Proceedings of GREAT Day

Volume 2020 Article 10

2021

Space Efficient Knot Mosaics for Prime Knots with Crossing Number 10 and Less

James Canning
SUNY Geneseo

Space Efficient Knot Mosaics for Prime Knots with Crossing Number 10 and Less

Erratum

Sponsored by Aaron Heap and Doug Baldwin

Space Efficient Knot Mosaics for Prime Knots with Crossing Number 10 and Less

James Canning

sponsored by Aaron Heap and Doug Baldwin

Abstract

The study of knot mosaics is based upon representing knot diagrams using a set of tiles on a square grid. This relatively new branch of knot theory has many unanswered questions, especially regarding the efficiency with which we draw knots as mosaics. While any knot or link can be displayed as a mosaic, it is still unknown how large the mosaic needs to be, and how many tiles are needed for every knot. In this paper we implement an algorithmic programming approach to find the tile and mosaic number of all knots with crossing number 10 and less. We also introduce an online tool in which users can search, create, and identify knot mosaics.

While knot theory has roots dating back to the 18th century, the study of knot mosaics is a rather new branch of this field that was introduced in the late 2000s. A knot mosaic is a two-dimensional representation of a knot, made up of a finite set of tiles arranged onto a square grid. All of the possible tiles are shown in Figure 4. Knot mosaics are helpful because they provide a uniform way to organize a knot diagram, and they can be easily represented as matrices. By assigning a number to each tile, any $n \times n$ knot mosaic can be represented as an $n \times n$ matrix, where the i, j th entry of the matrix is the number corresponding to the tile in the i th row and j th column of the knot mosaic, as shown in Figure 3.

Published by KnightScholar, 2021

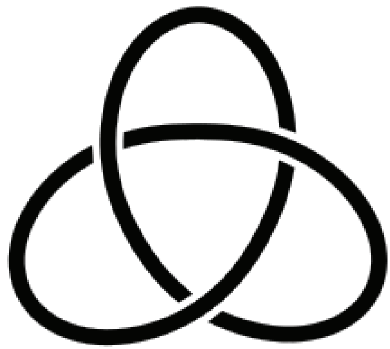

Figure 1: The Trefoil knot projection

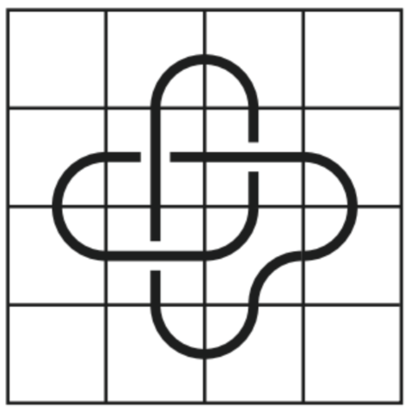

Figure 2: The Trefoil knot as a mosaic

$$\begin{pmatrix} 0 & 2 & 1 & 0 \\ 2 & 10 & 9 & 1 \\ 3 & 9 & 8 & 4 \\ 0 & 3 & 4 & 0 \end{pmatrix}$$

Figure 3: Matrix representation of the Trefoil knot mosaic

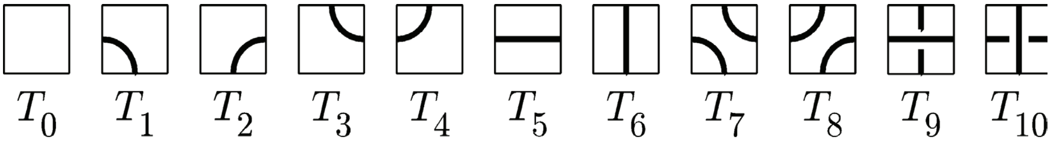

Figure 3: Matrix representation of the Trefoil knot mosaic

By representing a knot as a matrix, we can use all the wonderful and widely available tools for working with matrices on knots. Furthermore, computers can store matrices more efficiently than they can store images of knot diagrams, and so knot mosaics allow for simpler integration with programming.

1.1 Definitions

We will begin by defining some general terms in knot theory.

Definition 1.1

> *A* knot *is a simple closed curve in* S^3. *A* link *is the nonempty union of a finite number of knots.*

Definition 1.2

> *A* connected sum *of two knots is a knot formed by removing a segment (1-ball) inside each knot and attaching the resulting boundary points (1-spheres).*

Definition 1.3

> *A* composite knot *is a knot that is the connected sum of two non-trivial knots. A* prime knot *is any knot that is not composite.*

Definition 1.4

> *A* knot projection *or* knot diagram *is a projection of a knot into a plane with arcs at crossings drawn as over-crossings and under-crossings. A* link diagram *is a similar projection of a link.*

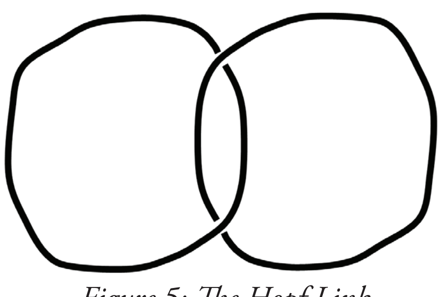

Figure 5: The Hopf Link

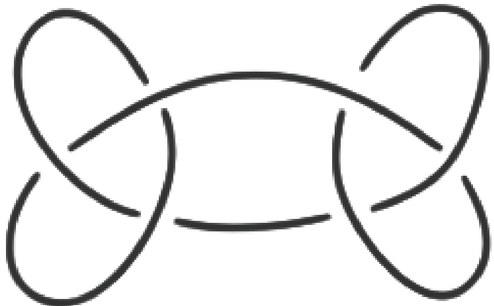

Figure 6: A composite knot projection

A link is simply two or more knots in the same space. Connected sums are created by "cutting" two knots and "gluing" them together at the cuts. The composite knot in Figure 6 is the connected sum of two Trefoil Knots. In this paper, we are interested only in prime knots.

There are an infinite number of ways to draw a knot as a projection. Knot theory is often concerned with determining whether two knots are unique, for which we need to define equivalence.

Definition 1.5

Two knots K_1 and K_2 are said to be knot equivalent *if there is a continuous function, $i : [0,1] \times S^3 \to S^3$ such that*

$i(0, S^3)$ is the identity function

$i(1, K_1) = K_2$

For all $t \in [0,1]$, $i(t, S^3)$ is a homeomorphism to S^3.

We call i *an ambient isotopy.*

An ambient isotopy can be thought of as a continuous deformation of a knot, where stretching and reshaping is allowed, but cutting or breaking the knot is not allowed. When we say that two knots are the same, we mean they are knot equivalent.

Now that we can determine knot equivalence, we need to be able to classify and categorize knots. This is done with a crossing number.

Definition 1.6

The crossing number *of a knot is the fewest number of crossings needed to represent the knot as a projection.*

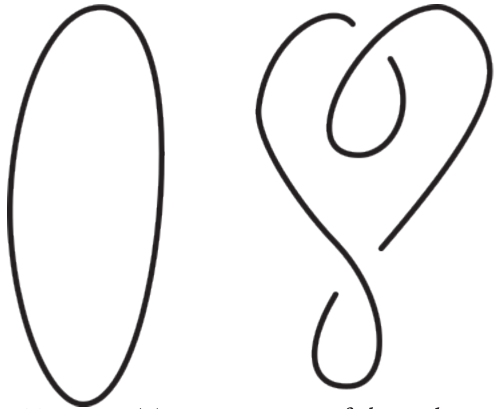

Figure 7: Two projections of the unknot

The unknot, or trivial knot, has crossing number zero. Figure 7 shows two projections of the unknot, one with the crossing number realized. Although the projections are different, the two knots shown are knot equivalent, as there is an ambient isotopy from one to the other. The trefoil knot is the only prime knot with crossing number three.

Having defined the basics of knot theory, we now provide a more formal definition of knot mosaics. Recall that a mosaic tile refers to one of the eleven tiles listed in Figure 4.

Definition 1.7

A connection point *of a mosaic tile is a midpoint of a tile edge that is also the endpoint of a curve drawn on the tile.*

Definition 1.8

A mosaic tile is suitably connected *if each of its connection points touches a connection point of an adjacent tile.*

Definition 1.9

An $n \times n$ mosaic*, or* n-mosaic*, is an* $n \times n$ *matrix whose entries are suitably connected mosaic tiles.*

As an example, Figure 8 shows a matrix with a tile that is clearly not suitably connected, and thus the matrix is not a knot mosaic. By changing that one tile to a tile that is suitably connected, we create a knot mosaic depicting the 4_1 knot, also known as the Figure Eight knot, in Figure 9.

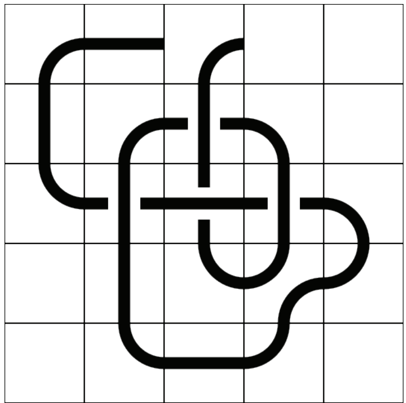

Figure 8: Not Suitably Connected

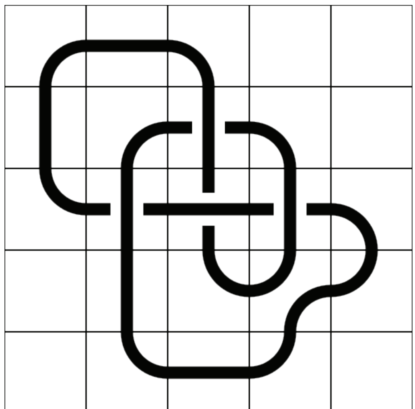

Figure 9: The 4_1 knot

There are many open questions regarding the efficiency with which we can create knot mosaics, including how small of a matrix we can draw a knot on, and how many non-blank tiles we need to use. To rigorously describe these problems we must introduce a few more definitions.

Definition 1.10

The mosaic number *of a knot is the smallest integer* n *such that the knot can be drawn as an* n*-mosaic.*

Definition 1.11

The tile number *of a knot is the fewest number of non-blank tiles that are needed to represent that knot.*

Definition 1.12

The minimal-mosaic tile number *of a knot is the minimum number of non-blank tiles needed to represent the knot as an n-mosaic, where* n *is its mosaic number.*

In Figure 2 we show the trefoil knot as a 4-mosaic. This is the smallest grid that this knot can fit on, and so the trefoil knot has a mosaic number of 4. Additionally, this mosaic uses the fewest number of non-blank tiles as possible. Therefore the trefoil knot has tile number and minimal-mosaic tile number 12.

We make the distinction between tile number and minimal mosaic tile number because some knots with mosaic number 6 can be represented as a 7-mosaic using fewer tiles. For example, the 9_{16} has mosaic number 6 shown in Figure 10 (Heap & Knowles, 2019). This mosaic uses 32 non-blank tiles, and thus the 9_{16} knot has minimal-mosaic tile number 32. However, the 9_{16} knot can also be drawn as a 7-mosaic using only 29 non-blank tiles, as shown in Figure 11. We will prove that this is the fewest number of non-blank tiles needed to represent this knot, and therefore the 9_{16} knot has tile number 29.

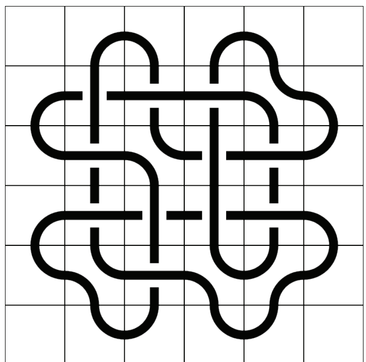

Figure 10: 9_{16} knot with minimal-mosaic tile number realized

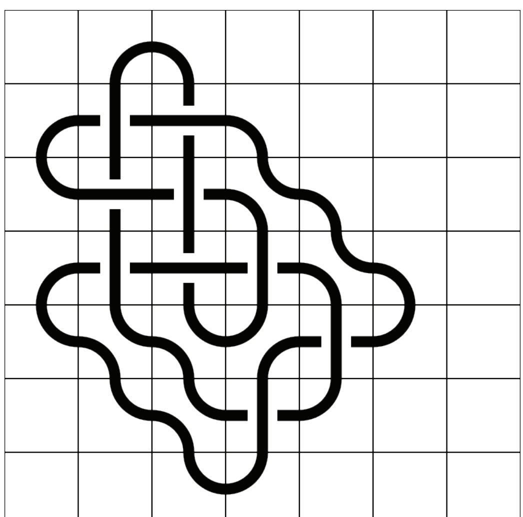

Figure 11: 9_{16} knot with tile number realized

Definition 1.13

> *A* space-efficient n-mosaic *is an* n-*mosaic in which the number of tiles used is minimized for a mosaic of size* n.

That is, an n-mosaic is space efficient if the knot shown cannot be drawn using fewer tiles on an $n \times n$ grid.

Definition 1.14

> *A* minimally space-efficient *mosaic is a space-efficient mosaic for which the mosaic number is realized.*

The mosaic shown in Figure 9 is not space-efficient, as it has unnecessarily large loops that could be reduced using fewer crossings. Note that both mosaics in Figures 10 and 11 are space-efficient, and the latter is also minimally space-efficient.

While it is fairly simple to tell the mosaic and tile number of knots with few crossings, it becomes much more difficult as we examine more complicated knots. For all knots with crossing number 8 and less the mosaic number [2] and tile number [1] are known. Therefore, we implemented programming algorithms in an attempt to speed up the search process, in hopes to find the mosaic and tile number for all 9 and 10 crossing knots.

Previous Work

In 2019, Knowles and Heap found the space-efficient mosaics of all knots with mosaic number 6 or less (Heap & Knowles, 2019). They accomplished this by proving that any space-efficient 6-mosaic will have one of the layouts listed in Figure 12 (Heap & Knowles, 2018). That is, if the mosaic is space-efficient, it will have an outer shell as shown in the layout with tiles T_7 through T_{10} (from Figure 4) filling in the inside of the shell. Going through these layouts by hand, they were able to find all possible space-efficient 6-mosaics. These mosaics included all knots with crossing number up to 8, and some knots with crossing number 9 through 13.

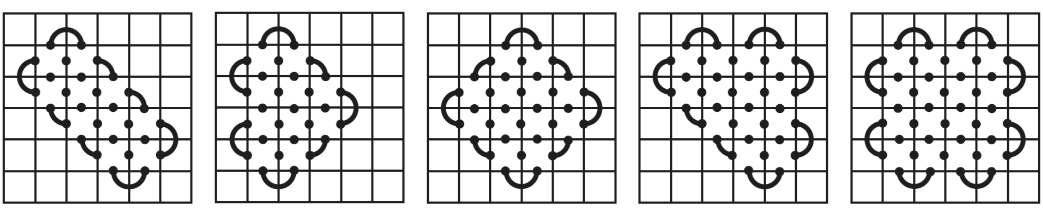

Figure 12: All Space-efficient 6-mosaics

We started to look at the 7×7 mosaics in hopes to find the remaining knots with crossing number 9 and 10. Fortunately, Heap and LaCourt have proven that the layouts shown in Figure 13 contain all possible space-efficient 7-mosaics (Heap & LaCourt, 2020).

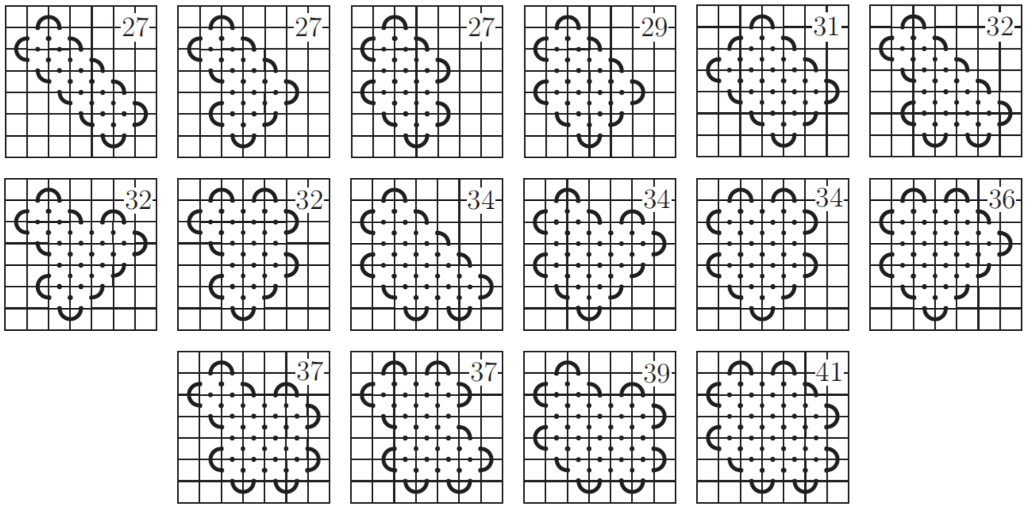

Figure 13: All Space-efficient 7-mosaics, with number of non-blank tiles in the top right corner

We used these layouts as starting points to find all the possible knots.

Presented here is a continuation of Gregory Vinal's work, showing the preliminary results of our work. Here we expand those results and provide images of the knots as proofs. Additionally we completed the knot mosaic website that was introduced in (Vinal, 2020).

Methods

To find every knot with crossing number 9 or 10 on a space-efficient 7-mosaic, we pieced together several programs into one pipeline. This pipeline creates every possible mosaic and identifies the name of the knot it depicts. In the following subsections we describe each piece of the pipeline. A diagram of an example run of the pipeline is shown in Figure 14.

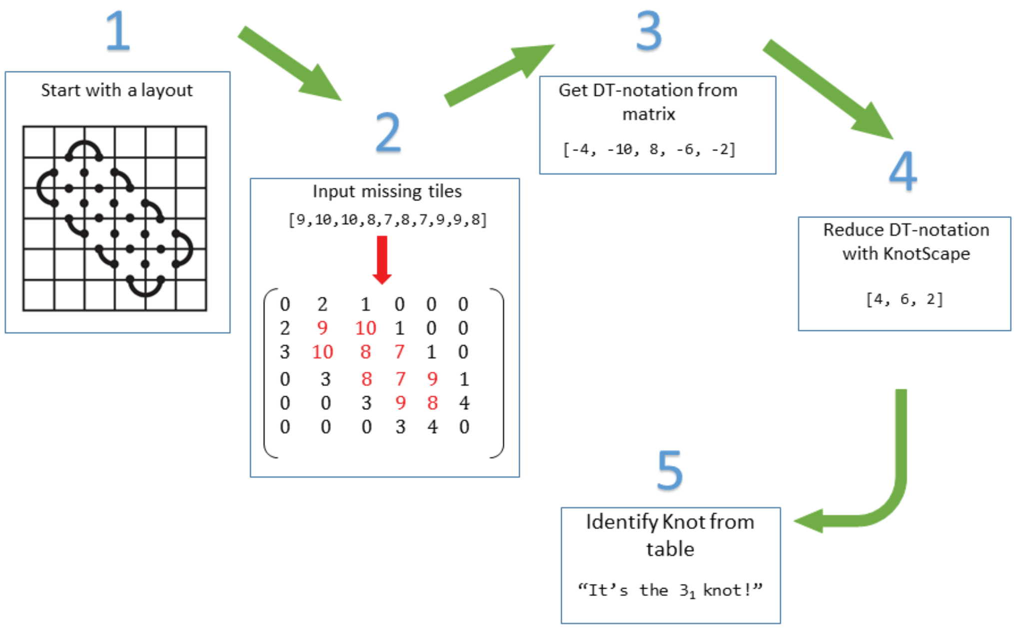

Figure 14: Example run of the pipeline

3.1 Generate All Knots

We begin the pipeline by selecting one of the identified layouts from Figure 13. Each layout has a given number of interior positions that need to be filled. We create a matrix in Python that represents the layout mosaic using the integers associated with each tile (as in Figure 3), leaving the empty inner tiles as a null value. For a layout with n tile positions to fill, we start the pipeline by generating a vector of length n for every possible combination of the numbers 7, 8, 9, and 10 as entries of this vector. Replacing the null values of the layout matrix with each of these vectors will create all possible mosaics for this layout. However this requires 4^n vectors, and as n increases it becomes infeasible to create so many vectors at once. So we split the process in half.

Let $k = floor(\frac{n}{2})$. Using the *product* function from the *itertools* package, we create 4^k vectors of length k and 4^{n-k} vectors of length n-k, including all possible configurations of integers 7 through 10 for each length. By concatenating every pair of vectors of the two lengths we can create all 4^n possibilities more quickly and using less memory.

Splitting this process into two parts also allows for further optimization. By only looking for knots with a minimum crossing number *m* (in our case, 9) we only need to check mosaics with at least *m* crossings. These correspond to concatenated vectors containing at least *m* entries that are 9 or 10, since 9 and 10 correspond to the crossing-tiles T_9 and T_{10}. We count the number of 9s and 10s in each of the smaller vectors, and only concatenate pairs of vectors that will result in a total of at least *m* crossing tiles. Doing so reduces the number of vectors created from $4^n = 2^{2n}$ to

$$\sum_{i=m}^{n} \binom{n}{i} 2^i * 2^{n-i} = 2^n \sum_{i=m}^{n} \binom{n}{i}$$

possibilities.

We then fill in the layout matrix by putting the *n* values from the vectors into the *n* empty spots of the matrix, one vector at a time. We send this matrix representation of a knot mosaic on to the next step in the pipeline.

3.2 Get the Reduced DT Notation

Now that we have a method to create all the possible mosaics in a layout, we need a way to identify them. To do so, we will use the Dowker-Thistlethwaite notation (DT notation) (Dowker & Thistlethwaite, 1983). The DT notation is a 1-dimensional representation of a knot diagram as a list of even integers, and is determined by the following method:

Start at an arbitrary point on a knot with *n* crossings.

Travel along the knot, labeling each crossing with integers 1 through 2*n* sequentially. When assigning an even number, if you are at an under crossing, assign the negative even number instead.

Each crossing will have two numbers, one even and one odd. Arrange the *n* odd numbers in ascending order with their associated even numbers, for example [(1, 4),(3, -6,),(5, 2)].

Take the even numbers in the order determined by the odds (4, -6, 2). This sequence of even numbers is the DT notation.

Although it is not a knot invariant, the DT notation does uniquely identify knot diagrams, and every DT notation of a knot can be reduced to the representation of the simplest knot diagram of that knot. That is, given any knot diagram, the DT notation of that diagram can be reduced to the DT notation that represents the same knot in its most reduced diagram.

We wrote a Python function to produce the DT notation given a knot mosaic represented as a matrix. The function identifies the first non-zero entry of the matrix (starting at the top left and continuing down the row) and then follows the knot all the way around until it reaches the initial spot again. At each crossing, it adds the associated part of the DT notation. If the program ever returns to the initial position before it has passed the expected number of positions we know that we have found a link, and so we can disregard that mosaic since we are only interested in finding knots.

Although we can create the DT notation, we cannot always determine the name of the knot just yet. We need to reduce the notation, and we do so using the program *Knotscape*, created by Morwen Thistlethwaite in the 1990s. We adapted this C program to take multiple notations in sequence and output the reduced DT notation of each one. If the program finds a composite knot, we throw that one out, since we are only interested in prime knots.

3.3 Identify the Knot

The final step in the pipeline is to identify the name of the knot. Using the newly reduced DT notation, we reference a table that contains all the 9 and 10 crossing knots and their reduced DT notations. We output the name of the knot (if the DT notation is found on the table) along with the original vector that created the mosaic, so that we can reproduce the knot mosaic we found. If the DT notation is not on the table, it is from a knot with crossing number less than 9 or greater than 10, and so we are not concerned with it.

We then reference all the knots found on the current layout with all the knots found on previous layouts. We identify any new ones, and note if we found a knot using fewer tiles than before. We also wrote a bash script to count the number of crossings in each mosaic of a layout and identify the mosaic for each unique knot with the fewest number of crossings.

To test the pipeline we ran the first four 6×6 layouts shown in Figure 12 and the first three 7×7 layouts from Figure 13 (Heap & Knowles, 2019).

We ran the pipeline on machines with 12 cores, two threads per core, 125 gigabytes of RAM, 32, 256, and 3072 KB of cache, and max CPU clock speed of 2900 MHz. The first two 6×6 layouts each take less than 5 minutes to run. The third layout took 40 minutes, and the fourth one took about 11 hours. The first three 7×7 layouts each took about 4.5 hours to complete, and the fourth layout took about 9 days. The run of the fifth layout was broken up into multiple sections due to a power outage, and it has not completed its run in an aggregate of about 70 days. The sixth layout finished in about 18 days, and the seventh took 24 days.

4. Results

After the successful test runs we began running the larger 7×7 layouts through the pipeline. We have successfully found all of the knots that can fit on the first four layouts listed in Figure 13, and we have initial results from the fifth layout. In these first five layouts we found all 9 and 10 crossing knots that had not been identified as knot mosaics in previous work, leading to the following theorem:

Theorem 4.1

All prime knots with crossing number 10 or less have mosaic number 7 or less and tile number 31 or less.

The specific results from the size 7 mosaics are listed below. The first layout uses 27 non-blank tiles, and it confirmed the following theorem from Heap & LaCourt (2020):

Theorem 4.2

(Heap & LaCourt, 2020) The following prime knots have mosaic number 7 and tile number 27:

9_6, 9_{15}, 9_{18}

10_5, 10_6, 10_7, 10_8, 10_9, 10_{10}, 10_{13}, 10_{14}, 10_{15}, 10_{16} , 10_{17}, 10_{18}, 10_{19}, 10_{24}, 10_{25}, 10_{26}, 10_{29}, 10_{30}, 10_{31}, 10_{32}, 10_{33}, 10_{35}, 10_{36}, 10_{38}, 10_{39}

The second and third layouts did not produce any knots that were not found on a previous layout. The fourth layout uses 29 non-blank tiles and leads to the next theorem (Vinal, 2020).

Theorem 4.3

The following prime knots have mosaic number 7 and tile number 29:

9_{22}, 9_{25}, 9_{29}, 9_{30}, 9_{32}, 9_{33}, 9_{34}, 9_{36}, 9_{38}, 9_{39}, 9_{42}, 9_{43}, 9_{44}, 9_{45}, 9_{47}, 9_{49}

10_{23}, 10_{27}, 10_{37}, 10_{40}, 10_{42}, 10_{43}, $10_{45\text{-}57}$, $10_{67\text{-}73}$, 10_{79}, 10_{82}, 10_{83}, 10_{84}, 10_{86}, 10_{87}, $10_{90\text{-}95}$, 10_{101}, 10_{102}, 10_{103}, 10_{106}, 10_{107}, 10_{112}, 10_{113}, 10_{114}, 10_{117}, $10_{128\text{-}136}$, 10_{145}, 10_{146}, 10_{147}, $10_{149\text{-}153}$, 10_{156}, 10_{158}, $10_{160\text{-}164}$

Although the fifth layout has not finished completely, the initial results have given us the following theorem:

Theorem 4.4

The following prime knots have mosaic number 7 and tile number 31:

9_{40}, 9_{41}

$10_{58\text{-}60}$, 10_{80}, 10_{81}, 10_{88}, 10_{89}, $10_{96\text{-}99}$, 10_{104}, 10_{105}, $10_{108\text{-}111}$, 10_{115}, $10_{118\text{-}123}$, 10_{137}, 10_{138}, 10_{154}, 10_{157}, 10_{165}

For many knots, the minimal-mosaic tile number is the same as the tile number. However, Heap had identified 13 9 and 10-crossing knots with mosaic number 6 and minimal mosaic tile number 32 that we were able to find with a smaller tile number (either 27 or 29) as a 7-mosaic (Vinal, 2020).

Theorem 4.5

The following prime knots have mosaic number 6 with a tile number that was realized on a 7-mosaic:

Tile number 27: 9_{10}, 10_{11}, 10_{20}, 10_{21}

Tile number 29: 9_{16}, 9_{35}, 10_{61}, 10_{62}, 10_{64}, 10_{74}, 10_{76}, 10_{77}, 10_{139}

Many space-efficient mosaics have more crossings than the crossing number of the knot that it represents. Mosaics in which the crossing number is realized have been found on the 7×7 grid for all 9-crossing knots and for all but two 10-crossing knots. To find these results, we have run the sixth 7×7 layout through the pipeline searching for 9 and 10-crossing knots, and we have initial results for the ninth and twelfth layouts.

Theorem 4.6

The crossing number for the following knots is first realized on a 7-mosaic in which the tile number is not realized, using the given number of tiles:

27 tiles: 9_{10}, 9_{13}, 9_{21}, 9_{26}

29 tiles: 9_{3}, 9_{7}, 9_{9}, 9_{15}, 9_{16}, 9_{19}, 9_{24}, 9_{37}, 9_{46}, 9_{48}, 10_{1}, 10_{5}, 10_{11}, $10_{13\text{-}16}$, 10_{21}, 10_{22}, 10_{24}, 10_{31}, $10_{33\text{-}36}$,10_{38}, 10_{39}, 10_{62}, 10_{63}, 10_{65}, 10_{74}, 10_{78}, 10_{139}, 10_{140}, 10_{142}, 10_{144}

31 tiles: 9_{4}, 9_{12}, 9_{29}, 9_{35}, 10_{6}, 10_{7}, 10_{9}, 10_{12}, 10_{17}, 10_{37}, 10_{61}, 10_{64}, 10_{67}, 10_{68}, 10_{70}, 10_{72}, 10_{77}, 10_{79}, 10_{84}, $10_{90\text{-}93}$, 10_{114}, 10_{152}, 10_{153}, 10_{158}, 10_{163}

34 tiles: 10_{20}*

It is possible that the 10_{20} knot listed in Theorem 4.6 could be found as a mosaic with crossing number realized using 31 tiles, pending the finished pipeline run of the fifth space-efficient layout from Figure 13.

Corollary 4.6.1

The prime knots with crossing number 9 or 10 not listed in Theorem 4.6 have mosaics in which both the crossing number and tile number are realized, except for the 10_{3} *and* 10_{76} *knots.*

Space-efficient mosaics in which the 10_3 and 10_{76} knots are displayed with crossing numbers realized have not yet been found. We know that such mosaics do not appear on the first four or the sixth 7×7 layouts listed in Figure 13.

In analyzing the results, we found that the first three size 7 layouts all produced the same exact knots. This was also true for the first two size 6 layouts and the sixth, seventh, and eighth size 7 layouts. This gives evidence to support the following conjecture proposed (Heap & LaCourt, 2020).

Conjecture 4.7

> *Space-efficient layouts of the same size and using the same number of non-blank tiles produce the same prime knots.*

Ongoing Work

The fifth layout from Figure 13 is currently running, searching only for knots with crossing number 9 and 10, in case we find a mosaic that realizes the crossing number for the two knots mentioned in Corollary 4.6.1 or the 10_{20} knot.

Having found all the 9 and 10-crossing knots, we now can expand our search to include knots with higher crossing numbers, as well as knots with crossing number less than 9 for which we have not found a mosaic that realizes the crossing number. We have completed running the first four 6×6 layouts shown in Figure 12 searching for all knots, and the fifth layout is still running. We have also finished running the first three 7×7 layouts. The fifth 6×6 layout and the fourth 7×7 layout are currently running. So far through these runs we have found mosaics that realize the crossing number for all knots with crossing numbers up to 8.

As a part of this research we have developed a website that serves as a tool for working with knot mosaics. On this website users can build a mosaic of their desired size by dragging the tiles onto a grid. They can choose to start from any of the space-efficient layouts for sizes 5 through 7, or start from a blank grid. Once they have created a mosaic they can click a button to identify the knot. This sends the matrix representation of their mosaic into our pipeline, and returns the name of the knot depicted. The website will recognize links and composite knots but will not identify or name them.

Additionally, the results of our research are stored in a database on this website so users can search for and display the knot mosaic representation of the knots that we have found. They are able to specify whether to find the mosaic with the fewest tiles, fewest crossings, or smallest size, and how to prioritize these three options. When a knot mosaic is displayed the website will also tell the user whether the tile number, crossing number, and mosaic number are realized.

This website is not currently available for public use. However once it is, it will be available and free to use around the world. A screenshot of the website is attached in Figure 15.

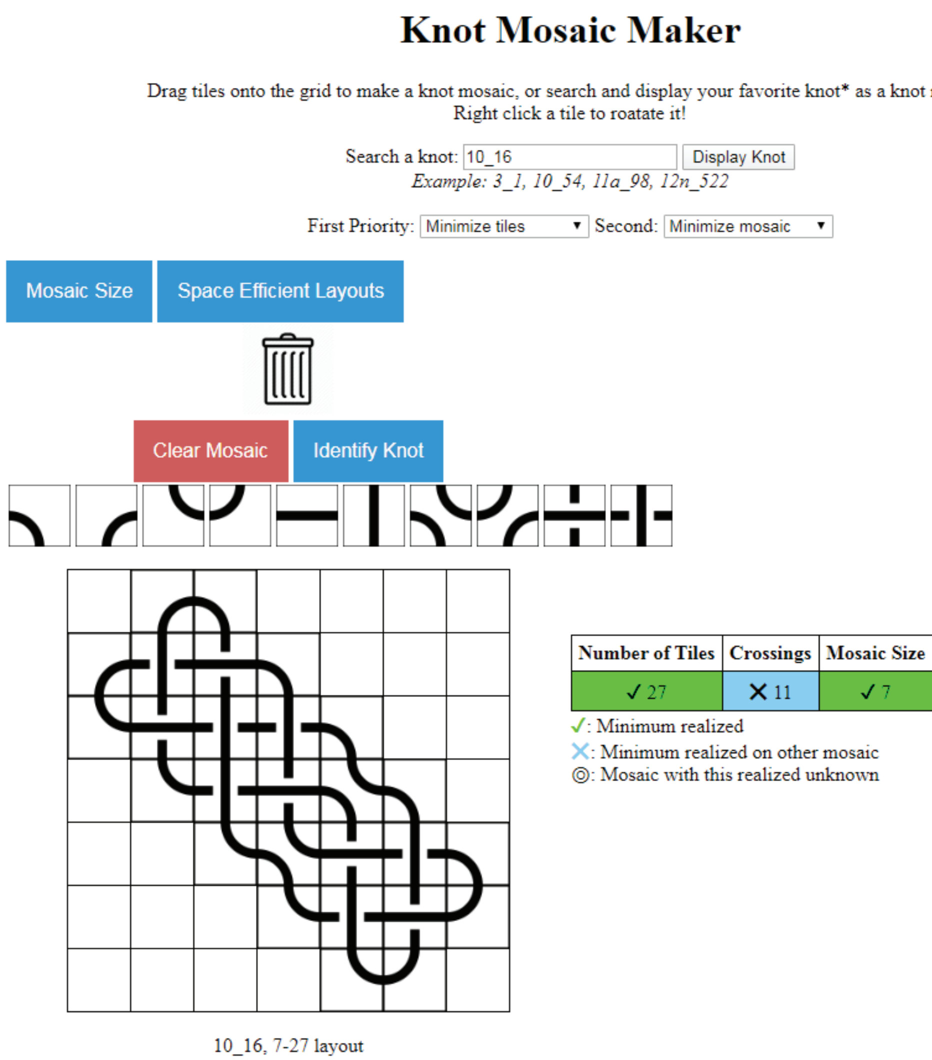

Figure 15: Screenshot of the website

Appendix

APPENDIX A: Knot Mosaics from Theorems

The appendix contains mosaics for the knots referenced in the theorems. We exclude mosaics from Theorem 4.2 as they were included in Heap & LaCourt, (2020). Mosaics that are marked with an * are space-efficient mosaics that have more crossings than the crossing number of the knot they represent. Our images were created using a program that we wrote that takes the matrix representation of a mosaic and draws the mosaic using the Python PyCairo package. The appendix is located in the online edition of *Proceedings of GREAT Day 2020*, found at https://knightscholar.geneseo.edu/proceedings-of-great-day/vol2020/iss1/9.

References

Dowker, C.H. & Thistlethwaite, M.B. (1983). Classification of Knot Projections. *Topology and its Applications*, *16*(1),19–31. doi.org/10.1016/0166-8641(83)90004-4

Heap, A. & Knowles, D. (2018). Tile Number and Space-efficient Knot Mosaics. *Journal of Knot Theory and Its Ramifications*, *27*(06): 1850041. doi.org/10.1142/S0218216518500414

Heap, A. & Knowles, D. (2019). Space-efficient Knot Mosaics for Prime Knots with Mosaic Number 6. *Involve, a Journal of Mathematics*, *12*(5), 767–789. doi.org/10.2140/involve.2019.12.767

Heap, A. & LaCourt, N. (2020), Space-efficient Prime Knot 7-Mosaics. *Symmetry*, *12*(4), 576. doi.org/10.3390/sym12040576

Lee, H.J., Ludwig, L., Paat, J., & Peiffer, A. (2017). Knot Mosaic Tabulation. *Involve, a Journal of Mathematics*, *11*(1), 13–26. doi.org/10.2140/involve.2018.11.13

Vinal, G. (2020). Space-efficient Knot Mosaics of Size 7. *Proceedings of GREAT Day*, *2019*(1), 13. https://knightscholar.geneseo.edu/proceedings-of-great-day/vol2019/iss1/13