendrian isotopic, which follows from Theorem 2.2.5. Similarly, all knots in the set $R$ are Legendrian isotopic. Next, since $S_i \cap S_j$, the knots represented by $S_i$ and $S_j$ are Legendrian isotopic. Then, by transitivity, all knots in both $L$ and $R$ are Legendrian isotopic. This implies that

every edge added to $G$ in Step 5aiii connects a pair of Legendrian isotopic knots. Similar logic shows that every edge added to $G$ in Step 5b also connects Legendrian isotopic knots.

Finally, since Step 6 selects one element from each connected component of $G$ that contains a knot of the form $S_{\pm}(\mathbb{G})$ for $\mathbb{G}$, and all knots in each connected component are Legendrian isotopic, we conclude that $\mathcal{S}'$ does indeed contain at least one grid representing every knot of the form $S_{\pm}(\mathbb{G})$ for $\mathbb{G} \in \mathcal{S}$. □

# Chapter 5

# Results

In this chapter, we present our results. The first section of this chapter provides a summary of our results: How many grids did we find at each layer of the mountain range, and what algorithms did we run to obtain those grids? The second section provides a close look at our results for a single topological knot type. The third section provides three tables containing our complete results.

Section 5.1

# An Overview of Results

### A Set of Grid Diagrams

We identified 3082 non-destabilizable Legendrian knots of grid size 10. We were able to subdivide these grids into 819 buckets, organized by the classical invariants $tb$ and $r$ and by topological type. Table 5.1 details the algorithms that we ran to obtain these grids.

Table 5.1: A list of the algorithms used to find the set of 3082 non-destabilizable Legendrian knots of grid size 10, as well as the number of grids and the number of buckets at each step.

| Actions Performed | Number of Grids | Number of Buckets |
|---|---|---|
| Generate a set of grids and remove links, as described in Section 3.1. | 89,763,984 | 1 |
| Remove grids by running Algorithm 3.2.7 with depth parameter $depth = 1$ | 5,881,569 | 1 |
| Remove grids by running Algorithm 3.2.7 with depth parameter $depth = 2$ | 2,129,118 | 1 |
| Remove grids by running Algorithm 3.2.7 with depth parameter $depth = 3$. | 1,375,597 | 1 |

| | | |
|---|---|---|
| Remove grids by running Algorithm 3.2.7 with depth parameter $depth = 4$. | 145,831 | 1 |
| Remove grids by running Algorithm 3.2.7 with depth parameter $depth = 5$. | 21,338 | 1 |
| Place grids into buckets according to their classical invariants $tb$ and $r$, calculated via Algorithm 3.3.4. | 21,338 | 108 |
| Remove grids by running Algorithm 3.2.7 with depth parameter $depth = 10$. | 15,777 | 108 |
| Remove grids by running Algorithm 3.2.7 with depth parameter $depth = 20$. | 13,633 | 108 |
| Remove grids by running Algorithm 3.2.7 with depth parameter $depth = 40$. | 13,306 | 108 |
| Split grids into smaller buckets according to their Alexander polynomial (in addition to their classical invariants), calculated using Algorithm 3.3.6. | 13,306 | 814 |
| Remove grids by running Algorithm 3.2.11 with depth parameter $depth = 4$. | 8,501 | 814 |

| | | |
|---|---|---|
| Remove grids by running Algorithm 3.2.11 with depth parameter $depth = 8$. | 4,855 | 814 |
| Remove grids by running Algorithm 3.2.11 with depth parameter $depth = 16$. | 3,568 | 814 |
| Split grids again into smaller buckets by computing their topological type with SnapPy, as described in Section 3.3. | 3,568 | 862 |
| Remove grids by running Algorithm 3.2.11 with depth parameter $depth = 64$. | 3,155 | 862 |
| Split grids again into smaller buckets by computing their topological type with SageMath, as described in Section 3.3. Then remove a few remaining knots of arc index less than 10, as well as a number of connect sums. | 3,082 | 819 |

Prior to bucketing by the classical invariants, all grids were theoretically in the same bucket. However, to improve runtime, these grids were often arbitrarily placed into smaller buckets of approximate size 1000, and then run in parallel. The only consequence of this bucketing was to decrease the number of grids removed at each step, because isotopies could not be discovered between grids in different buckets. Once we switched to bucketing by knot invariants, this no longer became an issue.

### Mountain Ranges

We computed the first three layers of the mountain range for each topological knot types among our set non-destabilizable Legendrian knots. The first layer of mountain ranges consisted of the 3082 knots we identified as non-destabilizable, organized by topological type, Alexander polynomial, and the classical Legendrian invariants into 819 buckets. The grids in this layer were computed using the algorithms listed in Table 5.1. The second layer of mountain ranges consisted of 2706 Legendrian knots, organized by the same properties into 1298 buckets. The grids in this layer were computed using the algorithms listed in Table 5.2. Finally, the third of layer of mountain ranges consisted of 3648 Legendrian knots, organized by the same properties into 1764 buckets. The grids in this layer were computed using the algorithms listed in Table 5.3.

Table 5.2: A list of the algorithms used to find the set of 2820 Legendrian knots in the second layer of mountain ranges, as well as the number of grids and the number of buckets at each step.

| Actions Performed | Number of Grids | Number of Buckets |
|---|---|---|
| Generate a set of stabilized grids by positively and negatively stabilizing each of the non-destabilizable grids. | 6330 | 1362 |
| Remove grids by running Algorithm 4.2.1 with depth parameter $depth = 15$?. | 2894 | 1362 |

| | | |
|---|---|---|
| Remove grids by running Algorithm 3.2.7 with depth parameter $depth = 100$ and skipping Step 3 (which would remove destabilizable grids). | 2826 | 1362 |
| Remove grids by running Algorithm 3.2.11 with depth parameter $depth = 8$. | 2820 | 1362 |
| Remove grids that are the stabilizations of knots that were identified by SageMath as having arc index less than 10 or as being connect sums. | 2706 | 1298 |

Table 5.3: A list of the algorithms used to find the set of 3814 Legendrian knots in the third layer of mountain ranges, as well as the number of grids and the number of buckets at each step.

| Actions Performed | Number of Grids | Number of Buckets |
|---|---|---|
| Generate a set of twice-stabilized grids by positively and negatively stabilizing each of the grids in the second layer of mountain ranges. | 5640 | 1849 |

| | | |
|---|---|---|
| Remove grids by running Algorithm 4.2.1 with depth parameter $depth = 12$. | 3836 | 1849 |
| Remove grids by running Algorithm 3.2.7 with depth parameter $depth = 40$, skipping step 3 (which would remove destbilizable grids). To handle memory constraints, we terminated calls to Algorithm 3.2.5 whenever more than 500,000 grids were simultaneously added to a bubble. | 3814 | 1849 |
| Remove grids that are the stabilizations of knots that were identified by SageMath as having arc index less than 10 or as being connect sums. | 3648 | 1764 |

In general, the mountain ranges we computed contain more Legendrian knots per bucket than the mountain ranges of the Legendrian knot atlas [CN13]. The authors of the Legendrian knot atlas found that specifying the topological knot type, Thurston-Benneqin number, and rotation number often completely specified a Legendrian knot. In contrast, we typically found two to three Legendrian knots meeting those criteria. For example, in the topmost layer of mountain ranges, we found on average 3.76 Legendrian knots per bucket, with some buckets containing as many as 17 distinct knots! [1] Things simplified slightly in the second and third layers, with only 2.08 and

[1] This is for the bucket corresponding to the topological type $10n21$, with $tb = -9$ and $r = 0$.

2.07 knots per bucket respectively.

For now, it is unclear whether this additional complexity is a feature of the larger grid sizes, or whether it is an artifact of the search algorithms employed to find isotopies. Our primary search algorithm, the bubble-generation algorithm Algorithm 3.2.5, is very time- and memory-intensive, especially when run on grids larger than size 11. As a result, we never searched for isotopies containing more than one stabilization and destabilization move, which means that our list of 3082 non-destabilizable grids may contain numerous duplicates. It is even possible that each of our 819 buckets only contains one unique knot, and the remaining grids are duplicates! On the other hand, it is also possible that these mountain ranges simply do get more complex with increasing grid size. There are two possible explanations for this behavior.

(a) Our bucketing is not optimal. Both SageMath and SnapPy 'forget' the orientation of a knot when identifying its topological type. Therefore, our bucketing scheme places Legendrian representatives of these non-invertible knots into the same bucket, even though an isotopy may not exist between them. As a result, an improved bucketing scheme would contain more buckets, and therefore fewer Legendrian representatives per bucket.

(b) It is possible that complexities in the mountain ranges are 'real' through one or two stabilizations, but resolve themselves further down the mountain range. We know that every two Legendrian representatives of the same topological knot type are equal after some number of positive and/or negative stabilizations, but we have no indication of how many stabilizations that might take.

Section 5.2

## Case Study

The following section presents our results for the arbitrarily-chosen knot identified as $m10n20$. (This notation is interpreted as the mirror image of the 20th non-alternating 10-crossing knot in the Rolfsen Knot Table. In Alexander-Briggs notation, this knot would be labeled $m10_{143}$.) This knot is a 10-crossing reversible knot (that is, an invertible knot that is not isotopic to its mirror image). Its symmetrized Alexander polynomial is

$$\Delta_{m10n20}(t) = t^3 - 3t^2 + 6t - 7 + 6t^{-1} - 3t^{-2} + t^{-3}.$$

The grid diagram $\mathbb{G} = (\mathbb{X}, \mathbb{O})$ with

$$\mathbb{X} = (0, 5, 9, 8, 6, 7, 4, 2, 3, 1), \quad \mathbb{O} = (2, 8, 7, 1, 0, 3, 9, 5, 6, 4)$$

satisfies $K(\mathbb{G}) = m10n20$. Figure 5.1 illustrates this grid diagram and the corresponding knot diagram.

While running the algorithms described in Section 5.1, we collected information on the topmost, second, and third layers of the mountain range of this knot. The data collected for each of these layers are contained Table 5.4, Table 5.5, and Table 5.6 respectively. Note that these tables are excerpted from the complete results as presented in Section 5.3, and written using familiar mathematical notation. Table 5.4 indicates that we found 14 grid diagrams corresponding to non-destabilizable Legendrian representatives, organized into 4 buckets. Table 5.5 indicates that we found 7 grid diagrams in the second layer of the mountain range, organized into 5 buckets. Finally, Table 5.6 indicates that we found 8 grid diagrams in the third layer

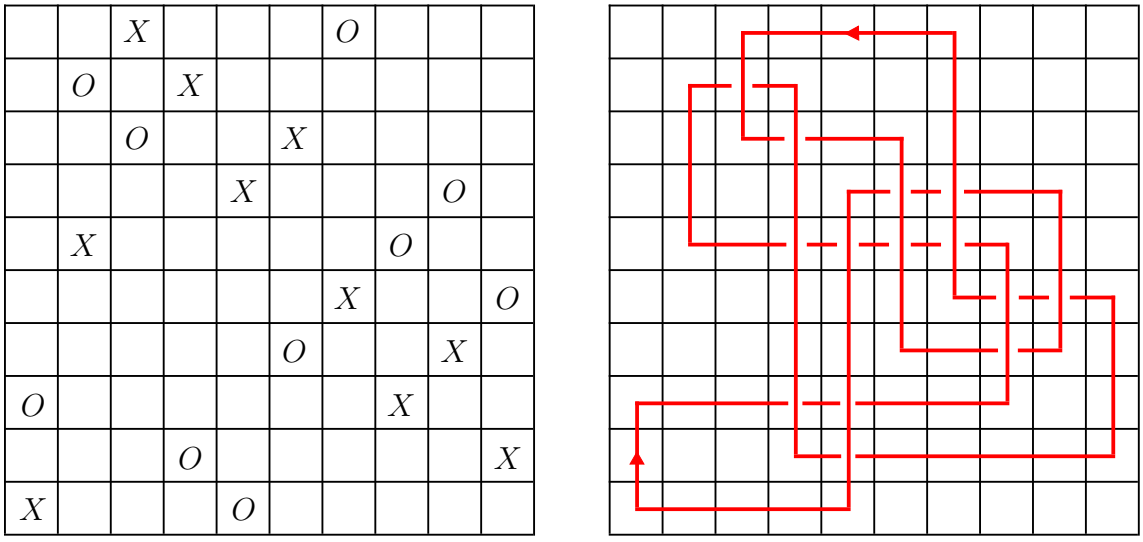


Figure 5.1: Left: A grid diagram, described by permutations $\mathbb{X} = (0, 5, 9, 8, 6, 7, 4, 2, 3, 1)$ and $\mathbb{O} = (2, 8, 7, 1, 0, 3, 9, 5, 6, 4)$, which specifies the knot $10_{143}$. Right: The corresponding oriented knot diagram.

of the mountain range, organized into 6 buckets. Figure 5.1 compactly displays this information in a mountain range.

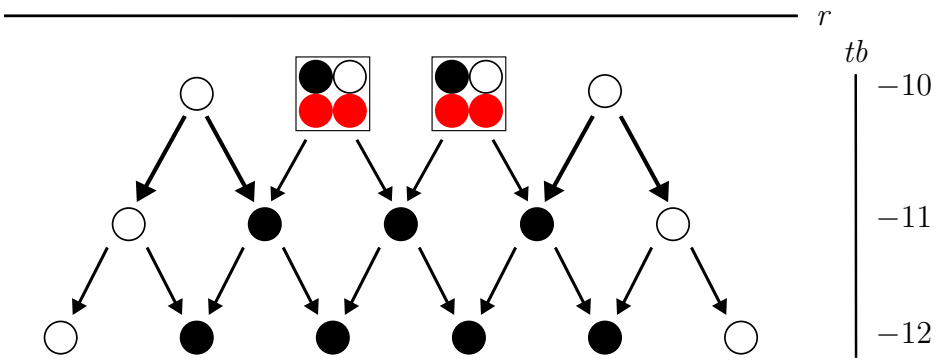


Figure 5.2: The first three layers of the mountain range of the knot $m10n20$. This diagram compactly summarizes the information in Table 5.4, Table 5.5, and Table 5.6.

Readers familiar with the Legendrian Knot Atlas might notice that our mountain range conventions are slightly different than theirs. Our mountain range conventions were chosen to compactly display information about the symmetry relations between grids in buckets. In general, black dots correspond to grids whose symmetry information is known, while red dots correspond to grids whose symmetry information

Table 5.4: The raw data for the grids in the topmost layer of the mountain range.

| Grid ID | Topological Type | $tb$ | $r$ | $\mathbb{X}$ | $\mathbb{O}$ | $-\Lambda$ | $\mu(\Lambda)$ | $-\mu(\Lambda)$ | Parents |
|---|---|---|---|---|---|---|---|---|---|
| $G_{133}$ | $m10n20$ | $-10$ | $-1$ | $(0,5,9,8,6,7,4,2,3,1)$ | $(2,8,7,1,0,3,9,5,6,4)$ | – | – | $G_{135}$ | – |
| $G_{134}$ | $m10n20$ | $-10$ | $-1$ | $(0,1,9,4,3,8,6,5,7,2)$ | $(3,6,2,7,5,4,0,9,1,8)$ | – | – | ? | – |
| $G_{135}$ | $m10n20$ | $-10$ | $-1$ | $(0,8,7,5,4,2,3,1,6,9)$ | $(7,6,1,9,8,5,0,4,2,3)$ | – | – | $G_{133}$ | – |
| $G_{136}$ | $m10n20$ | $-10$ | $-1$ | $(0,8,2,5,4,3,1,7,6,9)$ | $(7,1,9,0,6,8,5,4,2,3)$ | – | – | ? | – |
| $G_{137}$ | $m10n20$ | $-10$ | $-1$ | $(0,6,4,5,3,9,7,8,2,1)$ | $(2,1,9,0,8,6,4,5,7,3)$ | – | – | $G_{137}$ | – |
| $G_{138}$ | $m10n20$ | $-10$ | $-3$ | $(0,8,2,7,6,3,5,4,1,9)$ | $(4,1,9,0,8,7,2,6,5,3)$ | – | – | $G_{139}$ | – |
| $G_{139}$ | $m10n20$ | $-10$ | $-3$ | $(0,8,6,7,5,9,2,4,3,1)$ | $(2,1,9,0,8,3,6,7,5,4)$ | – | – | $G_{138}$ | – |
| $G_{140}$ | $m10n20$ | $-10$ | $1$ | $(0,8,4,3,7,6,9,5,1,2)$ | $(7,6,1,5,4,2,3,0,8,9)$ | – | – | $G_{142}$ | – |
| $G_{141}$ | $m10n20$ | $-10$ | $1$ | $(0,8,6,7,9,5,4,3,1,2)$ | $(5,1,9,0,4,3,2,8,6,7)$ | – | – | $G_{141}$ | – |
| $G_{142}$ | $m10n20$ | $-10$ | $1$ | $(5,1,9,0,4,3,2,8,6,7)$ | $(6,4,5,3,8,1,2,0,9,7)$ | – | – | $G_{140}$ | – |
| $G_{143}$ | $m10n20$ | $-10$ | $1$ | $(0,9,4,3,2,5,1,7,8,6)$ | $(7,3,1,0,8,9,6,4,5,2)$ | – | – | ? | – |
| $G_{144}$ | $m10n20$ | $-10$ | $1$ | $(0,6,5,7,4,9,2,8,3,1)$ | $(4,2,1,3,8,6,7,5,0,9)$ | – | – | ? | – |
| $G_{145}$ | $m10n20$ | $-10$ | $3$ | $(0,8,7,5,4,2,3,1,6,9)$ | $(7,6,4,3,1,9,0,8,2,5)$ | – | – | $G_{146}$ | – |
| $G_{146}$ | $m10n20$ | $-10$ | $3$ | $(0,3,8,7,6,4,2,5,1,9)$ | $(0,3,8,7,6,4,2,5,1,9)$ | – | – | $G_{145}$ | – |

Table 5.5: The raw data for the grids in the second layer of the mountain range. For any particular grid labeled $S_i$, the parents column indicates the grids in the topmost layer which may be stabilized to yield $\Lambda(S_i)$. Whether those stabilizations are positive or negative may be determined by examining the rotation numbers of all grids involved.

| Grid ID | Topological Type | $tb$ | $r$ | $-\Lambda$ | $\mu(\Lambda)$ | $-\mu(\Lambda)$ | Parents |
|---|---|---|---|---|---|---|---|
| $S_{110}$ | $m10n20$ | $-11$ | $-2$ | – | – | $S_{110}$ | $G_{133}$, $G_{134}$, $G_{135}$, $G_{136}$, $G_{137}$, $G_{138}$, $G_{139}$ |
| $S_{111}$ | $m10n20$ | $-11$ | $-4$ | – | – | $S_{112}$ | $G_{138}$ |
| $S_{112}$ | $m10n20$ | $-11$ | $-4$ | – | – | $S_{111}$ | $G_{139}$ |
| $S_{113}$ | $m10n20$ | $-11$ | 0 | $S_{113}$ | $S_{113}$ | $S_{113}$ | $G_{140}$, $G_{141}$, $G_{142}$, $G_{143}$, $G_{144}$, $G_{133}$, $G_{134}$, $G_{135}$, $G_{136}$, $G_{137}$ |
| $S_{114}$ | $m10n20$ | $-11$ | 2 | – | – | $S_{114}$ | $G_{145}$, $G_{146}$, $G_{140}$, $G_{141}$, $G_{142}$, $G_{143}$, $G_{144}$ |
| $S_{115}$ | $m10n20$ | $-11$ | 4 | – | – | $S_{116}$ | $G_{146}$ |
| $S_{116}$ | $m10n20$ | $-11$ | 4 | – | – | $S_{115}$ | $G_{145}$ |

Table 5.6: The raw data for the grids in the third layer of the mountain range. For any particular grid labeled $T_i$, the parents column indicates the grids in the second layer which may be stabilized to yield $\Lambda(T_i)$. Whether those stabilizations are positive or negative may be determined by examining the rotation numbers of all grids involved.

| Grid ID | Topological Type | $tb$ | $r$ | $-\Lambda$ | $\mu(\Lambda)$ | $-\mu(\Lambda)$ | Parents |
|---|---|---|---|---|---|---|---|
| $T_{130}$ | $m10n20$ | $-12$ | $-1$ | – | – | $T_{130}$ | $S_{113}$, $S_{110}$ |
| $T_{131}$ | $m10n20$ | $-12$ | $-3$ | – | – | $T_{131}$ | $S_{110}$, $S_{111}$, $S_{112}$ |
| $T_{132}$ | $m10n20$ | $-12$ | $-5$ | – | – | $T_{133}$ | $S_{111}$ |
| $T_{133}$ | $m10n20$ | $-12$ | $-5$ | – | – | $T_{132}$ | $S_{112}$ |
| $T_{134}$ | $m10n20$ | $-12$ | 1 | – | – | $T_{134}$ | $S_{114}$, $S_{113}$ |
| $T_{135}$ | $m10n20$ | $-12$ | 3 | – | – | $T_{135}$ | $S_{115}$, $S_{116}$, $S_{114}$ |
| $T_{136}$ | $m10n20$ | $-12$ | 5 | – | – | $T_{137}$ | $S_{116}$ |
| $T_{137}$ | $m10n20$ | $-12$ | 5 | – | – | $T_{136}$ | $S_{115}$ |

is unknown. A filled black dot represents a grid whose Legendrian knot is known to be equal to orientation reversal of its Legendrian mirror. An unfilled black dot corresponds to a pair of grids, representing possibly distinct Legendrian knots that are known to be equivalent under Legendrian mirroring plus orientation reversal. A filled red dot corresponds a grid that represents a knot whose symmetry relations are unknown; that knot might be equivalent to the orientation reversal of the Legendrian mirror of itself or of some other grid in the set. Finally, a box around a set of dots indicates that those dots all represent grids in the same bucket. Note that it is possible that within a bucket, every grid in fact represents the same Legendrian knot; in other words, we have not calculated any obstructions to Legendrian isotopies between grids in the same bucket.

Section 5.3

# Complete Results

The complete results of my work are contained in three tables. These tables are all around 3000 lines long, so to save space, they are not included in the document. They may be found at the following GitHub repository: https://github.com/NoahSchwartz1/schwartz-senior-thesis-2023.git.

The first table details the set of Legendrian knots in the first layer of our mountain ranges. The first column of this table labels knots as $G_1$, $G_2$, and so on, for ease of readability. Columns 2 through 6 provide the topological type, $tb$, $r$ of each identified Legendrian knot, as well as two pairs of permutations that describe a grid representing that knot. Columns 7 through 9 describe how each Legendrian knot is related to other knots in the table by various symmetry transformations.

The second and third tables detail the sets of Legendrian knots in the second and third layers of our mountain ranges. The first columns label these knots as $S_1$, $S_2$,... and $T_1$, $T_2$, and so on. Columns 2 through 4 provide the topological type, $tb$, and $r$ of each of these knots. Columns 5 through 7 describe how each of these knots is related to other knots in the table by various symmetry transformations. Finally, column 8 lists the 'parents' of knots in the mountain range.

# Bibliography


[Ass23] Ali Assaf, *Algorithm x in 30 lines!*, URL: https://www.cs.mcgill.ca/~aassaf9/python/algorithm_x.html, 2023.

[CDGW] 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 (05/27/2023).

[CN13] Wutichai Chongchitmate and Lenhard Ng, *An atlas of Legendrian knots*, Exp. Math. **22** (2013), no. 1, 26–37. MR 3038780

[JP10] Gyo Taek Jin and Wang Park, *A tabulation of prime knots up to arc index 11*, Journal of Knot Theory and Its Ramifications **20** (2010).

[Knu00] Donald E. Knuth, *Dancing links*, 2000, cs/0011047.

[Liv] Charles Livingston, Personal communication.

[LM23] Charles Livingston and Allison H. Moore, *Knotinfo: Table of knot invariants*, URL: knotinfo.math.indiana.edu, 2023.

[NOT08] Lenhard Ng, Peter Ozsváth, and Dylan Thurston, *Transverse knots distinguished by knot Floer homology*, J. Symplectic Geom. **6** (2008), no. 4, 461–490. MR 2471100

[OSS15] Peter S. Ozsváth, András I. Stipsicz, and Zoltán Szabó, *Grid homology for knots and links*, Mathematical Surveys and Monographs, vol. 208, American Mathematical Society, Providence, RI, 2015. MR 3381987

[sym23] *The knot atlas: Three dimensional invariants*, URL:http://katlas.org/wiki/Three_Dimensional_Invariants, 2023.

[The23] The Sage Developers, *Sagemath, the Sage Mathematics Software System (Version 9.8)*, 2023, https://www.sagemath.org.