Enumeration of Messy Knot Mosaics

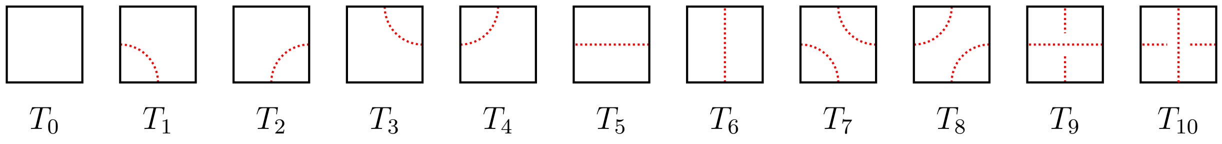
Jack Hanke **Richard Schank**
Northwestern University

Abstract

Lomonaco and Kauffman introduced a system of mosaics to model quantum knots. These systems of mosaics are composed of an $m \times n$ rectangular grid of 11 possible tiles. Oh and colleagues introduced a state matrix recursion method to exactly enumerate a subset of these mosaics that have the property of being suitably connected, which they call knot mosaics. We introduce and enumerate mosaics with the related property in which only some tiles must be suitably connected, which we call messy knot mosaics.

1 Introduction

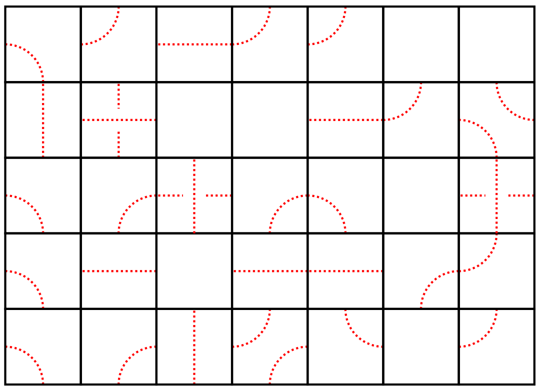
Lomonaco and Kauffman [3] introduced a model for quantum knots in which an $m \times n$ matrix is constructed using 11 distinct symbols called *tiles*. These tiles, diagrammed below, are composed of unit squares with dotted lines connecting 2 or 4 sides at their midpoint.



We denote the set of tiles $\mathbb{T} = \{T_0, \dots, T_{10}\}$. A *mosaic* of size (m, n) is an $m \times n$ matrix made up of elements from $\mathbb{T}$. Figure 1a shows an example mosaic of size $(5, 7)$. We denote the set of all mosaics of size (m, n) as $\mathbb{M}^{(m,n)}$. As there are 11 elements in $\mathbb{T}$, there are 11^{mn} mosaics in $\mathbb{M}^{(m,n)}$. A *mosaic system* is then a subset of $\mathbb{M}^{(m,n)}$ with some property.

We are interested in mosaics with the property of being *suitably connected*, which is defined as follows. Consider an edge shared between two tiles in Figure 1a. The edge has either 0, 1, or 2 dotted lines drawn from its midpoint. Also note that the edges of the tiles on the boundary of the matrix are not shared by another tile. Therefore these edges only have 0 or 1 dotted lines drawn from their midpoint. A mosaic is suitably connected if all edges have 0 or 2 dotted lines drawn from their midpoint.

Lomonaco and Kauffman [3] call these *knot mosaics* because, other than the mosaic consisting of all T_0 tiles, the dotted lines form *knots*. Following the notation in [9] we denote the subset of mosaics of size (m, n) that are knot mosaics as $\mathbb{K}^{(m,n)}$. Figure 1b shows a knot mosaic of size $(5, 7)$ that contains 3 knots, with the tiles that make up the knots highlighted in gray. Note that a mosaic can contain knots isomorphic to the unknot, as well as knots that encompass other knots.


(a) A mosaic

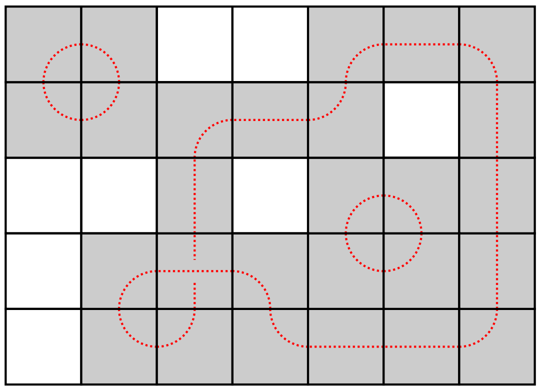

(b) A knot mosaic

Figure 1: Examples of mosaics of size $(5, 7)$ made of tiles in $\mathbb{T}$

Let $k_{m,n} = |\mathbb{K}^{(m,n)}|$ be the number of knot mosaics of size (m, n). First notice that if either m or n is 1, one can only construct a knot mosaic using T_0 tiles, so $k_{m,1} = k_{1,n} = 1$. Oh et al. [9] showed the following for $m, n \geq 2$.

Theorem 1.1 ([9])**.** *The number of knot mosaics of size (m, n) for $m, n \geq 2$ is $k_{m,n} = 2\,\|(X_{m-2} + O_{m-2})^{n-2}\|$, where X_{m-2} and O_{m-2} are $2^{m-2} \times 2^{m-2}$ matrices defined as*

$$X_{k+1} = \begin{bmatrix} X_k & O_k \\ O_k & X_k \end{bmatrix} \text{ and } O_{k+1} = \begin{bmatrix} O_k & X_k \\ X_k & 4O_k \end{bmatrix},$$

for $k = 0, 1, \ldots, m-3$, and $X_0 = O_0 = \begin{bmatrix} 1 \end{bmatrix}$. Here $\|N\|$ denotes the sum of elements of matrix N.

Oh and colleagues refer to these matrices X_k and O_k as *state matrices*. The authors utilize this state matrix recursion to bound the growth rate of knot mosaics $\delta = \lim_{n\to\infty} k_{n,n}^{\frac{1}{n^2}}$ [6, 8, 1], and Oh further adapts the method to solve problems in monomer and dimer tilings [5, 7]. An unexamined direction in this research program is modifying the suitably connected property. This motivates us to introduce *messy knot mosaics*.

2 Messy Knot Mosaics

Definition 2.1. A *messy knot mosaic* is a mosaic that contains at least one knot.

Figure 2 shows two examples of messy knot mosaic of size $(5, 7)$ that contains 3 knots[1], with the tiles that make up the knots highlighted in gray. All knot mosaics are messy knot mosaics.

[1]Certain permutations of $\{T_1, T_2, T_3, T_4\}$ and $\{T_7, T_8\}$ can make shapes that appear to be knots but have hanging connections, as seen in the $(0, 4)$ position in the right example in Figure 2. These are not considered knots by this paper and all referenced works.

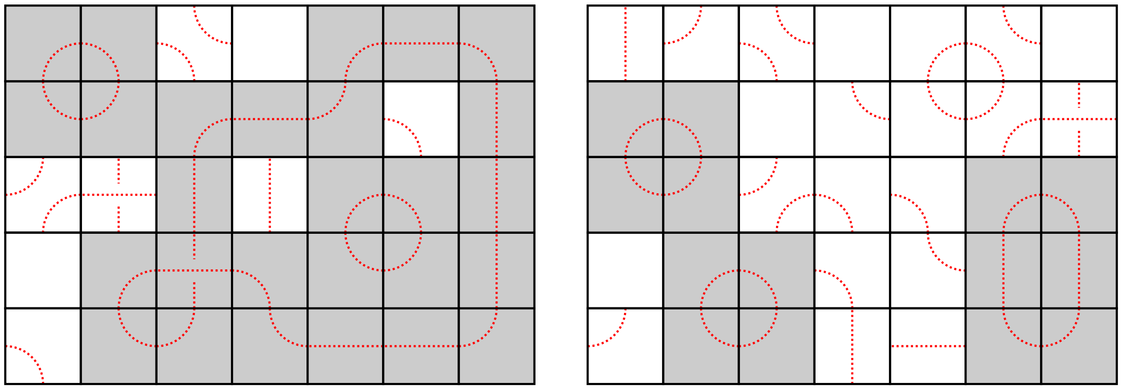
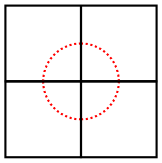


Figure 2: Messy knot mosaics

It turns out to be simpler to enumerate the number of mosaics that *do not* contain a knot. Therefore, let $\mathbb{S}^{(m,n)}$ be the set of mosaics that do not contain a knot, and let $|\mathbb{S}^{(m,n)}| = s_{m,n}$. Clearly the number of messy knot mosaics is then $11^{mn} - s_{m,n}$.

First, note that the knot made up of the least amount of cells is this:



We can then conclude that $s_{n,1} = 11^n$, and $s_{2,2} = 11^4 - 1$. For $n, m \geq 2$, we first define the state matrices for messy knot mosaics.

Definition 2.2. Define $A(2) = \begin{bmatrix} 11^2 & 1 \\ -1 & 1 \end{bmatrix}$. We recursively define $A(k+1) \in \mathbb{N}^{2^k \times 2^k}$ given $A(k)$. Begin by writing $A(k) = \begin{bmatrix} A_{0,0} & A_{0,1} \\ A_{1,0} & A_{1,1} \end{bmatrix}$, where the block matrices $A_{i,j}$ are square block matrices of size $2^{k-1} \times 2^{k-1}$. We then have

$$A(k+1) = \begin{bmatrix} 11A_{0,0} & \frac{1}{11}A_{0,0} & 11A_{0,1} & A_{0,1} \\ -\frac{1}{11}A_{0,0} & \frac{1}{11}A_{0,0} & 4A_{0,1} & A_{0,1} \\ 11A_{1,0} & -4A_{1,0} & 11A_{1,1} & A_{1,1} \\ A_{1,0} & -A_{1,0} & A_{1,1} & 11A_{1,1} \end{bmatrix}.$$

Construct $A(m)$ by starting with $k = 2$ and recursing until $k = m$.

Theorem 2.1. *The number of mosaics of size (m, n) that do not contain a knot is the $(0, 0)$ entry of $A(m)^n$.*

3 Preliminaries

We begin by defining a mapping f between $\mathbb{M}^{(m,n)}$ to a *binary lattice* of size (m, n). A binary lattice of size (m, n) is a rectangular lattice of $m+1$ by $n+1$ vertices, in which the boundary

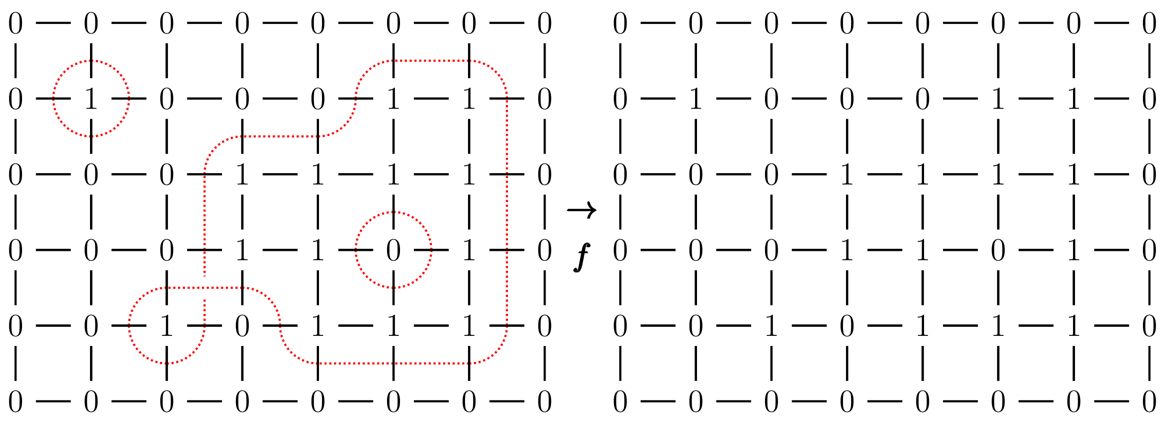


Figure 3: f applied to the left mosaic in Figure 2, resulting in a binary lattice

vertices are labeled 0, and the interior vertices are either 0 or 1. An example of a binary lattice of size $(5, 7)$ is shown on the right of Figure 3. Also let $\mathbb{L}^{(m,n)}$ be the set of all binary lattices of size (m, n), which gives $\left|\mathbb{L}^{(m,n)}\right| = 2^{(m-1)(n-1)}$.

Definition 3.1. $f : \mathbb{M}^{(m,n)} \to \mathbb{L}^{(m,n)}$ takes a mosaic and labels each vertex with the following rule. If the vertex is surrounded by an even number of knots (including 0 knots), label it 0. If the vertex is surrounded by an odd number of knots, label it 1. Removing the red dotted lines from the tiles gives the binary lattice.

Similarly, define the *preimage* of a set L under f to be

$$f^{-1}(L) = \{m \in \mathbb{M}^{(m,n)} | f(m) \in L\}.$$

We want to compute $s_{m,n}$ by computing $f^{-1}(\ell)$ for each $\ell \in \mathbb{L}^{(m,n)}$, and then summing over all ℓ. We begin by finding a simple way to compute $f^{-1}(\ell)$ for a binary lattice ℓ by examining the structure of ℓ.

Definition 3.2. Let a *cell* be the portion of the binary lattice that an individual tile maps to, and let C be the set of unique cells.

For convenience, we give a pair of indexes to each of the $|C| = 2^4$ unique cells. The first index is the binary number formed by reading the bottom two vertices from left to right. The second index is the binary number formed by reading the top two vertices from left to right. Below is a diagram of all 2^4 cells with their indexes listed below.

$\begin{array}{ccc}0&\text{—}&0\\ \vert&&\vert\\ 0&\text{—}&0\end{array}$	$\begin{array}{ccc}0&\text{—}&1\\ \vert&&\vert\\ 0&\text{—}&0\end{array}$	$\begin{array}{ccc}1&\text{—}&0\\ \vert&&\vert\\ 0&\text{—}&0\end{array}$	$\begin{array}{ccc}1&\text{—}&1\\ \vert&&\vert\\ 0&\text{—}&0\end{array}$	$\begin{array}{ccc}0&\text{—}&0\\ \vert&&\vert\\ 0&\text{—}&1\end{array}$	$\begin{array}{ccc}0&\text{—}&1\\ \vert&&\vert\\ 0&\text{—}&1\end{array}$	$\begin{array}{ccc}1&\text{—}&0\\ \vert&&\vert\\ 0&\text{—}&1\end{array}$	$\begin{array}{ccc}1&\text{—}&1\\ \vert&&\vert\\ 0&\text{—}&1\end{array}$
(00, 00)	**(00, 01)**	**(00, 10)**	**(00, 11)**	**(01, 00)**	**(01, 01)**	**(01, 10)**	**(01, 11)**
$\begin{array}{ccc}0&\text{—}&0\\ \vert&&\vert\\ 1&\text{—}&0\end{array}$	$\begin{array}{ccc}0&\text{—}&1\\ \vert&&\vert\\ 1&\text{—}&0\end{array}$	$\begin{array}{ccc}1&\text{—}&0\\ \vert&&\vert\\ 1&\text{—}&0\end{array}$	$\begin{array}{ccc}1&\text{—}&1\\ \vert&&\vert\\ 1&\text{—}&0\end{array}$	$\begin{array}{ccc}0&\text{—}&0\\ \vert&&\vert\\ 1&\text{—}&1\end{array}$	$\begin{array}{ccc}0&\text{—}&1\\ \vert&&\vert\\ 1&\text{—}&1\end{array}$	$\begin{array}{ccc}1&\text{—}&0\\ \vert&&\vert\\ 1&\text{—}&1\end{array}$	$\begin{array}{ccc}1&\text{—}&1\\ \vert&&\vert\\ 1&\text{—}&1\end{array}$
(10, 00)	**(10, 01)**	**(10, 10)**	**(10, 11)**	**(11, 00)**	**(11, 01)**	**(11, 10)**	**(11, 11)**

Next let $u_{(i,j)}$ be the number of tiles in $\mathbb{T}$ that can map to cell (i, j). These values are simple to calculate, as each tile in $\mathbb{T}$ can only be part of a knot in certain ways. For example, $u_{(00,01)} = 1$, as cell $(00, 01)$ can only be formed by a mosaic with T_3 in that location. We can see that $u_{(01,10)} = 4$, as cell $(01, 10)$ can be from the T_7, T_8, T_9, or T_{10} tiles. Finally, we have $u_{(00,00)} = 11$, as any tile can fail to contribute to forming a knot. Table 1 summarizes the tiles in the preimage for each cell (i, j).

Cell (i, j)	**Preimage**	$\mathbf{u_{(i,j)}}$	**Cell (i, j)**	**Preimage**	$\mathbf{u_{(i,j)}}$
$(00, 00)$	$\mathbb{T}$	11	$(10, 00)$	T_1	1
$(00, 01)$	T_3	1	$(10, 01)$	T_7, T_8, T_9, T_{10}	4
$(00, 10)$	T_4	1	$(10, 10)$	T_6	1
$(00, 11)$	T_5	1	$(10, 11)$	T_2	1
$(01, 00)$	T_2	1	$(11, 00)$	T_5	1
$(01, 01)$	T_6	1	$(11, 01)$	T_4	1
$(01, 10)$	T_7, T_8, T_9, T_{10}	4	$(11, 10)$	T_1	1
$(01, 11)$	T_1	1	$(11, 11)$	$\mathbb{T}$	11

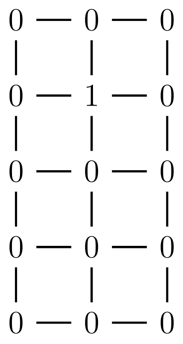
Table 1: Preimages of each unique cell

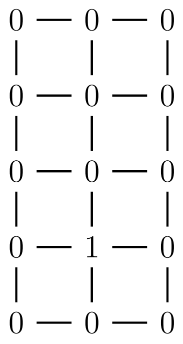
However, for some binary lattice ℓ the quantity

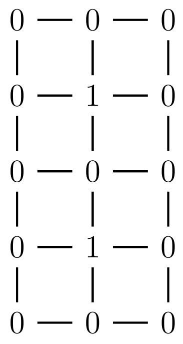
$$U(\ell) := \prod_{\text{Cell } (i,j)\in\ell} u_{(i,j)} \tag{1}$$

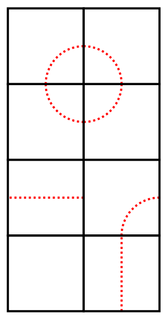
is not necessarily equal to $|f^{-1}(\{\ell\})|$, as $U(\ell)$ does not *just* count the number of mosaics that map to ℓ under f.

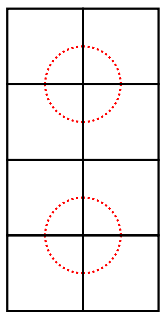
Example 3.1. Consider the following binary lattices for $s_{4,2}$.


$U(\ell_1) = 11^4$


$U(\ell_2) = 11^4$


$U(\ell_3) = 11^0$


Mosaic 1


Mosaic 2

$U(\ell_1)$ uniquely counts Mosaic 1, but both $U(\ell_1)$ and $U(\ell_2)$ count Mosaic 2, for which $f(\text{Mosaic 2}) = \ell_3$.

Definition 3.3. A knot is *specified* by a binary lattice ℓ if all mosaics in $f^{-1}(\{\ell\})$ contain the knot.

Example 3.2. From Example 3.1, ℓ_1 specifies the knot in the top 2 rows of Mosaic 1 and Mosaic 2, but not the knot in the bottom 2 rows of Mosaic 2.

Definition 3.4. Let $K : \mathbb{L}^{(m,n)} \to \mathbb{N}$ be the number of knots specified in ℓ.

Example 3.3. From Example 3.1, $K(\ell_1) = 1$, $K(\ell_2) = 1$, and $K(\ell_3) = 2$.

Definition 3.5. For a binary lattice ℓ, let $\mathbb{U}(\ell)$ be the set of binary lattices whose preimage mosaics under f are counted by $U(\ell)$. A binary lattice ℓ' is in $\mathbb{U}(\ell)$ if one can replace either some number of $(00,00)$ or $(11,11)$ cells in ℓ with other cells in C to create ℓ'. This corresponds with specifying new knots in the mosaics in the preimage of ℓ while leaving all knots specified by ℓ unchanged.

Example 3.4. From Example 3.1, $\mathbb{U}(\ell_1) = \{\ell_1, \ell_3\}$, $\mathbb{U}(\ell_2) = \{\ell_2, \ell_3\}$, and $\mathbb{U}(\ell_3) = \{\ell_3\}$.

From the definition of $\mathbb{U}$, we have

$$U(\ell) = \sum_{\ell' \in \mathbb{U}(\ell)} |f^{-1}(\ell')|. \tag{2}$$

Therefore

$$s_{m,n} \neq \sum_{\ell \in \mathbb{L}^{(m,n)}} U(\ell),$$

as the sum overcounts mosaics for all ℓ that have $\mathbb{U}(\ell) \neq \{\ell\}$. For instance, consider the binary lattice ℓ^* made up of all $(00,00)$ cells. ℓ^* specifies 0 knots and has $U(\ell^*) = 11^{mn} = \left|\mathbb{M}^{(m,n)}\right|$, which clearly overcounts $s_{m,n}$. Also note that $\mathbb{U}(\ell^*) = \mathbb{L}^{(m,n)}$.

Proposition 3.1. *By regrouping terms we have*

$$\sum_{\ell \in \mathbb{L}^{(m,n)}} U(\ell) = \sum_{\ell \in \mathbb{L}^{(m,n)}} \sum_{\ell' \in \mathbb{U}(\ell)} |f^{-1}(\ell')| = \sum_{\ell \in \mathbb{L}^{(m,n)}} \left(\binom{K(\ell)}{0} + \cdots + \binom{K(\ell)}{K(\ell)} \right) |f^{-1}(\ell)|. \tag{3}$$

Proof. The first equality follows directly from Equation 2. For the second equality, consider a binary lattice ℓ with $K(\ell) > 0$. From the definition of $\mathbb{U}$,

TODO $\square$

Proposition 3.2. *The number of mosaics of size (m, n) that do not contain a knot $s_{m,n}$ has*

$$s_{m,n} = \sum_{\ell \in \mathbb{L}^{(m,n)}} (-1)^{K(\ell)} U(\ell). \tag{4}$$

Proof. By the binomial theorem,

$$0 = (1-1)^{K(\ell)} = \left(\binom{K(\ell)}{0} - \binom{K(\ell)}{1} + \cdots + (-1)^{K(\ell)} \binom{K(\ell)}{K(\ell)} \right),$$

all terms in Equation 3 where $\ell \neq \ell^*$ are 0. Therefore,

$$\sum_{\ell \in \mathbb{L}^{(m,n)}} (-1)^{K(\ell)} U(\ell) = \binom{0}{0} |f^{-1}(\ell^*)| = s_{m,n}.$$

$\square$

It is important to note here that a knot that contains tiles T_7 or T_8 can appear to be two separate knots which we consider as 1 knot.

Example 3.5. Figure 4 shows a mosaic that appears to have 2 knots, but we only consider as 1 knot. A rule of thumb that can be followed is if knots that contain T_7 or T_8 tiles can be replaced by T_9 or T_{10} tiles to form a single knot, then the original "knots" we consider as 1 knot.

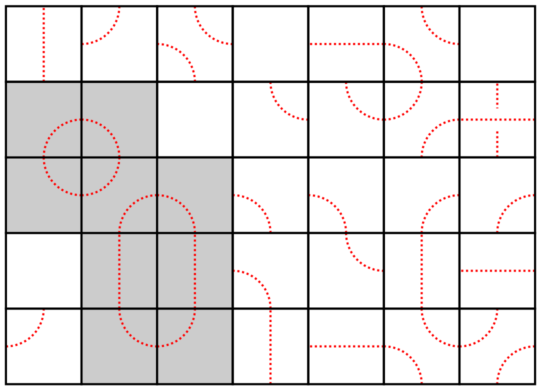


Figure 4: A messy knot mosaic with 1 knot

4 A "Per-Cell" Identity

Though Equation 4 does compute $s_{m,n}$, the K function is a function that references the entire, global structure of ℓ. It is useful to recover the $(-1)^{K(\ell)}$ term using only "per-cell", local information.

Proposition 4.1. *Let $\mathcal{K}$ be the set of cells in a binary lattice ℓ that specify a knot.*

There exists $p_{(i,j)} \in \{-1, 1\}$ so that

$$\prod_{\textit{Cell } (i,j) \in \mathcal{K}} p_{(i,j)} = -1 \tag{5}$$

for all possible $\mathcal{K}$.

Proof. **TODO** □

Proposition 4.1 allows us to rewrite the term in the sum of Equation 4 to

$$(-1)^{K(\ell)} U(\ell) = (-1)^{K(\ell)} \prod_{\text{Cell } (i,j) \in \ell} u_{(i,j)} = \prod_{\text{Cell } (i,j) \in \ell} p_{(i,j)} u_{(i,j)} \ \forall \ell. \tag{6}$$

We summarize the values of $v_{(i,j)} := p_{(i,j)} u_{(i,j)}$ in Table 2.

Cell (i, j)	$\mathbf{u_{(i,j)}}$	$\mathbf{p_{(i,j)}}$	$\mathbf{v_{(i,j)}}$	**Cell (i, j)**	$\mathbf{u_{(i,j)}}$	$\mathbf{p_{(i,j)}}$	$\mathbf{v_{(i,j)}}$
$(00, 00)$	11	1	11	$(10, 00)$	1	-1	-1
$(00, 01)$	1	1	1	$(10, 01)$	4	1	4
$(00, 10)$	1	1	1	$(10, 10)$	1	1	1
$(00, 11)$	1	1	1	$(10, 11)$	1	1	1
$(01, 00)$	1	1	1	$(11, 00)$	1	1	1
$(01, 01)$	1	1	1	$(11, 01)$	1	1	1
$(01, 10)$	4	-1	-4	$(11, 10)$	1	-1	-1
$(01, 11)$	1	1	1	$(11, 11)$	11	1	11

Table 2: Values of $p_{(i,j)}$ and $v_{(i,j)}$

Equation 6 allows us to simplify Equation 4 to

$$s_{m,n} = \sum_{\ell \in \mathbb{L}^{(m,n)}} \prod_{\text{Cell } (i,j) \in \ell} v_{(i,j)}. \tag{7}$$

$s_{m,n}$ can be calculated more efficiently than in Equation 7 using the state matrix recursion introduced in Theorem 1.1.

5 Proof of Theorem 2.1

TODO induction and definition for state matrix

TODO induction for matrix powers

TODO complexity

Equation 7 has $mn2^{(m-1)(n-1)}$ multiplications.

Theorem 2.1 uses $4^2 + 4^3 + 4^4 + 4^{m-1} = 4^2 \frac{4^{m-2}-1}{4-1}$ multiplications for the construction of $A(m)$, and using the naive matrix multiplication algorithm we have $2^{3(m-1)} \lceil \log_2(n) \rceil$ multiplications to compute $A(m)^n$.

If we set $m = n$, calculating $s_{m,m}$ with Equation 7 is $O(m^2 2^{m^2})$, while Theorem 2.1 is $O(\log(m)8^m)$.

6 OLD STUFF

Proof. We consider the same vertex labeling in the proof of Theorem **??**, namely labeling a vertex by the number of polygons that contain it mod 2. Though avoided in the previous proof for clarity, it is important to think of each individual cell labeling as having its own weight, which corresponds with how many distinct tiles can map to the cell labeling.

$$\begin{array}{cccccccc}
\begin{matrix}0 - 0\\ | \quad | \\ 0 - 0\end{matrix} & \begin{matrix}0 - 0\\ | \quad | \\ 0 - 1\end{matrix} & \begin{matrix}0 - 0\\ | \quad | \\ 1 - 0\end{matrix} & \begin{matrix}0 - 0\\ | \quad | \\ 1 - 1\end{matrix} & \begin{matrix}0 - 1\\ | \quad | \\ 0 - 0\end{matrix} & \begin{matrix}0 - 1\\ | \quad | \\ 0 - 1\end{matrix} & \begin{matrix}0 - 1\\ | \quad | \\ 1 - 0\end{matrix} & \begin{matrix}0 - 1\\ | \quad | \\ 1 - 1\end{matrix} \\
w_1 & w_2 & w_3 & w_4 & w_5 & w_6 & w_7 & w_8 \\
\begin{matrix}1 - 0\\ | \quad | \\ 0 - 0\end{matrix} & \begin{matrix}1 - 0\\ | \quad | \\ 0 - 1\end{matrix} & \begin{matrix}1 - 0\\ | \quad | \\ 1 - 0\end{matrix} & \begin{matrix}1 - 0\\ | \quad | \\ 1 - 1\end{matrix} & \begin{matrix}1 - 1\\ | \quad | \\ 0 - 0\end{matrix} & \begin{matrix}1 - 1\\ | \quad | \\ 0 - 1\end{matrix} & \begin{matrix}1 - 1\\ | \quad | \\ 1 - 0\end{matrix} & \begin{matrix}1 - 1\\ | \quad | \\ 1 - 1\end{matrix} \\
w_9 & w_{10} & w_{11} & w_{12} & w_{13} & w_{14} & w_{15} & w_{16}
\end{array}$$

The proof of Theorem **??** can be seen as assigning $w_7 = w_{10} = 0$, and all other weights to 1. The usefulness of this view can be seen when considering the operation for creating the recursive definition for $A(k)$ in the previous proof. Previously, we defined the coefficient matrix V by computing $A(2)$ and $A(3)$ directly and comparing. Now with a weight assigned to individual cell labelings, we can define the values of $A(2)$ and V directly in terms of these weights.

$$A(2) = \begin{bmatrix} w_1w_1 & w_2w_5 \\ w_3w_9 & w_4w_{13} \end{bmatrix}, V = \begin{bmatrix} \frac{w_1w_1}{w_1} & \frac{w_2w_5}{w_1} & \frac{w_1w_2}{w_2} & \frac{w_2w_6}{w_2} \\ \frac{w_3w_9}{w_1} & \frac{w_4w_{13}}{w_1} & \frac{w_3w_{10}}{w_2} & \frac{w_4w_{14}}{w_2} \\ \frac{w_1w_3}{w_3} & \frac{w_2w_7}{w_3} & \frac{w_1w_4}{w_4} & \frac{w_2w_8}{w_4} \\ \frac{w_3w_{11}}{w_3} & \frac{w_4w_{15}}{w_3} & \frac{w_3w_{12}}{w_4} & \frac{w_4w_{16}}{w_4} \end{bmatrix} = \begin{bmatrix} w_1 & \frac{w_2w_5}{w_1} & w_1 & w_6 \\ \frac{w_3w_9}{w_1} & \frac{w_4w_{13}}{w_1} & \frac{w_3w_{10}}{w_2} & \frac{w_4w_{14}}{w_2} \\ w_1 & \frac{w_2w_7}{w_3} & w_1 & \frac{w_2w_8}{w_4} \\ w_{11} & \frac{w_4w_{15}}{w_3} & \frac{w_3w_{12}}{w_4} & w_{16} \end{bmatrix}.$$

With this identity, we can enumerate $t_{m,n}$ once we have proper assignments for the 16 weights. As in the previous proof, we have $w_7 = w_{10} = 0$, as again these are impossible vertex labelings for our tile set.

Next consider the cell labelings for w_1 and w_{16}. When enumerating polygon mosaics (and their messy variant), these cell labelings do not contribute to the cells of a polygon. For polygon mosaics, only the T_1 tile are permitted to not contribute to the shape of a polygon. However, in messy polygon mosaics, all 7 tiles are permitted to not contribute to the shape of the polygon, so $w_1 = w_{16} = 7$.

However, this means we now lose the uniqueness of the map from vertex labeling to messy polygon mosaics. For instance, the sub-grid vertex labelings below are now ambiguous as to whether or not they represent a polygon.

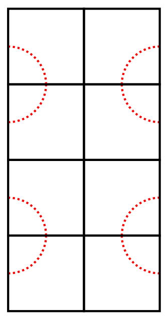
```
0 — 0 — 0     1 — 1 — 1
|   |   |     |   |   |
0 — 0 — 0     1 — 1 — 1
|   |   |     |   |   |
0 — 0 — 0     1 — 1 — 1
```

This ambiguity is explored in the following example.

Example 6.1. Consider the four vertex labelings below, along with the messy polygon mosaic on the right. Write the product of the weights of all cell labelings below each, assuming all weights not defined above are 1.

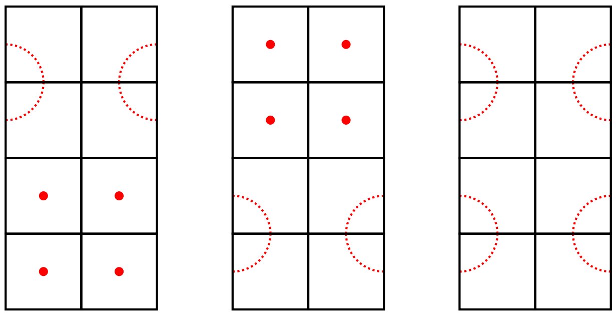
```
0 — 0 — 0     0 — 0 — 0     0 — 0 — 0     0 — 0 — 0
|   |   |     |   |   |     |   |   |     |   |   |
0 — 0 — 0     0 — 1 — 0     0 — 0 — 0     0 — 1 — 0
|   |   |     |   |   |     |   |   |     |   |   |
0 — 0 — 0     0 — 0 — 0     0 — 0 — 0     0 — 0 — 0
|   |   |     |   |   |     |   |   |     |   |   |
0 — 0 — 0     0 — 0 — 0     0 — 1 — 0     0 — 1 — 0
|   |   |     |   |   |     |   |   |     |   |   |
0 — 0 — 0     0 — 0 — 0     0 — 0 — 0     0 — 0 — 0
```

7^8 $\qquad$ 7^4 $\qquad$ 7^4 $\qquad$ 7^0



Notice that each vertex labeling could include the right-most messy polygon mosaic. In fact, the left-most vertex labeling contains all possible mosaics! If we were to add these weight products together, we would count the right messy polygon mosaic 4 times.

The double counting demonstrated in Example 6.1 motivates the following idea. If vertex labelings with an odd number of polygons are negative, then the addition of these weight products would incorporate the *inclusion-exclusion principle*, mitigating the double counting. In Example 6.1, the sum $7^8 - 7^4 - 7^4 + 7^0$ would then represent the number of mosaics that *do not* contain the following three classes of messy polygon mosaics, where cells that can be any tile are marked with a dot.



Therefore, the sum over the products for all vertex labelings, where the product is negative if the vertex labeling represents an odd number of polygons in the mosaic, would be the number of mosaics that *do not* include messy polygon mosaics.

We can accomplish this by finding a weight assignment such that the product over the cell labeling weights of *any* single polygon equals -1. It is not obvious that such an assignment can even be found!

Luckily such assignments exist. The proof of this fact can be found in the Appendix, and choosing an assignment gives us the following weight assignments.

$$\begin{array}{cccccccc}
\begin{array}{ccc}0&\text{---}&0\\|&&|\\0&\text{---}&0\end{array} &
\begin{array}{ccc}0&\text{---}&0\\|&&|\\0&\text{---}&1\end{array} &
\begin{array}{ccc}0&\text{---}&0\\|&&|\\1&\text{---}&0\end{array} &
\begin{array}{ccc}0&\text{---}&0\\|&&|\\1&\text{---}&1\end{array} &
\begin{array}{ccc}0&\text{---}&1\\|&&|\\0&\text{---}&0\end{array} &
\begin{array}{ccc}0&\text{---}&1\\|&&|\\0&\text{---}&1\end{array} &
\begin{array}{ccc}0&\text{---}&1\\|&&|\\1&\text{---}&0\end{array} &
\begin{array}{ccc}0&\text{---}&1\\|&&|\\1&\text{---}&1\end{array} \\
w_1=7 & w_2=1 & w_3=1 & w_4=1 & w_5=1 & w_6=1 & w_7=0 & w_8=1 \\
\begin{array}{ccc}1&\text{---}&0\\|&&|\\0&\text{---}&0\end{array} &
\begin{array}{ccc}1&\text{---}&0\\|&&|\\0&\text{---}&1\end{array} &
\begin{array}{ccc}1&\text{---}&0\\|&&|\\1&\text{---}&0\end{array} &
\begin{array}{ccc}1&\text{---}&0\\|&&|\\1&\text{---}&1\end{array} &
\begin{array}{ccc}1&\text{---}&1\\|&&|\\0&\text{---}&0\end{array} &
\begin{array}{ccc}1&\text{---}&1\\|&&|\\0&\text{---}&1\end{array} &
\begin{array}{ccc}1&\text{---}&1\\|&&|\\1&\text{---}&0\end{array} &
\begin{array}{ccc}1&\text{---}&1\\|&&|\\1&\text{---}&1\end{array} \\
w_9=-1 & w_{10}=0 & w_{11}=1 & w_{12}=1 & w_{13}=1 & w_{14}=1 & w_{15}=-1 & w_{16}=7
\end{array}$$

This immediately gives us a way to construct an analagous definition for $A(k+1)$ given $A(k)$. Once we write $A(k) = \begin{bmatrix} A_{0,0} & A_{0,1} \\ A_{1,0} & A_{1,1} \end{bmatrix}$, we have

$$A(2) = \begin{bmatrix} 7^2 & 1 \\ -1 & 1 \end{bmatrix}, V = \begin{bmatrix} 7 & \frac{1}{7} & 7 & 1 \\ -\frac{1}{7} & 1 & 0 & 1 \\ 7 & 0 & 7 & 1 \\ 1 & -1 & 1 & 7 \end{bmatrix}$$

Subsituting V into Equation **??** gives the result.

$\square$

7 Extensions

Hong and Oh [2] study the mosaic system with the tile set $\mathbb{T}^* = \{T_0, \ldots, T_7\}$. This tile set constructs shapes we call *polygons*[2]. If we let $p_{m,n}$ be the number of polygon mosaics of size

[2]Polygons are more commonly called "self-avoiding polygons" in the literature to emphasize their relationship with self-avoiding walks.

(m, n), Hong and Oh showed the following results[3].

Theorem 7.1 ([2]). *The number of polygon mosaics of size* (m, n) $p_{m,n}$ *for* $m, n \geq 2$ *has*

$$2^{m+n-3}\left(\frac{17}{10}\right)^{(m-2)(n-2)} \leq p_{m,n} \leq 2^{m+n-3}\left(\frac{31}{16}\right)^{(m-2)(n-2)}.$$

Though not stated in Hong and Oh [2], $p_{m,n}$ is exactly enumerated by Theorem 1.1 by replacing the 4 in the definition of O_{k+1} with a 0.The array $p_{n,m}$ is A181245 on the OEIS [4, OEIS].

TODO

8 Acknowledgements

The authors would like to thank Michael Maltenfort for the edits, improvements and ideas for this paper.

References

[1] Dooho Choi et al. "Quantum knot mosaics and bounds of the growth constant". In: *Reviews in Mathematical Physics* 36.10 (2024), p. 2450025. DOI: 10.1142/S0129055X24500259. eprint: https://doi.org/10.1142/S0129055X24500259. URL: https://doi.org/10.1142/S0129055X24500259.

[2] Kyungpyo Hong and Seungsang Oh. "Bounds on Multiple Self-avoiding Polygons". In: *Canadian Mathematical Bulletin* 61.3 (Sept. 2018), pp. 518–530. ISSN: 1496-4287. DOI: 10.4153/cmb-2017-072-x. URL: http://dx.doi.org/10.4153/CMB-2017-072-x.

[3] Samuel J. Lomonaco and Louis H. Kauffman. "Quantum knots and mosaics". In: *Quantum Information Processing* 7.2 (2008), pp. 85–115. DOI: 10.1007/s11128-008-0076-7. URL: https://doi.org/10.1007/s11128-008-0076-7.

[4] OEIS Foundation Inc. *The On-Line Encyclopedia of Integer Sequences*. Published electronically at http://oeis.org.

[5] Seungsang Oh. "Domino tilings of the expanded Aztec diamond". In: *DISCRETE MATHEMATICS* 341.4 (Apr. 2018), pp. 1185–1191. ISSN: 0012-365X. DOI: 10.1016/j.disc.2017.10.016.

[6] Seungsang Oh. "Quantum knot mosaics and the growth constant". In: *Topology and its Applications* 210 (2016), pp. 311–316. ISSN: 0166-8641. DOI: https://doi.org/10.1016/j.topol.2016.08.011. URL: https://www.sciencedirect.com/science/article/pii/S0166864116301857.

[7] Seungsang Oh. "State matrix recursion method and monomer-dimer problem". In: *DISCRETE MATHEMATICS* 342.5 (May 2019), pp. 1434–1445. ISSN: 0012-365X. DOI: 10.1016/j.disc.2019.01.022.

[3]The authors did not consider the mosaic containing all T_0 tiles a polygon mosaic, and so define $p_{m,n}$ as one less than what we define.

[8] Seungsang Oh and Youngin Kim. “Growth rate of quantum knot mosaics”. In: *Quantum Information Processing* 18.8 (2019), p. 238. DOI: `10.1007/s11128-019-2353-z`. URL: `https://doi.org/10.1007/s11128-019-2353-z`.

[9] Seungsang Oh et al. “Quantum knots and the number of knot mosaics”. In: *Quantum Information Processing* 14.3 (2015), pp. 801–811. DOI: `10.1007/s11128-014-0895-7`. URL: `https://doi.org/10.1007/s11128-014-0895-7`.

9 Appendix

We demonstrate that weight assignments exist such that the product over the cell labeling weights of *all* single polygons equals -1. We do this by first asserting that the product of the weights associated with the smallest polygon multiply to -1, ie. $w_2w_3w_5w_9 = -1$.

Lemma 9.1. *One can construct all larger polygons from the smallest polygon using a finite set of transformations S.*

Proof. TODO Something about changing vertex values □

This is because one can find $w_1, \ldots, w_{16}$ so that the following two constraints hold:

Constraint 9.2. *The weights associated with the smallest polygon multiply to -1, ie.* $w_2w_3w_5w_9 = -1$.

Constraint 9.3. *All transformations in S preserve the weight product of a changed polygon.*

Constraint 9.2 and Constraint 9.3 amount to a series of constraints on the values of w_i. Choosing a solution set from these constraints gives the following weights.

Flipping the parity of a single vertex in a vertex labeling changes the 4 surrounding cells. This creates a constraint on a subset of $w_1, \ldots, w_{16}$.

The flipping of parity of a single vertex results in 2 distinct types of constraints. Let a constraint of *Type 1* be a parity flip that does not change the number of polygons represented in the vertex labeling. For example, consider the following flip of the center vertex in the following sub vertex labeling.

$$\begin{array}{ccccc} 0 & — & 0 & — & 0 \\ | & & | & & | \\ 0 & — & 0 & — & 1 \\ | & & | & & | \\ 0 & — & 1 & — & 1 \end{array} \to \begin{array}{ccccc} 0 & — & 0 & — & 0 \\ | & & | & & | \\ 0 & — & 1 & — & 1 \\ | & & | & & | \\ 0 & — & 1 & — & 1 \end{array}$$

As this does not change the associated number of polygons in the larger vertex labeling, we want this to preserve the sign of the weight product. This gives the following associated constraint.

$$\mathrm{sign}(w_1w_2w_5w_9) = \mathrm{sign}(w_2w_4w_6w_{16}).$$

Now let a constraint of *Type 2* be a parity flip that does change the number of polygons. For example, consider flipping the center vertex of the following portion of a vertex labeling.

$$\begin{array}{ccccc} 0 & — & 0 & — & 0 \\ | & & | & & | \\ 1 & — & 0 & — & 1 \\ | & & | & & | \\ 0 & — & 0 & — & 0 \end{array} \to \begin{array}{ccccc} 0 & — & 0 & — & 0 \\ | & & | & & | \\ 1 & — & 1 & — & 1 \\ | & & | & & | \\ 0 & — & 0 & — & 0 \end{array}$$

The above transformation corresponds with *either* two distinct polygons joining into one polygon *or* one polygon splitting into two distinct polygons. In either case, we want the sign of the product to switch. This corresponds with the following constraint.

$$\operatorname{sign}(w_3w_2w_9w_5) = -\operatorname{sign}(w_4w_4w_{13}w_{13}).$$

All Type 1 constraints are as follows. For the following set of equations, assume the equals sign (=) means *only* equal in sign.

$$\begin{array}{rlrlrl}
w_1w_1w_2w_3 &= w_2w_3w_6w_{11} & w_1w_1w_2w_4 &= w_2w_3w_6w_{12} & w_1w_1w_4w_3 &= w_2w_3w_8w_{11} \\
w_1w_1w_4w_4 &= w_2w_3w_8w_{12} & w_1w_2w_1w_5 &= w_2w_4w_5w_{13} & w_1w_2w_1w_6 &= w_2w_4w_5w_{14} \\
w_1w_2w_2w_8 &= w_2w_4w_6w_{16} & w_1w_2w_4w_8 &= w_2w_4w_8w_{16} & w_3w_1w_9w_1 &= w_4w_3w_{13}w_9 \\
w_3w_1w_{11}w_1 &= w_4w_3w_{15}w_9 & w_3w_1w_{12}w_3 &= w_4w_3w_{16}w_{11} & w_3w_1w_{12}w_4 &= w_4w_3w_{16}w_{12} \\
w_3w_2w_{12}w_8 &= w_4w_4w_{16}w_{16} & w_1w_6w_1w_5 &= w_2w_8w_5w_{13} & w_1w_6w_1w_6 &= w_2w_8w_5w_{14} \\
w_1w_6w_2w_8 &= w_2w_8w_6w_{16} & w_1w_6w_4w_8 &= w_2w_8w_8w_{16} & w_3w_6w_{12}w_8 &= w_4w_8w_{16}w_{16} \\
w_5w_9w_1w_1 &= w_6w_{11}w_5w_9 & w_5w_{13}w_1w_1 &= w_6w_{15}w_5w_9 & w_5w_{14}w_1w_5 &= w_6w_{16}w_5w_{13} \\
w_5w_{14}w_1w_6 &= w_6w_{16}w_5w_{14} & w_5w_{14}w_2w_8 &= w_6w_{16}w_6w_{16} & w_5w_{14}w_4w_8 &= w_6w_{16}w_8w_{16} \\
w_{11}w_1w_9w_1 &= w_{12}w_3w_{13}w_9 & w_{11}w_1w_{11}w_1 &= w_{12}w_3w_{15}w_9 & w_{11}w_1w_{12}w_3 &= w_{12}w_3w_{16}w_{11} \\
w_{11}w_1w_{12}w_4 &= w_{12}w_3w_{16}w_{12} & w_{11}w_2w_{12}w_8 &= w_{12}w_4w_{16}w_{16} & w_{11}w_6w_{12}w_8 &= w_{12}w_8w_{16}w_{16} \\
w_{13}w_9w_1w_1 &= w_{14}w_{11}w_5w_9 & w_{15}w_9w_9w_1 &= w_{16}w_{11}w_{13}w_9 & w_{15}w_9w_{11}w_1 &= w_{16}w_{11}w_{15}w_9 \\
w_{15}w_9w_{12}w_3 &= w_{16}w_{11}w_{16}w_{11} & w_{15}w_9w_{12}w_4 &= w_{16}w_{11}w_{16}w_{12} & w_{13}w_{13}w_1w_1 &= w_{14}w_{15}w_5w_9 \\
w_{13}w_{14}w_1w_5 &= w_{14}w_{16}w_5w_{13} & w_{13}w_{14}w_1w_6 &= w_{14}w_{16}w_5w_{14} & w_{13}w_{14}w_2w_8 &= w_{14}w_{16}w_6w_{16} \\
w_{13}w_{14}w_4w_8 &= w_{14}w_{16}w_8w_{16} & w_{15}w_{13}w_9w_1 &= w_{16}w_{15}w_{13}w_9 & w_{15}w_{13}w_{11}w_1 &= w_{16}w_{15}w_{15}w_9 \\
w_{15}w_{13}w_{12}w_3 &= w_{16}w_{15}w_{16}w_{11} & w_{15}w_{13}w_{12}w_4 &= w_{16}w_{15}w_{16}w_{12} & w_{15}w_{14}w_9w_5 &= w_{16}w_{16}w_{13}w_{13} \\
w_{15}w_{14}w_9w_6 &= w_{16}w_{16}w_{13}w_{14} & w_{15}w_{14}w_{11}w_5 &= w_{16}w_{16}w_{15}w_{13} & w_{15}w_{14}w_{11}w_6 &= w_{16}w_{16}w_{15}w_{14}
\end{array}$$

Similarly, all Type 2 constraints are as follows. Again, for the following set of equations, assume the equals sign (=) means *only* equal in sign.

$$\begin{array}{rrr}
-w_3w_2w_9w_5 = w_4w_4w_{13}w_{13} & -w_3w_2w_9w_6 = w_4w_4w_{13}w_{14} & -w_3w_2w_{11}w_5 = w_4w_4w_{15}w_{13} \\
-w_3w_2w_{11}w_6 = w_4w_4w_{15}w_{14} & -w_3w_6w_9w_5 = w_4w_8w_{13}w_{13} & -w_3w_6w_9w_6 = w_4w_8w_{13}w_{14} \\
-w_3w_6w_{11}w_5 = w_4w_8w_{15}w_{13} & -w_3w_6w_{11}w_6 = w_4w_8w_{15}w_{14} & -w_5w_9w_2w_3 = w_6w_{11}w_6w_{11} \\
-w_5w_9w_2w_4 = w_6w_{11}w_6w_{12} & -w_5w_9w_4w_3 = w_6w_{11}w_8w_{11} & -w_5w_9w_4w_4 = w_6w_{11}w_8w_{12} \\
-w_5w_{13}w_2w_3 = w_6w_{15}w_6w_{11} & -w_5w_{13}w_2w_4 = w_6w_{15}w_6w_{12} & -w_5w_{13}w_4w_3 = w_6w_{15}w_8w_{11} \\
-w_5w_{13}w_4w_4 = w_6w_{15}w_8w_{12} & -w_{11}w_2w_9w_5 = w_{12}w_4w_{13}w_{13} & -w_{11}w_2w_9w_6 = w_{12}w_4w_{13}w_{14} \\
-w_{11}w_2w_{11}w_5 = w_{12}w_4w_{15}w_{13} & -w_{11}w_2w_{11}w_6 = w_{12}w_4w_{15}w_{14} & -w_{11}w_6w_9w_5 = w_{12}w_8w_{13}w_{13} \\
-w_{11}w_6w_9w_6 = w_{12}w_8w_{13}w_{14} & -w_{11}w_6w_{11}w_5 = w_{12}w_8w_{15}w_{13} & -w_{11}w_6w_{11}w_6 = w_{12}w_8w_{15}w_{14} \\
-w_{13}w_9w_2w_3 = w_{14}w_{11}w_6w_{11} & -w_{13}w_9w_2w_4 = w_{14}w_{11}w_6w_{12} & -w_{13}w_9w_4w_3 = w_{14}w_{11}w_8w_{11} \\
-w_{13}w_9w_4w_4 = w_{14}w_{11}w_8w_{12} & -w_{13}w_{13}w_2w_3 = w_{14}w_{15}w_6w_{11} & -w_{13}w_{13}w_2w_4 = w_{14}w_{15}w_6w_{12} \\
-w_{13}w_{13}w_4w_3 = w_{14}w_{15}w_8w_{11} & -w_{13}w_{13}w_4w_4 = w_{14}w_{15}w_8w_{12} &
\end{array}$$

Solving all Type 1 and Type 2 constraints gives the following solution set.

w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}	w_{11}	w_{12}	w_{13}	w_{14}	w_{15}	w_{16}
7	-1	-1	-1	-1	-1	0	-1	1	0	-1	-1	-1	-1	1	7
7	-1	-1	-1	-1	1	0	1	1	0	1	1	-1	1	-1	7
7	-1	-1	-1	1	-1	0	-1	-1	0	-1	-1	-1	1	-1	7
7	-1	-1	-1	1	1	0	1	-1	0	1	1	-1	-1	1	7
7	-1	-1	1	-1	-1	0	1	1	0	-1	1	1	1	-1	7
7	-1	-1	1	-1	1	0	-1	1	0	1	-1	1	-1	1	7
7	-1	-1	1	1	-1	0	1	-1	0	-1	1	1	-1	1	7
7	-1	-1	1	1	1	0	-1	-1	0	1	-1	1	1	-1	7
7	-1	1	-1	-1	-1	0	-1	-1	0	-1	1	-1	-1	-1	7
7	-1	1	-1	-1	1	0	1	-1	0	1	-1	-1	1	1	7
7	-1	1	-1	1	-1	0	-1	1	0	-1	1	-1	1	1	7
7	-1	1	-1	1	1	0	1	1	0	1	-1	-1	-1	-1	7
7	-1	1	1	-1	-1	0	1	-1	0	-1	-1	1	1	1	7
7	-1	1	1	-1	1	0	-1	-1	0	1	1	1	-1	-1	7
7	-1	1	1	1	-1	0	1	1	0	-1	-1	1	-1	-1	7
7	-1	1	1	1	1	0	-1	1	0	1	1	1	1	1	7
7	1	-1	-1	-1	-1	0	1	-1	0	-1	-1	-1	-1	-1	7
7	1	-1	-1	-1	1	0	-1	-1	0	1	1	-1	1	1	7
7	1	-1	-1	1	-1	0	1	1	0	-1	-1	-1	1	1	7
7	1	-1	-1	1	1	0	-1	1	0	1	1	-1	-1	-1	7
7	1	-1	1	-1	-1	0	-1	-1	0	-1	1	1	1	1	7
7	1	-1	1	-1	1	0	1	-1	0	1	-1	1	-1	-1	7
7	1	-1	1	1	-1	0	-1	1	0	-1	1	1	-1	-1	7
7	1	-1	1	1	1	0	1	1	0	1	-1	1	1	1	7
7	1	1	-1	-1	-1	0	1	1	0	-1	1	-1	-1	1	7
7	1	1	-1	-1	1	0	-1	1	0	1	-1	-1	1	-1	7
7	1	1	-1	1	-1	0	1	-1	0	-1	1	-1	1	-1	7
7	1	1	-1	1	1	0	-1	-1	0	1	-1	-1	-1	1	7
7	1	1	1	-1	-1	0	-1	1	0	-1	-1	1	1	-1	7
7	1	1	1	-1	1	0	1	1	0	1	1	1	-1	1	7
7	1	1	1	1	-1	0	-1	-1	0	-1	-1	1	-1	1	7
7	1	1	1	1	1	0	1	-1	0	1	1	1	1	-1	7

Any of these assignments are sufficient for calculating $t_{m,n}$.